How Can We Make Language Models Better at Handling the Diversity and Variability of Natural Languages ?

Benjamin Muller

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Inria
inventeurs du monde numérique

Sorbonne Université

Ecole Doctorale Informatique, Télécommunications et Electronique
ED130

Inria de Paris / Équipe-Projet ALMAnaCH

Thèse de doctorat

Discipline : Informatique

x Présentée par

Benjamin Muller

Pour obtenir le grade universitaire de

Docteur de Sorbonne Université

How Can We Make Language Models Better at Handling the Diversity and Variability of Natural Languages ?

Présentée et soutenue publiquement le 17 novembre 2022 devant le jury composé de :

Lilja Øvrelid	University of Oslo	Rapportrice & Examinatrice
Yoav Goldberg	Bar Ilan University	Rapporteur & Examinateur
Natalie Schluter	IT University of Copenhagen	Examinatrice
Benjamin Piwowarski	Sorbonne Université	Examinateur
Benoît Sagot	Inria - ALMAnaCH	Directeur
Djamé Seddah	Inria - ALMAnaCH	Co-Encadrant

Acknowledgment

I wish to thank Benjamin Piwowarski, Natalie Schulter, Lilja Øvrelid, and Yoav Goldberg for generously accepting to review this manuscript and participating in my Ph.D. committee.

I sincerely thank Benoît Sagot and Djamé Seddah, my Ph.D. supervisors. I want to thank them for their guidance, care, and the expertise and knowledge they have shared with me for the past four years. I want to thank them for the freedom they have given me and the opportunities and impactful research projects I was able to work on under their guidance.

I am also incredibly grateful for the passionate and talented collaborators and mentors I was lucky to work with during these four years. Specifically, Ganesh Jawahar, Pedro Ortiz Suarez, Louis Martin, Yanai Elazar, Antonis Anastasopoulos, Rob Van Der Goot, Hila Gonen, and Omer Goldman. I have learned a lot from them and with them, and I want to thank them all warmly for it.

I want to thank the entire ALMAnaCH team from INRIA Paris. I was one of the first member student in the team, and I am pleased I was able to witness and help ALMAnaCH grow.

I want to thank ENSAE Paris for hiring me as a lecturer to teach NLP. I want to give a shout out to Gaël Guibon, Ghazi Felhi, Roman Castagné, Matthieu Futeral-Peter, Karim Lasri, and Salomé Do for helping me improve this course.

I was also lucky to intern at several companies these past four years. I want to thank Sid, Deepanshu, and Jean-Philippe from Apple, Luca from AI2, Rik, Eric, and Alessandro from Amazon, and John, Sebastian, Jon, Tom, Livio, and Roee from Google. I wish to thank them all deeply for the opportunities they have given me and for their support and care that helped me grow and learn.

Je remercie infiniment mes amis et ma famille pour leur soutient ces quatres dernières années. Un remerciement particulier à mes amis d'enfance Maxime, Thomas et Cyril; à mes anciens colocataires de la rue de Belleville Yannis et David. Merci à mes parents, Joël et Pascale, pour leur amour et soutient sans faille où que je sois. Une pensée enfin à ma grand-mère bientôt centenaire qui aurait aimée assister à ma soutenance.

I want to finally thank my partner, Padmini, for being here these past three years. I want to thank her for her care, love, and emotional support during countless key moments in Singapore, London, Paris, and the US.

Abstract

Deep Learning techniques applied to Natural Language Processing (NLP) have led to impressive empirical progress in recent years. In essence, this progress is due to the development of better-contextualized representations of textual data that can be easily used — or transferred — for a wide variety of NLP tasks. In their most recent and popular forms, these models consist of large-scale deep-learning language models, first pretrained on a large quantity of raw data and then adapted to specific tasks. These language models are now essential for search engines, question-answering pipelines, machine translation systems, etc.

However, these models usually require substantial computing power and large amounts of raw textual data. This makes natural language's inherent diversity and variability a vivid challenge in NLP. Indeed, collecting large datasets for low-resource languages is challenging and costly, and training models from scratch for every domain and language is unreasonable in practice. Additionally, understanding the behavior of deep learning-based models is intrinsically tricky, making the development of more cost-effective techniques even more challenging.

For these reasons, we focus on the following question: "How can we make language models better at handling the variability and diversity of natural languages?".

As a starting step, we explore the generalizability of language models by building one of the first large-scale replication of a BERT model for a non-English language. We analyze the critical training ingredients and show that it can achieve state-of-the-art performance with only a few gigabytes of diverse data.

Our results raise the question of using these language models on highly-variable domains such as these found in user-generated content. Focusing on domain-gap reduction via lexical normalization, we show that this task can be addressed accurately with BERT-like models. However, we show that it only partially helps downstream performance. In consequence, we focus on direct adaptation techniques using what we refer to as *representation transfer* and explore challenging settings such as the zero-shot setting, low-resource language varieties like Bambara or Uyghur, and highly variable and non-standardized code-mixed dialects such as a North-African Arabic dialect written in the Latin script. We show that multilingual language models can be adapted and used efficiently with low-resource languages, even with the ones unseen during pretraining, and that the script is a critical component in this adaptation.

NLP technologies are becoming increasingly critical to accessing knowledge, connecting with our friends, and extracting meaningful information from large quantities of text. In this thesis, we present concrete and usable solutions to ensure that we can build accurate NLP systems for the most significant number of domains and languages at a reasonable cost.

Contents

1 Introduction

1.1 Context

Natural Language Processing (NLP) technologies have become ubiquitous in our daily lives. They impact what content we see online through search engines, what information we learn about events with recommender systems, and how we communicate with our friends with messaging applications. It is now possible to interact with computers in an even more natural way through voice. In short, for an increasing number of people worldwide, NLP technologies are now taking a significant role in shaping their worldviews and daily experiences.

Although the mass adoption of NLP technologies has happened in recent years, the design and development of NLP systems are about three-quarters of a century old. As early as 1947, Warren Weaver, a former student of the Information Theory pioneer Claude Shannon, described his idea of using computers to build automatic translation systems. In his *Memorandum on Translation* (Weaver, 1952), Weaver detailed his idea. He proposed to use co-occurrence statistics, which he referred to as "statistical semantic" and the typological rules that govern languages to build an automatic translation system. In the aftermath of World War II, Weaver notably saw the design of machine translation systems as a force for peace. By helping people from a wide diversity of nations speaking different languages understand each other, Weaver saw NLP as a way to help "the constructive and peaceful future of the planet".

1.1.1 A Minimalist View of the Standard NLP pipeline

This vision was followed by numerous attempts in machine translation and other NLP tasks such as question answering, named entity recognition, syntactic parsing, etc. Al-

though these attempts differ in the linguistic resources they require and the probabilistic methods they rely on, as different as they may seem, they can all be illustrated in a minimalist manner with the diagram in Figure 1.1. Indeed, any NLP technique starts with a raw signal, i.e., textual data (reading the chart from bottom to top).

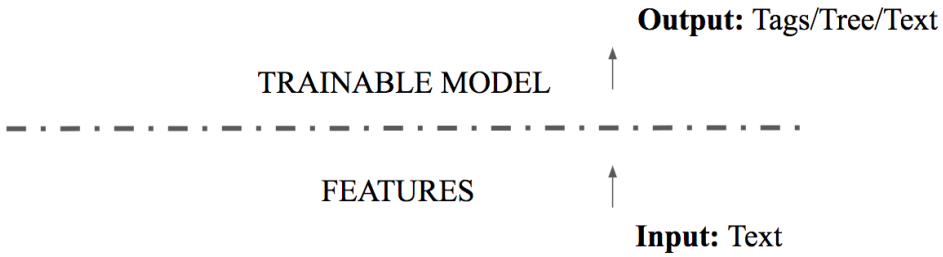


Figure 1.1: Minimalist representation of "any" data-driven NLP pipeline

Using this raw signal, the first step is always to define a set of modeling units or features, a process that we refer to as *featurization*. This step can be as trivial as considering that any character in the text is a modeling unit. Sometimes, it can require more carefully-defined rules to segment this text and possibly associate each segmented sequence to categories. These categories may be linguistically motivated (e.g., part-of-speech tags, named entities) or statistically motivated (e.g., considering that *San Francisco* is a single entity based on the number of occurrences in a dataset).

This first step is typically followed by a second step which consists in feeding these features to what we commonly refer to as *a model*. This model is designed to take the sequence of features as input and make a prediction (e.g., translating into another language, predicting a category, identifying the grammatical relations, etc.). The models we will describe in this thesis are "trainable", i.e., their behavior is defined by a training algorithm that uses data to estimate the best prediction given an input sequence. How these models are defined — a process referred to as the *parametrization* — and how we train them have varied very importantly in the past decades. Among many modeling paradigm, we can point to count-based approaches (Manning and Schütze, 2002), probabilistic graphical model framework (Manning and Schütze, 2002; Koller and Friedman, 2009), tree-based algorithms (James et al., 2013), and deep learning (Goodfellow et al., 2016), the most

popular and (empirically) successful framework of the recent years for NLP. We note that each modeling approach has usually re-defined the limit between what is a feature and what is part of the trainable model. In the early days of statistical NLP, a lot of work was dedicated to engineering the right features for the task of interest. In the last ten years, deep learning brought impressive empirical progress across nearly all NLP tasks. Notably, the recent and most accurate deep-learning models use elementary featurization techniques (the text is split into sequences of characters based on their frequency, §4.2.5) while a lot of attention is put on the type of deep learning architecture, the training objective, and the optimization of the model (§4.4). Additionally, deep learning is a powerful framework to do what is referred to as *transfer learning*. In short, transfer learning consists in using a model trained on a given task and dataset for another task and dataset. As illustrated with the success of the BERT model (Devlin et al., 2018a) (§ 4.6.3), deep learning-based language models (Peters et al., 2018a; Devlin et al., 2018a) can be used to build state-of-the-art NLP systems for nearly any task.

1.1.2 Six Historical Trends up to Today's Modern NLP

Historically, this success relies on the convergence of six chronological trends that we summarize here:

More powerful hardware First, the performance of computers[1] (*hardware*) has increased exponentially since the first computers in the 1930s. For instance, the ENIAC (Burks, 1947) could deliver 0.0000000385 GFLOPs[2] and had a cost of about 7,195,000 USD[3] which amounts to 1.88 quadrillion USD / GFLOPs. Nowadays, a standard computer built with an 11th Generation Intel Core i7[4] can deliver approximately up to 50 GFLOPs which amounts to 0.05 USD/GFLOPs. In recent years, specialized hardware

[1] We use the FLOPs as a metric of reference to report the performance of computers. A FLOPs is a standard metric to measure the performance of a computer. It corresponds to the number of floating-point operations per second. Regarding price, we report the USD/GFLOPs corresponding to the GFLOPs available per USD. Unless explicitly stated, the reported number only accounts for the price of the machine and not the operational cost (electricity, maintenance, etc.)

[2] GFLOPs refers to GigaFLOPs which amounts to 10^9 FLOPs.

[3] We account for inflation in all the number reported, and all prices are listed in 2019 USD.

[4] https://www.hp.com/us-en/shop/pdp/omen-by-hp-laptop-16-b0014nr

such as Graphical Processing Units (GPUs) (Nickolls and Dally, 2010) and Tensor Processing Units (TPUs) (Jouppi et al., 2021), have led to significant progress (in the face of the diminishing Moore's law (Hennessy and Patterson, 2019)) and delivered increasingly powerful hardware for deep learning models.

Better Software Along with this progress in hardware, programming languages (*software*) also made tremendous progress in efficiently using the computing power available and creating programs that are easier to write, test, maintain and share. Progress in software provided essential building blocks to build better NLP systems. One of the first programming languages used for large-scale scientific projects is Fortran[5] developed in the 1950s by IBM. In the 1980s, C (Ritchie et al., 1978; Kernighan and Ritchie, 1988) and C++ (Stroustrup, 1986) programming languages became popular among developers. In the past 15 years, the Python[6] programming language has become very popular for machine learning projects. With the advance of Graphics Processing Units (GPUs) computing, CUDA[7][8] was released by NVIDIA in 2007 as a C-style programming language for GPUs. Based on CUDA, C++ and python, multiple coding frameworks were developed supporting automatic differentiation such as TensorFlow (Abadi et al., 2016), PyTorch (Paszke et al., 2019) and JAX (Frostig et al., 2018). Finally, sharing model parameters has become increasingly valuable for researchers and engineers in recent years. This led to the success of the transformers library (Wolf et al., 2020), which makes the training and sharing of deep learning models easier.

Larger and More Diverse Linguistic Resources Building NLP systems requires linguistic resources. These resources are necessary to train and, even more importantly, to evaluate NLP systems. In a nutshell, before the 90s, most works in NLP was focused on building rule-based systems. Trainable NLP systems started receiving a lot of attention in the 90s with the release of large datasets and lexical resources such as the Penn Treebank (Marcus et al., 1993), the WordNet (Miller, 1995). In the 2000s and 2010s, the scale, diversity, and number of linguistic resources and datasets increased

[5] https://www.ibm.com/ibm/history/ibm100/us/en/icons/fortran/
[6] https://www.python.org/about/quotes/
[7] https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html
[8] CUDA stands for Compute Unified Device Architecture

exponentially. Among many, the Europarl dataset (Koehn, 2005), the Universal Dependency project (Nivre et al., 2015, 2016, 2018), a framework and collection of treebanks in 100+ languages, the SQUAD datasets — a large scale question answering dataset — and finally very large scale corpora of raw data usually crawled from the Web such as Wikipedia dumps[9] (Attardi, 2015), the OSCAR corpora (Ortiz Suárez et al., 2019), CCNET (Wenzek et al., 2020) and mC4 (Xue et al., 2021) were keys in the development of statistical and deep learning-based systems for NLP. We will detail in chapter 4 the resources we use in this thesis.

Richer Representation Methods of Textual Data In addition, the success of modern NLP systems emerged from the progress made in building better models. The progress made relied on the better vector representation of textual data. Methods that use explicit linguistic categories of sequences of words were outperformed by purely data-driven approaches (Mikolov et al., 2013b; Peters et al., 2018a; Devlin et al., 2018b) such as word2vec, ELMo, and BERT. In section 4.3, we will present the relevant related work that contributed to the progress made in building better text representation.

More Stable and Robust Optimization of Deep Learning Architectures Deep Learning models are challenging to train. Indeed, their training objective is non-convex, and very little optimization theory can be used in practice to train large models. It took decades of empirical investigation to understand how to train large-scale deep learning models successfully. Among many tricks and methods, parameter initialization (Glorot and Bengio, 2010; He et al., 2015), regularization techniques such as Dropout and Layer normalization (Srivastava et al., 2014; Ba et al., 2016) and finally, more stable Stochastic Gradient Descent optimization algorithm such as Adam (Kingma and Ba, 2015) contributed to the better generalization of deep learning models. We present the deep learning methods used in this thesis in section 4.4.

More Expressive and Efficient Task-Specific Models Finally, based on the five trends described above, NLP benefited from progress made in task-specific modeling. Initially built on simple count-based models, NLP systems evolved toward richer

[9] https://meta.wikimedia.org/wiki/Data_dumps

probabilistic frameworks such as Maximum Entropy or Graphical Models. In the last ten years, Deep Learning architectures such as Recurrent Neural Network (firstly introduced by (Rumelhart et al., 1985; Hochreiter and Schmidhuber, 1997)) and Transformers (introduced by Vaswani et al. (2017)) pretrained on large datasets with language modeling objectives and a lot of computing power improved the state-of-the-art performance significantly for nearly all NLP tasks (Devlin et al., 2018b). We present the critical related work that contributed to the progress made in task-specific modeling in section 4.6.

1.2 Motivations and Research Question

Based on these six historical trends, scaling up deep learning-based language models- mainly in model size, training dataset size, and training computing power- has become one of the main driving forces of empirical progress in NLP. As illustrated in (Peters et al., 2018b; Devlin et al., 2018a; Brown et al., 2020; Zhang et al., 2022; Chowdhery et al., 2022), it entailed better downstream performance and better zero-shot and few-shot abilities across a great variety of NLP tasks.

This observation supports the three core motivations of the work done for this thesis:

Toward More Cost-Effective NLP Systems First, transformers-based NLP systems are costly to build. They need a lot of computing power to be trained, which is financially and environmentally costly. As reported by Strubell et al. (2019), training a BERT model[10] in the US amounts to approximately 0.65 tCO2e,[11] which corresponds to about one passenger trip from Paris to Miami on average. The more recent 540 billion parameters PaLM language model (Chowdhery et al., 2022) led to the emission of 271.43 tCO2e, which amounts roughly to three trips from New York and San Francisco for a passenger jet. When it got published, the cost of pretraining the GPT-3 model was estimated to be 4.6 million USD.[12] Additionally, collecting and annotating data is very

[10] BERT-base model of 110M parameters.

[11] tCO2e stands for tonnes of carbon dioxide equivalent or CO2 equivalent. It is a measure "used to compare the emissions from various greenhouse gases based on their global-warming potential, by converting amounts of other gases to the equivalent amount of carbon dioxide with the same global warming potential" (cf. https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Thematic_glossaries)

[12] https://lambdalabs.com/blog/demystifying-gpt-3/

costly. Seddah et al. (2020) reported that the total annotation cost of a North African Arabic dialect treebank was around 87k USD for 1,500 sentences.

Consequently, it is unreasonable to train models from scratch for every new use case and language. One approach is to design systems that do not require newly collected datasets for every new domain and language. In this thesis, we will study cross-lingual transfer techniques that address this challenge and allow practitioners to reuse large-scale models for many languages.

Toward More Robust NLP Models Language varies based on the context. For instance, Wikipedia documents differ from what is found on social media. Wikipedia documents typically do not include the first person or imperative sentences, while social media data do. Furthermore, User Generated Content (UGC) found online varies much more in terms of syntactic structure and style than encyclopedic text (Foster, 2010a; Seddah et al., 2012; Eisenstein, 2013a; Michel and Neubig, 2018). For these reasons, designing NLP systems that are robust to language variability is critical for the success of NLP technologies in practice. Building NLP models that can cope with the variability found in UGC is necessary to build models that can, for instance, detect hateful content online (Zhang and Luo, 2019; Laaksonen et al., 2020) or limit the spread of misinformation (Álvaro Figueira and Oliveira, 2017; Asr and Taboada, 2019).

Toward Multilingual-First NLP Systems Finally, the progress made in language modeling in recent years has mainly been focused on the English language and a few high-resource languages (Joshi et al., 2020b; Blasi et al., 2022). There are thousands of languages in the world (Eberhard and Fennig, 2021), most of which are entirely ignored by NLP practitioners and researchers. Biases in how technology is built and deployed may have unexpected consequences. For instance, in the early 2000s, the lack of any decent keyboard to write in the Arabic script boosted the use of Arabizi (Haghegh, 2021) — the Arabic language written in the Latin script. In this case, the lack of a given technology (a usable keyboard) suited for a specific writing system led to the spread of new ways of writing Arabic. About 3,000 languages are considered endangered (Eberhard and Fennig, 2021) and about 90% of the world languages have little or no online presence (Kornai, 2013). In this context and given the widespread use of NLP technologies online, NLP

systems biased toward specific languages may favor some linguistic communities against others in accessing knowledge and communicating online.

As supported by the UNESCO in its 2003 *Recommendation concerning the Promotion and Use of Multilingualism and Universal Access to Cyberspace*[13] building multilingual systems is of first importance to support linguistic diversity and its underlying communities. In this thesis, we will present training and adaptation techniques that provide usable models for languages that only have small amount of data available and that are left-out by most of the research done in NLP.

For these three reasons, in this thesis, we focus on the following research question: *How can we make language models better at handling the diversity and variability of natural languages?* . To answer it, we explore three main directions.

1. Domain Gap Reduction Approach (Part III),
2. Behavioral and Structural analysis of language models (Part IV),
3. Cross-Lingual Adaptation techniques (Part IV).

First, we focused on a domain gap reduction approach. This approach aims to make the out-of-domain highly variable data more similar to the training data. To achieve this, we designed a lexical normalization model and reached competitive performance on standard benchmarks. However, the data scarcity of this task and domain led us to design more direct adaptation techniques. Second, Transformer-based language models are complex objects. One of the first steps taken for this thesis has been to understand the behavior of these models in various training and evaluation scenarios. That is what we refer to as *behavioral analysis*. This step also led us to create new evaluation datasets. Finally, we present adaptation techniques that directly model the target out-of-domain data and possibly in a language different from the training data (*cross-lingual*).

[13] cf. https://en.unesco.org/recommendation-mulilingualism and https://unesdoc.unesco.org/ark:/48223/pf0000151952

1.3 Contributions

1.3.1 Contributions' Summary

CamemBERT: Building and Analysing a state-the-art Language Model for French (Chapter 5)

The cornerstone of our research is language modeling. In this thesis, we work with BERT-like models (Devlin et al., 2018a). We present our contribution to developing a BERT model for French, CamemBERT, done in collaboration with other members of ALMAnaCH, in particular, Louis Martin and Pedro Ortiz. Before this work, only English and a few other languages benefited from the release of a large-scale monolingual transformers-based language model. CamemBERT was one of the first non-English monolingual transformers language models. With CamemBERT, we extended the state-of-the-art performance on four downstream tasks in French. We then analyzed the key pretraining elements. In contrast with what had been described before this work, we showed that more data is not always necessary and that pretraining on a diverse corpus of Web crawled data of only 4 Gigabytes of text is enough to reach state-of-the-art performance.

Enhancing BERT for Lexical Normalization (Chapter 6)

User Generated Content (UGC) is very challenging for NLP systems. Indeed, it typically includes jargon, grammatical errors, spoken language, emojis, etc. Additionally, there are very few task-specific annotated datasets of UGC data. One approach to address this variability and this scarcity of data is to perform lexical normalization, i.e., to translate the non-standard words into standard ones.

For this purpose, we reframe lexical normalization to make it a token-level classification task. Based on this, we enhance BERT's architecture to fine-tune it on a lexical normalization dataset. Our model did not outperform the former state-of-the-art performance — the MoNoise (van der Goot, 2019) feature-based approach — but managed to compete with it without needing external lexicons and millions of raw tweets.

Understanding the cross-lingual abilities of mBERT (Chapter 7) Cross-lingual transfer consists of using a model trained on a *source* language for another *target* language. Pretrained multilingual language models (Devlin et al., 2018a; Conneau et al., 2020a; Xue et al., 2021) have been shown to reach non-trivial zero-shot cross-lingual transfer performance (Libovický et al., 2019; Pires et al., 2019a). This transfer is remarkable as at no point of the training process (pretraining and fine-tuning) the model receives any training signal to learn shared representations across different languages.

Explaining the performance and behavior of large deep learning models is inherently challenging. Indeed, they are made of hundreds of millions of parameters trained end-to-end on a vast quantity of data (Goodfellow et al., 2016).

To overcome this challenge, in collaboration with Yanai Elazar from Bar-Ilan University, we developed a structural and behavioral analysis of mBERT, a popular multilingual language model. We introduce Random-Init that consists in selectively randomly initializing specific layers to study their impact on downstream performance. Using Random-Init, we show that mBERT is schematically made of two modules. The lower layers are critical for cross-lingual transfer and align representations across different languages. The upper layers are task-specific and do not contribute to cross-lingual transfer.

Adapting Multilingual Language Models on Unseen languages (Chapter 8) There are about 6,500 natural languages in the world. Most of the NLP community's focus is on English and a few high-resource languages (Joshi et al., 2020b). However, in the past five years, a few large-scale language models trained in about 100 languages have been released (Devlin et al., 2018b; Conneau et al., 2020a).

In this work, done in collaboration with Antonis Anastasopoulos from George Mason University, we aim to design techniques to build accurate models for the following 1000 languages. We specifically study how to adapt mBERT to deliver good performance on these languages and compare it to language models trained from scratch and strong non-contextual baselines such as LSTM models.

For this purpose, we start with extensive experiments on North African Arabizi (Narabizi), a non-standard Arabic dialect written in the Latin script with no standard writing

rules characterized by a rich morphology and a high degree of code-mixing with French. We show that mBERT can reach non-trivial performance on this dialect despite being unseen by the model at all the steps of the pretraining. We show that the high degree of code-mixing with French explains this transfer and that performing masked-language modeling adaptation improves the performance of Narabizi.

We then extend this analysis to 17 unseen languages. For most of these languages, performing task-specific fine-tuning and Masked-Language Modeling fine-tuning enable the model to perform better and outperform other models. However, for a subset of these languages, this recipe does not work. We show it only works if the target language is related to a language included in the pretraining corpus with which they share the same script. However, for these languages, we show that solving this script discrepancy with transliteration solves this problem and leads to significant performance progress.

Dataset Creation During this thesis, we also contributed by building datasets. As part of (Seddah et al., 2020)'s work, we collected and annotated data for the Narabizi dialect. I took part in the design of the raw data collection protocol and the evaluation of the dataset. As part of an internship, I took part in the making of a multilingual generative Question Answering dataset for five languages. Finally, for the multilingual clause-level shared task,[14] we built the French section using a syntactic lexicon (Sagot, 2010).

[14] https://sigtyp.github.io/st2022-mrl.html

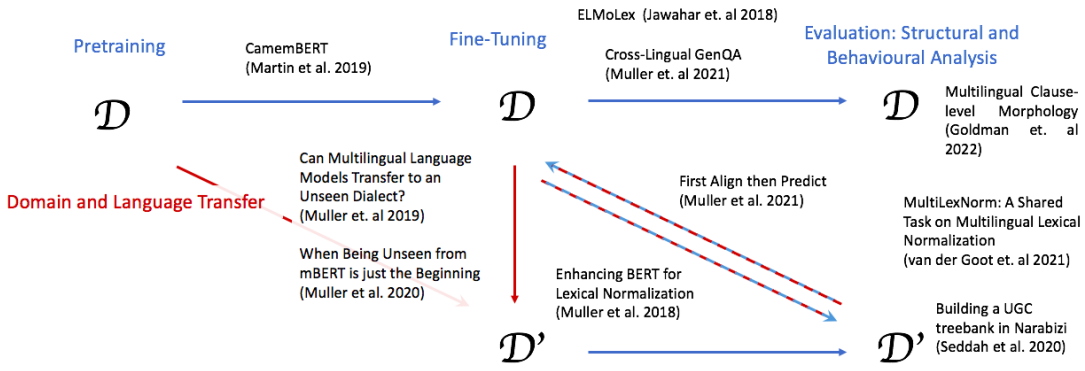


Figure 1.2: A visual summary of my work: D corresponds to one set of domains or languages (e.g., Web Data in French) and D' to another set of domains.

1.3.2 Publications Related to this thesis

We present the publications related to this thesis in Figure 1.2, which aggregates in a diagram the research papers done for this thesis. Schematically, the research presented can be seen along two dimensions. The first dimension (illustrated by the vertical axis in Fig. 1.2) is about the evaluation setting. More specifically, in some of our work, the training domain and language are the same as the evaluation domain and language. In some other contributions, we work in a setting in which the evaluation domain (e.g., referred to as *D'*) is different from the training one (referred to as *D*). For instance, if the training data is Wikipedia data in English while the test data is social media data in dialectal Arabic from Tunisia. Second, our experimental process is typically made of three steps: a pretraining step, a fine-tuning step, and an evaluation step. The second dimension (illustrated by the horizontal axis in Fig. 1.2) corresponds to the stage of the training-evaluation process we focus on.

We further list the publications related to this thesis:

- *Fifth Workshop on Universal Dependencies* - ELMoLex: Connecting ELMo and Lexicon Features for Dependency Parsing - (Jawahar, Muller, Fethi, Martin, Villemonte de la Clergerie, Sagot, and Seddah, 2018),
- *Fifth Workshop on Noisy User-generated Text (W-NUT)* - Enhancing BERT for Lexical Normalization (Muller, Sagot, and Seddah, 2019),
- *58th Annual Meeting of the Association for Computational Linguistics* - CamemBERT: a Tasty French Language Model, (Martin*, Muller*, Ortiz Suárez*, Dupont, Romary, de la Clergerie, Seddah, and Sagot, 2020),
- *The first annual EurNLP Summit* - Can multilingual language models transfer to an unseen dialect? A case study on north african arabizi (Muller, Sagot, and Seddah, 2020b),
- *58th Annual Meeting of the Association for Computational Linguistics* - Building a User-Generated Content North-African Arabizi Treebank: Tackling Hell (Seddah, Essaidi, Fethi, Futeral, Muller, Ortiz Suárez, Sagot, and Srivastava, 2020),
- *12th Conference on Language Resources and Evaluation* - Establishing a New State-of-the-Art for French Named Entity Recognition (Suarez, Dupont, Muller, Romary, and Sagot, 2020)

- *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics* - First Align, then Predict: Understanding the Cross-Lingual Ability of Multilingual BERT (Muller, Elazar, Sagot, and Seddah, 2021b),
- *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* - When Being Unseen from mBERT is just the Beginning: Handling New Languages With Multilingual Language Models (Muller, Anastasopoulos, Sagot, and Seddah, 2021a)
- *Proceedings of the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics* - Cross-Lingual Open-Domain Question Answering with Answer Sentence Generation (Muller, Soldaini, Koncel-Kedziorski, Lind, and Moschitti, 2021c)
- *Seventh Workshop on Noisy User-generated Text (W-NUT)* - MultiLexNorm: A Shared Task on Multilingual Lexical Normalization (van der Goot, Ramponi, Zubiaga, Plank, Muller, San Vicente Roncal, Ljubešić, Çetinoğlu, Mahendra, Çolakoğlu, Baldwin, Caselli, and Sidorenko, 2021)

1.4 Societal Risks of Building NLP Models

Research, engineering, and deployment of NLP systems are currently receiving a lot of interest and are used intensively on a global scale. It is therefore highly needed to think and aim at preventing the potential risks that NLP technologies pose to society. We list here three critical risks.

Environmental Cost Training and deploying large-scale NLP systems consumes electricity. A substantial portion of these systems is trained and deployed using public cloud providers[15] such as Amazon AWS, Microsoft Azure, Google Cloud, and Alibaba Cloud.[16] In academia, training NLP systems is usually done based on university or governmental clusters. Most of the experiments done during this PhD curriculum were

[15] https://info.flexera.com/CM-REPORT-State-of-the-Cloud

[16] These four providers made nearly 75% of the public cloud market in market share according to https://holori.com/cloud-market-2022/

done on the NEF cluster[17] and the Jean-Zay [18] cluster. While having detailed statistics on the overall electricity consumption of NLP systems is tricky, we can provide upper bounds. The total global data center industry consumption was around 200-250 TWh in 2020 (IEA, 2022)[19] which amounts to about 1% of the World electricity consumption.[20] Furthermore, only a small portion of that is dedicated to training and deploying NLP models. For instance, Google, a major user of machine learning, reported that only 15% of its electricity consumption comes from Machine Learning models (Patterson et al., 2022). The global electricity production produces about 40% of the global CO2 emissions.[21][22] However, these emissions vary a lot based on the power grid used. For instance, France derives about 70% of its electricity (which partly powered the clusters used for this thesis) from nuclear energy[23] which has a low carbon footprint. In comparison, about 60% of electricity in the US comes from coal and natural gaz[24] which have a very high carbon footprint.

CO2 is an important element of environmental cost but not the only one. Industrial processes involved in hardware production (such as wires, hard drives, RAM, CPUs, GPUs, and TPUs) significantly impact the environment. Hardware is made of silica and a large variety of rare earth elements (REE), like hafnium for CPUs and palladium and tantalum for GPUs.[25] Mining these elements, as done today in the vast majority of mines, is the source of many negative externalities like water and soil pollution.[26] Additionally, mine workers face dangerous working conditions that can potentially have a disastrous effect on their health.[27]

Cultural, Gender and Demographic Biases in NLP technologies NLP technologies are inherently biased due to how they are designed, trained, and evaluated

[17] https://wiki.inria.fr/ClustersSophia/Clusters_Home
[18] http://www.idris.fr/eng/jean-zay/jean-zay-presentation-eng.html
[19] https://www.iea.org/reports/data-centres-and-data-transmission-networks
[20] https://www.statista.com/statistics/280704/world-power-consumption/
[21] https://www.iea.org/articles/greenhouse-gas-emissions-from-energy-data-explorer
[22] The total global CO2 emission in 2018 amounts to 48.9 GtCO2e (Giga Tones of CO2 emissions)
[23] According to www.statista.com
[24] According to https://www.eia.gov/energyexplained/electricity/electricity-in-the-us.php
[25] https://euromines.org/files/key_value_chain_electronics_euromines_final.pdf
[26] https://e360.yale.edu/features/china-wrestles-with-the-toxic-aftermath-of-rare-earth-mining
[27] https://www.carexcanada.ca/Timis_Mining_REE.pdf

(Crawford, 2017; Suresh and Guttag, 2019; Blodgett et al., 2020). These biases concern all the dimensions of the identity of an individual. For instance, it can involve gender, sexual orientation, ethnicity, race, or culture. These biases may impact how specific individuals are represented regarding other individuals or communities (called representational biases). It can also affect what resources or opportunities NLP systems recommend to a specific individual or community (called allocation bias) (Savoldi et al., 2021).

Harmful Content Generation Finally, with the emergence of powerful generative NLP systems (Radford et al., 2019; Brown et al., 2020; Chowdhery et al., 2022), the generation of harmful content has become a very substantial risk (Bender et al., 2021). These models can generate toxic, offensive, adult, racist, and homophobic content (Gehman et al., 2020). Using such models and deploying them in a safe environment is therefore critical.

The research presented in this thesis contributes to mitigating some of the costs and risks listed above. More specifically, Part IV details our contribution to adapting multilingual language models for low-resource languages. With these contributions, we provide actionable solutions that can help build models that are less costly to train and less centered on high-resource languages.

Part I

Background

2 The Variability and Diversity of Natural Language(s)

In language lie our abilities to express our feelings and thoughts, to develop complex reasoning, to signal our position in a social group, and to become a community. In short, language is quintessential to our human nature.

Aiming to develop techniques that can automatically process human language(s), we begin by describing human languages themselves to get a glimpse of the challenge we face.

Our first challenge is conceptual. Indeed, building NLP systems requires defining what target use case we focus on and, more specifically, what languages and domains we are dealing with.

In this chapter, we define what we mean by the terms *language*, *languages*, *dialects* and *domains*. Even though most of these terms are used in our day-to-day life as if they were characterizing well-defined objects, they are in fact, way more complex than they seem. In short, in this chapter we will see that:

- Language is an arbitrary system of symbols through which humans communicate, (partially) think, and build relationships. In consequence, there is no such thing as the concept of *language* independent of the human beings that use it (Sapir, 1968),
- Language varieties are not well-defined linguistic objects. There are conventionally defined based on a historical, sociological, and political context. Language varieties are very diverse, specifically their linguistic properties such as phonological, morphological, and syntactical properties (Dryer and Haspelmath, 2013),

- Speaking and writing are social phenomena that can vary greatly depending on what group we are speaking/writing to and what means we are using to speak or write (Trudgill, 2000; Wardhaugh and Fuller, 2014; Heller et al., 2016),
- In this thesis, we study language(s) through a corpus of textual data that are defined as collections of documents and tokens.

After defining these concepts, we will illustrate the diversity and variability of human languages statistically.

2.1 What is *Language*?

There is nothing more familiar to humans than language. We use language to communicate our emotions and ideas with our peers, grasp the world around us, and think about ourselves and others. Still, it is challenging to define formally what human language is. We start by describing a few essential functions and characteristics intrinsic to human language.

2.1.1 The Functions of Human Language

Language as a Mean of Communication One of the first things that come to mind when we define what is *language* is that it is a means of communication. In every human community, we are taught to speak in a certain way to communicate with our peers. By the age of three, most children are experts at using language (Mehler and Dupoux, 2002). For a large proportion of humans, we are also taught to write and read.[1] Using speech and writing,[2] we can share information about the world with other people, and we can express ideas and emotions. This relatively intuitive statement echoes Sapir (1968) who states that *Language is a purely human and non-instinctive method of communicating ideas, emotions, and desires*. Consequently, language is inherently rooted in what makes us human and, more specifically, *social beings*. Furthermore, we note that Sapir (1968) characterizes *language* as a *method*. How can we define this method, and what key properties are intrinsically related to it?

[1] According to the World Bank, the world population reached a literacy rate of 86.6% in 2020 (cf. https://data.worldbank.org/indicator/SE.ADT.LITR.ZS)

[2] Leaving aside other modalities such as sign language.

Language and Thought We express ourselves using language. To a certain extent, we also "talk to ourselves" and think using language. However, the precise impact that human language and the language(s) we speak has on our thinking process is still an ongoing research topic (Gleitman and Papafragou, 2005) that goes beyond this thesis's focus.

Language as a social behavior: Phatic function Beyond concrete communication, language plays a key role in our relationships with our peers. Indeed, language is often used without other purposes than creating or maintaining contact with someone else. This is what is referred to as the *phatic* function of language by (Jakobson and Halle, 1956; Yaguello, 1981; Jumanto, 2014).

Language as an expression of identity We covered two critical characteristics of human language. These two characteristics imply something fundamental about language: it is intrinsically related to the individual that uses it and the group of humans in which its use occurs. In consequence, not only are languages the primary means humans use to communicate and create social relationships with others, but they are also a way people identify each other as individuals and as groups (Bucholtz and Hall, 2004).

Based on the language we speak and how we speak and write (e.g., our accents or the words we use), people can guess information about our identity (Giles and Coupland, 1991; Watt, 2009), where we grew up, our social class, our ethnicity, etc. In short, language can be perceived as proxy for many socio-demographic variables. A proxy so precise that selecting proper socio-demographic subsegment of a given corpus can enhance NLP system performance (Hovy and Søgaard, 2015). In addition, at a collective level, how we speak enables a group to bond and exist (Trudgill, 2000; Wardhaugh and Fuller, 2014; Heller et al., 2016).

2.1.2 Language as an Arbiratry System of Symbols

A key notion to grasp the nature of language is that it is made of *symbols*. By symbols, we mean an abstract representation of "something".

Saussure (1916), in his *Cours de Linguistique Générale*, gave to this notion a formal framework grounded in psychology. He uses the term of *linguistic sign* that he defines as a two-sided psychological entity that unites a *sound-image* (or *signifier*) and a *concept* (or *signified*). He defines the former as the *sensory imprint* that the sound or image of a linguistic sign has on someone. We note that sound-image not only refers to speech but also to the mental imprint that occurs when we write, read or simply talk to ourselves. Based on this definition, we can easily derive that signs are rooted in each individual experience. For instance, two people speaking different languages will use different sound-image for the same concept. For instance, a French-speaking person will refer to the concept of "tree" as *arbre* while an Italian speaker will refer to it as *albero*.

In other words, the relationship between concept and sound-image is arbitrary. Two distinct sound-images may refer to the same concept for two people (e.g. *arbre* vs. *albero*), while a single sound-image may characterize two distinct concepts for two people.

2.1.3 What is A Language?

It is "common knowledge" to talk about languages as independent, well-defined objects. In the NLP research community, it has become common even to enumerate the number of languages that exist in the world. For instance, the *Ethnologue* publication of 2021 (Eberhard and Fennig, 2021) set the number of languages in the world to 7,459. Still, defining what *a language* is conceptually challenging.

The notion of languages is based on the idea of *mutual intelligibility*. If person A can speak and be understood by person B and person B can speak and be understood by person A, this means that "they are speaking the same language". Intuitively, based on "mutual intelligibly," it seems almost trivial to define groups of speakers that speak "the same language". Based on this group, we could then easily derive the concept of *languages* and enumerate the different languages that exist in the world. Still, this trivial reasoning is challenged by the fact that mutual intelligibility is not a transitive property. In a nutshell, this means that if person A can speak and be understood by person B and person B can speak and be understood by person C, person A may not necessarily be able to speak and be understood by person C. This lack of transitivity is observed in practice in many parts of the world. For instance, in Europe, across Switzerland, Germany, and the Netherlands,

people are likely to understand each other if they meet someone living not too far from their home, regardless of regional or national borders. Still, a Swiss may not be able to speak to a person several hundred kilometers from their home in the Netherlands. Such a phenomenon is called a *Dialect Continuum*. It is observed in Europe with German Varieties (Gooskens et al., 2011), in China with Mandarin varieties (Norman et al., 1988), or with Arabic in the Arabic peninsula and north Africa (Versteegh, 1997; Čéplö et al., 2016).

Based on these well-established empirical observations, defining objectively what a language is, grounded solely on mutual intelligibility is impossible. In this regard, the notion of *languages* as it is used in the day-to-day life is a conventional term that depends on the historical, sociological, and political context that defines some specific language varieties. In some other contexts, some language varieties may be referred to as *dialects*.

In the rest of this thesis, as we will detail in §2.5, we will take a strictly data-driven approach to the definition of what constitutes a language and base our experiments on corpora that may be identified as originating from a specific language. We will collect textual data from "a given language" (e.g., English, French or Maltese) and estimate statistics and models using this data.

2.2 Linguistic Analysis

From a structural linguistic point of view, it is usual to categorize different levels of linguistic analysis of natural language. Each level focuses on a specific aspect of natural language by first defining a specific unit of interest.

The notions introduced here will be helpful to characterize more precisely the models we will develop in the following sections of this thesis.

2.2.1 Linguistic Units

The notion of *word* is used very commonly. In this chapter, we use the term *word* following its "common usage". Defining rigorously what is a word and doing it in a way that is consistent across languages is challenging. A simplistic way to do it is to use some language-specific typographic rules to define it. For instance, we can define words based

on blank spaces for most languages that use the Latin script or the Cyrillic script. For these languages, a word is any sequence of characters, excluded from special characters such as punctuation symbols, between two blank spaces. For languages that use logographic script, e.g. Mandarin, (§ 2.3.2), any Chinese character could be simplistically considered as words. In the rest of this thesis, we will favor the term of *token*, whose definition and scope will vary according to the algorithm that produced them (cf. §4.2.2).

2.2.2 The Seven Levels of Linguistics of Analysis

Phonetics

The first level of linguistics analysis is the study of sound patterns. Phonetics focuses on studying speech sounds, also referred to as *phones*. More specifically, it studies how humans produce and perceive these sounds. We can classify phones based on what *speech organs* among the lips, teeth, tongue, palate, uvula, nasal and oral cavities, and vocal cords, is involved in the phone production. That is how the International Phonetic Alphabets (IPA) was defined according to (Jespersen, 2013). We can split phones into vowels and consonants. Phonetically, we define a vowel as any sound with no audible noise produced by constriction in the vocal tract and a consonant as a sound with audible noise produced by a constriction (Loos et al., 2003). A *syllable* is a unit of sound composed of a peak of sonority, usually a vowel, and the consonants that cluster around this central peak (Loos et al., 2003).

Phonology

Phonology describes how speech sounds encode meaning in a given language. The basic unit of phonology is the *phoneme*. A phoneme is the shortest speech sound that, if swapped for another phoneme, can modify the meaning of a word. We note that the concept of phonemes is, therefore, specific to a given language. For instance, the word "cat" has three phonemes: /c/ /a/ and /t/. Indeed, if we swap /c/ for /h/ we easily observe that it changes the meaning of the word "cat" to "hat".

Graphemics

Graphemics is the study of the writing system. It describes how, for a given language and a given sociocultural context, phonemes are transposed into writing with a specific writing system. The basic unit of study is the *grapheme*, also referred to as a character, defined as the smallest functional unit of a given writing system. For instance, in the Latin script, the letter *a* is considered a grapheme.

Morphology & Syntax

Morphology and Syntax study what is referred to as *wordforms*.[3] Morphology studies how smaller basic units called *morph* are combined to form wordforms. Syntax studies how wordforms are combined to make larger units of interest, such as sentences. The conceptual challenge lies in the fact that there is no universal[4] definition of morphs, wordforms, and sentences. Indeed, any definition suits or favors a specific language or language family. For instance, a definition of the wordform in Mandarin is likely to differ a lot from how we would define it for French. Still, finding a consistent definition across languages is necessary to analyze the typological similarities and differences across languages. In this thesis, we follow the Universal Dependency (UD) framework described by de Marneffe et al. (2021). UD is a cross-lingually consistent morphological and syntactic annotation framework that defines wordforms. We present in chapter 3, two morpho-syntactic tasks that we will study in this thesis, namely POS tagging and dependency parsing, that consists in predicting the morpho-syntactic categories and relations between wordforms.

Semantics

Broadly speaking, semantics is the study of the meaning of words and phrases (Cruse et al., 1986). Linguists have considered many angles to define and approach the concept of meaning. For instance, generative semantics states that syntactic structures are computed based on meaning. In contrast, interpretive semantics claims that the meaning of words and phrases emerges from syntactic structures and linguistics context (Chierchia and Mcconnell-Ginet, 1990). Beyond these theoretical concerns, semantics encompasses a

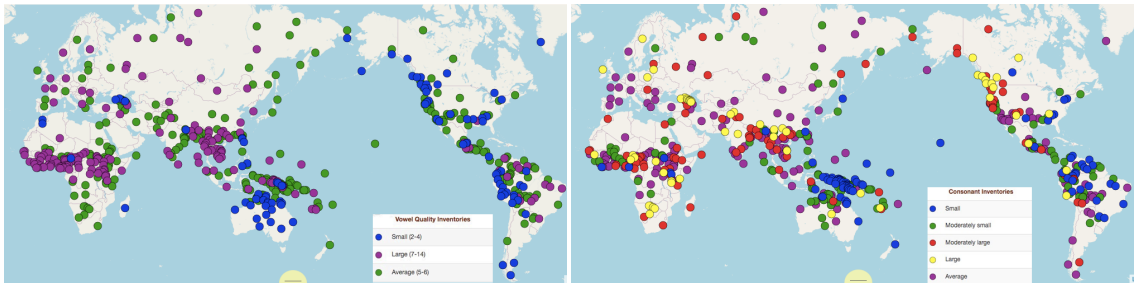
[3] Also commonly referred to as *syntactic word* or *grammatical word*
[4] In the sense, language-agnostic

large number of fields, such as the study of *semantic shift* (Stern, 1975) (which consists in studying the evolution of word meaning), and ontological semantics (Nirenburg and Raskin, 2000) which consists in encoding semantic descriptions and relations between words in a structured way. This was famously implemented for English at scale with the Wordnet (Miller, 1995).

Pragmatics

Pragmatics is the study of how the non-linguistic context interacts with the meaning of words. For instance, it studies how social context, personal relationship, knowledge of the world, and common sense of the speakers, impact the emergence of meaning in language (Mey, 1993).

2.3 A Large Typological Diversity[5]



(a) Vowel Inventory (b) Consonant Inventory

Figure 2.1: Geographical distribution of Vowels and Consonants Inventory Sizes (Maddieson, 2013)

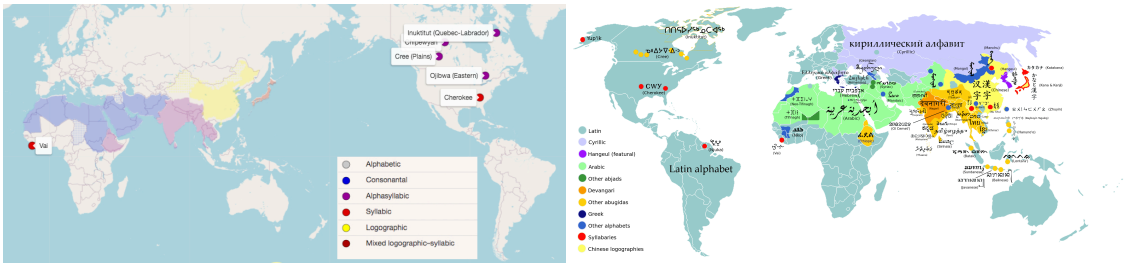
We illustrate the large diversity of languages across the world by observing several fundamental linguistic properties and their distribution across human languages. To

[5]Section 2.3 is inspired by Benoît Sagot NLP course's material https://github.com/edupoux/MVA_2022_SL

do so, we use the World Atlas of languages (WALS) (Maddieson, 2013). The WALS is a large collection of phonological, morphological, syntactical, and semantical properties collected by linguists on about a thousand languages and dialects.

2.3.1 Phonological Diversity

We use the WALS to observe the distribution of consonants inventory and vowels inventory across languages. These statistics are collected on around 500 languages. For vowels, the smallest vowel inventory recorded includes only two elements while the largest 14 (the German language is the only recorded language that uses 14 vowels). We illustrate in Figure 2.1 the geographical distribution of vowel inventories. For instance, we observe many large vowel inventories (between 7 and 14 vowels) in Africa in the Sub-Saharian region. This cluster includes languages that belong to the Niger-Congo, Nilo-Saharan, and Afro-Asiatic families. For consonants inventory, the number of consonants per language recorded varies from 6 to 34. We observe a high concentration of small consonants inventory in New Guinea and the Amazonian basin.



(a) Writing System Type (Maddieson, 2013) (b) Writing System (cf. `wikipedia`)

Figure 2.2: Geographical distribution of Writing systems

2.3.2 Diversity of Writing Systems

Not all language varieties have a writing form. According to Eberhard and Fennig (2021), about 40% of them are not frequently used to write. Within the set of language varieties typically written, we observe a large diversity of writing systems. In a nutshell, there are about a dozen writing systems in the world.

Having the same script does not imply the sharing of common linguistic properties. For instance, Turkish and German uses the Latin script. Still, these two languages are very different syntactically and morphologically. On the other side of the spectrum, Serbian and Croatian are structurally similar and could even be considered the same language. Still, Croatian is exclusively written in the Latin script, while Serbian is written in both the Latin script and the Cyrillic script.

Building a typology of writing systems is challenging. Indeed, writing systems are complex objects, and any typology will overvalue or under value one property compared to another. Simplistically, we can divide writing systems into five categories: logographic, syllabic, alphabetic, consonantal and alphasyllabic (Comrie, 2013).

Logographic

Logographic writing systems use logograms as their basic units. A logogram is a character that represents a word or a morph. The Chinese writing system is the most broadly used logographic writing system. It is based on Chinese Characters or *Hanzi* (汉字 in simplified Chinese). Chinese characters, originated in mainland China, have been partially integrated into many other writing systems, such as the Korean writing system (*Hangul*) and the modern Japanese writing system.

Alphabetic

An Alphabetic writing system uses unique symbols (a character) for vowels and consonants. The most prevalent Alphabetic scripts are the Latin and Cyrillic scripts. Depending on the language varieties used, the number of characters in the Latin script varies from around 26 letters (excluding diacritics). In comparison, it is approximately 32 letters in the Cyrillic script.

Consonantal

A consonantal writing system is a variant of the alphabetic writing system for which only consonants are represented. For example, the Hebrew or Arabic writing systems are consonantal. In many consonantal writing systems, such as Arabic, vowels can be represented with *diacritics* (i.e. a mark added above or underneath a character).

Sylabic

Syllabic scripts, also referred to as syllabaries, are scripts for which a grapheme encodes an entire syllable. The Japanese hiragana grapheme comes close to this definition (Comrie, 2013).

Alphasyllabic

Finally, the Alphasyllabic scripts or alphasylabaries are very close to consonantal scripts — i.e., only consonants have graphemes, and diacritics are used for vowels — the only difference being that diacritics must be written. Thai is, for instance, an Alphasyllabic script.

Directionality

The direction used to write characters sequentially also varies across writing systems. Right-to-left, top-to-bottom is the direction used for most Alphabetic writing systems such as Latin, Greek, and Cyrillic. This contrasts with Arabic and Hebrew, which are written left-to-write top-to-bottom. Many scripts, mostly in Asia, use a top-to-bottom right-to-left direction, such as Chinese, Japanese and Korean. Still, we note that they can also be used in the right-to-left top-to-bottom direction. Finally, a small minority of scripts use a bottom-to-top approach. The Hanunoo script is used to write the Hanunoo language, an indigenous language of the Philippine written from bottom-to-top and left-to-right (Daniels and Bright, 2010).

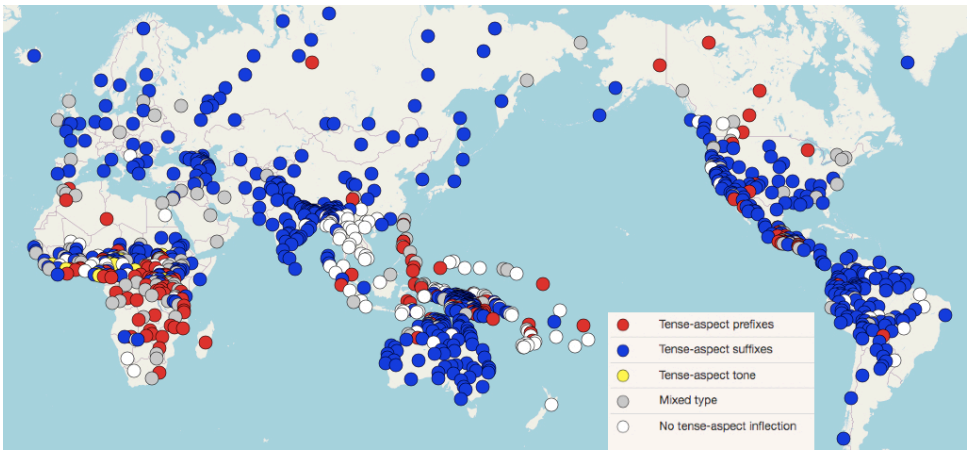


Figure 2.3: (Dryer, 2013b) Tense-Aspect Affixation of the world languages (on a sample of 969 languages).

2.3.3 Morphological Diversity

The way morphs are combined into wordforms varies greatly. For instance, the way wordforms are inflected or reinflected to encode different grammatical role (Lieber, 2009) varies across languages. Vowel gradation (as observed in *sing* → *sang*) is used in multiple indo-european languages for verb inflection (Eska and Szemerenyi, 2000). Reduplication which consists in duplicating a morph (Rubino et al., 2002) is observed in languages such as Thai (Iwasaki and Ingkaphirom, 2005) or Yoruba (e.g. *gbọ́mọ gbọ́mọ gbọ́mọ* (carry child carry child carry child) → *gbọ́mọgbọ́m* (kidnapper) (Arokoyo, 2006). [6] One inflection process used across a large number of languages is *affixation*. Affixation occurs when a morph is attached as a prefix (i.e., at the beginning) or as a suffix (i.e. at the end) of a word to derive another word. Affixation is used for many functions. For some languages, it is used as a plural mark (e.g., a dog, two dog**s**), a possessive mark, or even an interrogative mark. In Figure 2.3, we show Tense-aspect affixations and the distributions across the world languages. Some languages like Romance languages use suffixes to mark tenses (e.g.

[6] https://bolanlearokoyo.com/2020/06/22/reduplication/

(io) *prend**o*** (I take) → (tu) *prend**e*** (you take) in Italian). By contrast, many languages from the Bantu language family, like Swahili or Zulu, use prefixes to mark verb tenses.

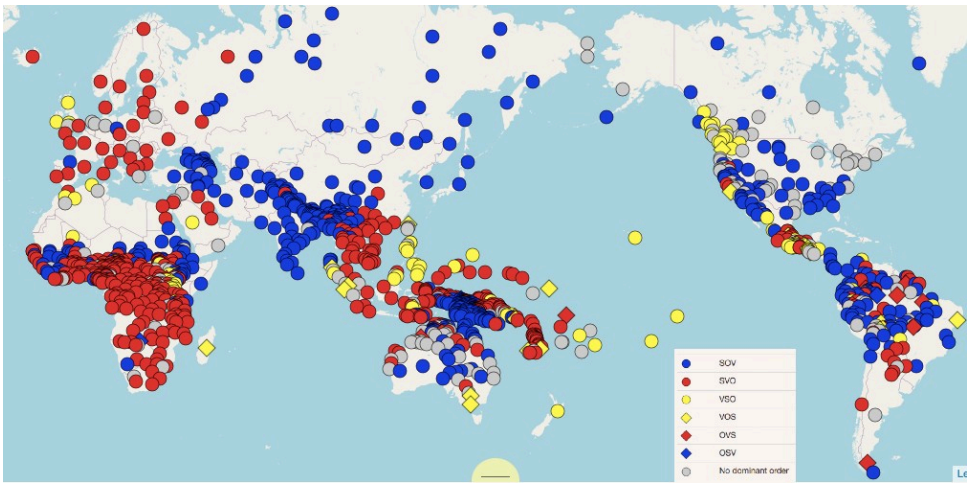


Figure 2.4: Geographical distribution of {Subject (S), Object (O), Verb (V)} word order (Dryer, 2013a)

2.3.4 Syntactic Diversity

Third, we take the {Subject (S),Object (O),Verb (V)} order to illustrate the large diversity in syntactic structures. As seen in Figure 2.4, a majority of languages have a *SVO* word order (the Subject followed by the Verb followed by the Object). The structure SOV predominates in most parts of Asia (if we exclude the Middle East and South-East Asia). Finally, we note that some languages, such as German, do not have a dominant {Subject (S), Object (O), Verb (V)} word order. Indeed, for German, this order is syntactically determined by the presence of an auxiliary verb.

In summary, human languages are very diverse at every level of analysis. As we will see now, language is highly variable even within a given dialect or language.

2.4 Variability of Natural Languages

Even between speakers or writers of the same language or dialect, there is a high level of variability. At the speech level, this variability may come from how one pronounces certain words, how they stress certain syllables, and what intonation one uses. It may come from how one will pick specific words, how one will build sentences, and with what level of formality one will express their thoughts (Trudgill, 2000).

Additionally, even if we look at the language production of a single individual, speaking and writing also vary a lot. At work, someone speaking with their boss is likely to speak differently than in a restaurant with friends. Through email, the formality someone uses with a colleague is likely higher than if one comes across them at the cafeteria. Moreover, the topic of the conversation or the writings also impacts how language is used. Having a scientific conversation will be based on different words and sentence constructions than when talking about the weather. The impact of the non-linguistic socio-cultural context on language production has been described extensively by Trudgill (2000).

Schematically, language varies based on:

- *Who* is speaking/writing?
- *To Whom ?*
- *About what?* e.g. the weather, a scientific paper, politics.
- *In what context?* Is this happening at work, in a bar, or a conference?
- *With what medium?* e.g., speaking, writing in an email, texting.
- *In What Year?* in 2022, in 2002, in the 1950s.

Ideally, we would like to build NLP systems by integrating and modeling each of these factors. However, it is often not possible to do so. For this reason, we must define broader categories of language production along which language varies.

2.4.1 External Determiners of Language Variability

We now define some broad determiners of language variability.

Medium We produce language differently, whether speaking or writing. As described extensively by Chafe and Tannen (1987), among other phenomenons, the Vocabulary

used, sentence length, and sentence structure usually differ significantly between speaking and writing productions.

Technology The technology medium used to produce written text also impacts how we write. The various digital mediums and platforms that now exist to communicate and produce language also highly impact how we write. We do not write in the same way on *iMessage*, *WhatsApp* and *Messenger* mainly because each application enables different ways of communicating (for instance, emojis will be recommended differently, or autocomplete systems will suggest different words).

Technology's impact on how language is produced is, in some cases, even stronger than that. Digital technologies such as online chats, short message services (SMS), and mobile phones were all developed primarily for English speakers and writers. Consequently, speakers of non-Latin script languages had to find ways to express themselves using Latin-script keyboards. This has been described by Henry and Pramoolsook (2014) as one of the root causes of the widespread use of the Latin-script to write Arabic. This phenomenon is also prevalent for many other non-Latin scripts, such as South-Indian languages written in Devanagari, Bengali, or Tamil scripts (Roark et al., 2020). Technologies also impact the content of what people write. Pierozak (2003) described how *ergographic* phenomena, such as the simplification of the text produced (e.g., character deletion, use of phonetics, etc.), emerged from SMS texting in French.

Time All languages evolve with time. They evolve by being in contact with other languages geographically close (e.g., *selfie* was initially introduced in English and has been adopted by many languages worldwide). They evolve by integrating new concepts and words (e.g., *COVID* was a hardly known term in December 2019). As described extensively in (Campbell, 2013), in linguistics, this phenomenon is called *diachronic variability* of languages.

Demographic, Sociological, and Cultural Context Language varies based on many complex social factors (Hovy and Yang, 2021). Social class (Guy, 2011), origins and ethnicity (Fought, 2011), age (Eckert, 2017), gender (Wodak and Benke, 2017),

professional context (McGroarty, 2012) are all social factors that impact what words one uses, the way one construct sentences, the type of discourse one produces.

Topics The topic of language production, whether it is an utterance or a written sentence, simply refers to *what is being talked about.* (Soon et al., 2001; Bender, 2013). On a large scale, language production, such as discourse, conversation, and scientific or encyclopedic articles, may include one or several topics.

Genre While topic characterizes short language production such as sentences or paragraphs, the notions of *genre* and *register*, characterize larger collection of textual data. These terms have been used in various ways in linguistics and literary studies to categorize language production. In *corpus linguistics*, pioneered by Kučera and Francis (1967), *genres* are usually defined to categorize a collection of text based on their situational characteristics. These characteristics can be the audience, the purpose of the language production, or the activity type in which it is being produced (e.g., the context a text is being delivered: a TV show, in a chapter of a novel...) (Biber et al., 1998; Lee, 2001). Consequently, genres are conventional and related to the cultural behaviors of a group of people. A well-known Genre and Sub-Genre Categorization can be found in the British National Corpus (BNC)[7] which is a collection of 100 million words of written and transcripted spoken text in British English of the late 20st century. It includes around 70 genres such as *Academic Writing, Newspaper, Fiction, Conversation...*[8].

2.4.2 Linguistic Characteristics of Language Variability

To describe how these external determiners impact the language being produced, we characterize the different linguistic dimensions along which language varies.

Vocabulary The vocabulary (i.e. the set of words chosen by the writer or speaker) (Manning and Schütze, 2002; Brysbaert et al., 2016) used in a language production varies a lot with regard to the genre (Biber et al., 1998), socio-demographic determiners[9]

[7] http://www.natcorp.ox.ac.uk/cpr.xml?ID=reference

[8] BNC distribution across Genre at https://varieng.helsinki.fi/series/volumes/19/lijffijt_nevalainen

[9] https://langcog.github.io/wordbank-book/demographics.html

(Fenson et al., 1994; Feldman et al., 2000; Maguire et al., 2018) of the author and all the other determiners we have listed above. A set of words used frequently in a given context (e.g., News in 2021) might be nonexistent or infrequent in another context (e.g., British literature of the 19th century).

Sentence Type The type of sentences and their frequencies varies from one context to another. From a functional standpoint, sentences are usually divided into four broad categories: declarative sentences that make an assertion, interrogative sentences to ask questions, imperative sentences to make a command, and exclamative to express an exclamation (Halliday et al., 2014).

Code-Switching In some contexts, language production might be based on lexicon or morpho-syntactic rules originating from distinct languages (Woolford, 1983). Switching between different languages may occur at any level of language production. Even a single phrase may include a lexicon originating from multiple languages. This phenomenon, called code-switching, usually emerges from multilingual speakers or writers. It is a widespread and natural phenomenon that impacts nearly all languages worldwide. For instance, Spanish and English exhibit a high degree of code-mixing in specific Hispanic communities in the US (Lipski, 2014). In Singapore, Chinese (mainly Mandarin) and English code-mix in a dialect commonly referred to as Singlish (Lee, 2003). As part of the research led during this thesis, we studied in § 8.4 specifically code-mixing occurring in the Arabic Dialects spoken in Tunisia and Algeria that have a high degree of code-mixing with French (Sayahi, 2011).

Style In linguistics, the way we speak or write is referred to as the *style*. Style encompasses a large number of phenomenons such as formality (Heylighen et al., 1999; Pavlick and Tetreault, 2016), the complexity of the language one uses (Sweet, 1899; Yasseri et al., 2012), the type of literary expressions used (e.g. metaphorical), the sentiments and emotions (Troiano et al., 2021).

2.5 Experimental Framework

We aim to build models that can accurately process several languages and are robust to the variability of a given language. We define our modeling framework as follows. In all our experiments, we assume we have a *corpus*. A corpus has the following structure:

- A *corpus* is defined as a collection of *documents* $\{D_1, .., D_C\}$.
- A *document* is defined as a sequence of *tokens* $(t_1, .., t_D)$. A document can be made of a few tokens, a sentence, a paragraph, or a collection of paragraphs.
- A *token* is a basic unit of discrete data. A token can be defined as a wordform (following a given morpho-syntax framework § 2.2), a punctuation mark, a sequence of characters, or even a character itself. The set of all possible tokens constitutes the *vocabulary* $\{t_1, ..., t_V\}$. This means that each token can be defined with a unique index in $[|1, V|]$ We do not assume anything about the language or the script that each token originates from.

At this stage, we do not assume any ordering structure at the corpus level. The documents could be ordered sequentially (e.g., if we take each paragraph of a given book as a document) or not. Additionally, at this stage, we do not make any assumptions about the origin of the corpus. It could be as heterogeneous and diverse as the entire textual data found on the internet or much more constrained, such as the scientific articles from Nature in 2019.

Examples

Here are some corpora we will be using in the rest of this thesis.

- The Open Super-large Crawled Aggregated corpus (OSCAR) (Ortiz Suárez et al., 2019) is a large multilingual corpus that comes from filtering the CommonCrawl snapshot. In OSCAR, each document is a paragraph found online.
- The French Treebank (Abeillé et al., 2000) is a collection of sentences in French annotated with syntactic dependencies. We consider the raw sentences of the French Treebank as a corpus for which documents are sentences.
- The TydiQA (Clark et al., 2020a) dataset is a collection of question passages and answers. We can consider questions and passages as a corpus of textual data.

Domain and Language Definition

Domain is an extensively used term in the NLP literature. However, as noted by Plank (2016), there is *no common-ground on what constitutes a domain*. In this thesis, following our modeling framework defined above, we use the notion of *domain* in a strict data-driven manner. In our experiments, we will typically use a single corpus to train and evaluate a model (e.g., the French Treebank (Abeillé et al., 2000)). This corpus defines our domain, and we will refer to this experimental setting as an *in-domain* setting. In some cases (e.g., §7.4), we will evaluate our models on a corpus different from our training corpus and originating from a different source. Two domains are at play in such a case, and we will refer to this setting as an *out-of-domain* setting.

We note that our definition of a domain may overlap in some cases with the notions introduced earlier. For instance, some domains may correspond to a specific *genre* (e.g., News articles, Literature) or may be characteristic of a particular socio-demographic context that impact the lexicon, sentence types, and topics.

Language Similarly, we will use the term *language* in a strict data-driven manner. We will consider that we experiment in "a given language" based on the corpora we use, which typically identifies the set of languages it includes depending on human judgment (e.g., with the TyDiQA dataset (Clark et al., 2020a)) or automatic language detection (e.g., with OSCAR (Ortiz Suárez et al., 2019)).

2.5.1 Illustrating the Cross-Domain Variability in English

To illustrate language variability described in section 2.4, we take 3 large collection of textual data.

- Wikipedia. We use the Wikipedia English dumps from May 2020 and sample 1% of it twice (noted Wiki0 and Wiki1). Wikipedia is now widely used in the pretraining of transformers-based models,
- BookCorpus (Books) is a collection of fictional books such as Harry Potter or Fight-Club. (Zhu et al., 2015). It was a popular corpus of novels based on which the BERT model was originally pretrained (Devlin et al., 2018a).

- OpenSubtitles (SUBTITLES) is a collection of movie and TV subtitles (Lison and Tiedemann, 2016). The Opensubtitles corpus provides an example of speech data transcripted to text.

We download these datasets using the datasets library (Lhoest et al., 2021) and sample 1 million sentences from each one.

We analyze the lexical divergence between these datasets using two standard metrics. Given two datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ we define the Out-of-Vocabulary Rate of $\mathcal{D}_1$ with regard to $\mathcal{D}_2$, noted $OOV_{\mathcal{D}_2||\mathcal{D}_1}$ as the number of words in $\mathcal{D}_1$ that are not observed in $\mathcal{D}_2$ in proportion to the number of words in $\mathcal{D}_1$:

$$OOV_{\mathcal{D}_2||\mathcal{D}_1} = \frac{\#\{ w \in V_1 \setminus V_2\}}{\#\{ w \in V_1\}}$$

We also compare the distribution of unigram and bi-gram at the word level. To do so we use the Jensen Shanon Divergence (JSD)[10]. The JSD is a symmetric version of the Kullback-Leibler (KL) Divergence.[11] We compute it between the unigram and bi-gram distributions of each pairs of datasets. We first compute the shared set of words and the pair of words observed in each dataset. Then we compute the distribution:

$$P_{uni1} = (\frac{\#w_i}{N}, w_i \in V_1) \quad P_{uni2} = (\frac{\#w_i}{N}, w_i \in V_2)$$

$OOV_{D_2\|\|D_1}$ $D2$: $D1$	WIKI0	WIKI1	BOOKS	SUBTITLES
WIKI0	x	60.58	88.45	83.93
WIKI1	61.20	x	88.55	84.07
BOOKS	51.94	51.61	x	66.19
SUBTITLES	61.09	60.80	80.32	x

Table 2.1: OOV rates i.e. proportion of D_1 (row) that is not in $D2$ (column).

[10] $JSD_{P_1||P_2} = \frac{1}{2}KL_{P_1||P_2} + \frac{1}{2}KL_{P_2||P_1}$
[11] $KL_{P_1||P_2} = \sum_w P_1(w)log(\frac{P_1(w)}{P_2(w)})$

$JSD_{D_2\|\|D_1}$ $D2$: $D1$	Wiki0	Wiki1	Books	Subtitles
Wiki0	x	0.15	0.42	0.49
Wiki1		x	0.42	0.48
Books			x	0.35
Subtitles				x

Table 2.2: JSD divergence between unigram distribution of D_1 (row) vs. $D2$ (column)

We take Wiki0 as our reference dataset, and we compare the other datasets using the JSD and the OOV rate. We observe a clear gap between how different Wiki 0 is to Wiki 1 and how it differs from Books and Subtitles. For instance, about 60% of words of Wiki0 lexicon are not in Wiki1 lexicon, while it jumps to 83% and 88% for subtitles and books, respectively. In terms of unigram frequencies, according to the JSD, Wiki0 is almost three times more similar to Wiki1 than to Books and 3.5 times more similar to Wiki1 compared for Subtitles.

In conclusion, looking at two straightforward metrics of similarity between two datasets, the word-level Out-of-Vocabulary rate and the JSD of unigram distribution, we found that two English datasets originating from different sources differ very significantly at the lexical level.

In the following chapters, we will study how this shift impacts the performance of our NLP models and what techniques can be used to cope with it.

3 Using NLP Technologies

3.1 Definition of NLP technologies

We define NLP technologies as any system that can use data from natural languages, like written text or recorded speech, and that "does something" with it. The purpose can be *descriptive*; for instance, we may want to measure the frequency of a given predicate-argument relation in a given corpus. Alternatively, it could be *predictive*, for instance, given a question in English, we may want to predict the answer.

3.2 Applications of NLP

Nowadays, NLP technologies have become ubiquitous. They are used in most computers, smartphones, or other digital assistant devices. We broadly divide the applications of NLP into four categories.

Linguistic Studies One primary application of NLP is to gain a better understanding of languages. As we already described, languages are highly variable and diverse. Learning about their structure usually requires analyzing a large quantity of language production. For this purpose, NLP technologies provide powerful tools. For instance, NLP can be used for sociolinguistics (Trudgill, 2000), which aims at analyzing the lexical, syntactic, or semantic structure of sentences and their relation to a socio-economic context. It can be used to decipher an extinct language (Luo et al., 2019). It may also be used to find the historical relations between languages by automatically discovering etymological cognates — i.e., words with the same etymological root (Bouchard et al., 2007).

Human-Knowledge Interface Building tools that allow us to search for information has been one of the earliest large-scale applications of Natural Language Processing. Searching for information online or in a large-scale offline database has revolutionized our access to knowledge (Fisher et al., 2015; Hamilton and Yao, 2018). Google Search is used by around 4.3 Billion people today.[1] Search technologies — a.k.a. Information Retrieval (IR) — have made great progress in the past 25 years (Page et al., 1999; Singhal and Google, 2001; Datta et al., 2008; Devlin et al., 2018a; Karpukhin et al., 2020). It is now possible to search using complex natural language queries[2] to retrieve news articles, websites, images, videos,[3] and even snippet of text (Voorhees and Tice, 2000). Another type of human-knowledge interface that is emerging is automatic summarization (Nenkova et al., 2011). For instance, automatic summarization can provide us with only a glimpse of the original documents given single or several documents on a given topic. These types of applications are still in their infancy, but they have the potential to change our access to knowledge fundamentally.

Human-Computer Interface Building a richer Human-Computer Interface has always been at the core of NLP (Manaris, 1998a). For instance, using voice to command a computer has become mainstream[4] these last years with the progress of Siri, Google Assistant, or Alexa. It is now possible to play a song, get the weather, turn on the lights using voice, and even search the internet. Better human-computer interface, for instance, using automatic text simplification (Martin, 2021), can help people with disabilities such as aphasia (Manaris, 1998b) and dyslexia (Rello et al., 2013) in communicating online and searching the internet.

Human-to-Human Interface Facilitating Human-to-Human communication has been one of the first ambitions of NLP (Weaver, 1952). The most mainstream

[1] With a market share of 92% of the search market

[2] https://blog.google/products/search/search-language-understanding-bert/

[3] An increasing number of young people now search engines in Instagram and Tik-Tok to find video content (cf. https://techcrunch.com/2022/07/12/google-exec-\suggests-instagram-and-tiktok\-are-eating-into-googles-core-products-search-and-maps/?guccounter=1)

[4] https://voicebot.ai/2022/06/20/over-half-of-u-s-adults-have-\smart-home-devices-nearly-30-use-voice-assistants-with-them-new-report/

and adopted application is Automatic Translation.[5] Using DeepL, Google Translate, or Bing, it is now possible to translate from around 150 languages to 150 languages.[6][7]

Building "Intelligent" Systems Finally, NLP is at the heart of what is commonly called "Artificial Intelligence". Building systems that can mimic some aspect of human intelligence is an old ambition that was framed with the first generation of computers in the 50s (McCarthy et al.)[8]. Creativity (Ramesh et al., 2021), abstract reasoning (Chollet, 2019), solving complex equations (Lample and Charton, 2020), sense of humour (Chowdhery et al., 2022), telling stories (Hutson, 2021) are tasks that only recently entered the scope of modern NLP systems and on which large-scale generative language models (Brown et al., 2020; Rae et al., 2021; Smith et al., 2022; Chowdhery et al., 2022) are making fast progress (Srivastava et al., 2022).

3.3 NLP Tasks

These broad applications of NLP can be approached by building systems divided into distinct modules, each taking care of a specific *task*. NLP tasks can usually be studied independently from one another by for instance building models to perform them. Using annotated evaluation dataset, we then evaluate the performance of a given model based on an evaluation metric. For all the NLP tasks we cover in this thesis, the NLP community uses standard metrics shown to capture the quality of a model at the given task. For most of these tasks, we will see that designing and training these models using data in a machine learning framework is usually the solution that leads to the best performance (§ 4).

In this section, we list the main NLP tasks we study in this thesis along with the standard evaluation metrics associated with them.

[5] https://www.mordorintelligence.com/industry-reports/machine-translation-market
[6] https://ai.facebook.com/research/no-language-left-behind/
[7] https://ai.googleblog.com/2022/05/24-new-languages-google-translate.html
[8] http://raysolomonoff.com/dartmouth/boxa/dart564props.pdf

3.3.1 Part-Of-Speech Tagging

Part-Of-Speech (POS) tagging consists in identifying the grammatical categories of wordforms (§2.2.2) in a given sentence. Many formalisms have been developed to define grammatical categories across languages in the most general and consistent way. In this thesis, we mainly rely on the *Universal POS tagset* formalism defined within the Universal Dependency (UD) framework (Nivre et al., 2016) which is a revised version of the universal tagset introduced by Petrov et al. (2012). UD defines 17 universal POS tag listed in table 3.1. We distinguish between tags associated with closed wordform class — i.e., that there is a fixed list of possible wordforms in a given language associated with such a POS tag — and open wordform classes associated with an unlimited number of wordforms.

For instance, Det is a closed wordform class tag because, in nearly all languages, there is a fixed number of possible determiners (e.g., the, a, an, this, etc. in English). While Noun is an open class worform.

Formally, POS tagging is defined as follows. Given a sequence of wordforms $(X_1, .., X_t)$ in a vocabulary $\mathcal{V}$, the task consists in assigning tags $(L_1, ..., L_t)$ in a predefined tagset $\mathcal{L}$, i.e.:

$$\begin{aligned} POS: \quad \mathcal{V}^{\mathcal{T}} \quad &\rightarrow \quad \mathcal{L}^{\mathcal{T}} \\ (X_1, .., X_T) &\mapsto (L_1, .., L_T). \end{aligned}$$

Evaluation Metric Given D sequences $(x_1, .., x_T)_d \in \mathcal{V}^{\mathcal{T}}$ of N tokens in total (i.e. $T_1 + .. + T_D = N$) associated with $\hat{y}_i$ predicted tags and y_i gold tags. POS tagging is usually evaluated with the accuracy defined as:

$$Accuracy = \frac{\sum_{i=1}^{N} \mathbb{1}_{\{y_i = \hat{y}_i\}}}{N} \tag{3.1}$$

POS tagging is a sequence labeling task because its goal is to assign a single label per input token. It can also be seen as a structured prediction task. Indeed, each predicted

Tag	UD Label	Class
adjective	ADJ	Open
adposition	ADP	Open
adverb	ADV	Open
auxiliary	AUX	Closed
coordinating conjunction	CCONJ	Closed
determiner	DET	Closed
interjection	INTJ	Open
noun	NOUN	Open
numeral	NUM	Closed
particle	PART	Closed
pronoun	PRON	Closed
proper noun	PROPN	Closed
punctuation	PUNCT	Open
subordinating conjunction	SCONJ	Closed
symbol	SYM	Other
verb	VERB	Open
unspecified	X	Other

Table 3.1: UD POS Tags (Nivre et al., 2016). Closed wordform class tags are associated with a fixed list of possible wordforms while open wordform class are potentially associated to an unlimited number of wordforms.

label depends *a priori* on its neighboring labels. For this reason, the output sequence forms a "structured" sequence.

3.3.2 Syntactic Parsing

Etymologically, the word *syntax* originates from *syntaxis* in Ancient Greek which means "setting out together or arrangement". Describing and analyzing the syntactic structure of languages has a very long history. In the 6th century B.C., Pānini — who is often considered to be "the father of descriptive linguistics" — described, in the Aādhyāyī, rules that govern the structure of the Sanskrit language (Kahrs, 1990).

We have seen in section 3.3.1 that POS tagging consists in assigning morpho-syntactic categories to wordforms. These categories inform us of the syntactic function that wordforms (§2.2.2) takes in a sentence. Syntactic parsing goes one step further. In a nutshell, syntactic parsing consists in extracting the grammatical structure of a sentence or a phrase. This syntactic structure is usually represented with a tree (Jurafsky and Martin, 2000).

Traditionally, there are two complementary frameworks to extract the syntactic trees of sentences. On the one hand, *constituency* parsing defines syntactic trees by forming groups of words defining what is referred to as *constituents*. Each constituent may be made of one or several sub-constituents. Constituency trees define hierarchical structures from the entire sentence to every single wordform (i.e., the leaf nodes of the constituency tree). On the other hand, *dependency* parsing defines a syntactic tree by explicitly defining relations between wordforms. In this thesis, we will only build dependency parsing models. However, for the sake of completeness, we introduce constituency parsing briefly.

Constituency Parsing

Constituency Parsing is based on the notion of *constituency grammar*. Constituency grammars are essentially systems of rules that govern how contiguous words in sentences are grouped to form sequences of words referred to as *constituents*. Those systems also govern how groups of constituents form larger constituents. The most well-known, and probably most widespread formalism used to model fragments of a given language, notably English, is that of Context-Free Grammars. They comprise a set of rules (or productions) that define what group of words are allowed and a set of symbols (words in a predefined vocabulary and non-terminal symbols) that represent the constituents. The derivation of those rules when applied to a sentence produces a parse tree. Within that framework, a constituent is simply a sequence of words/tokens dominated by a non-terminal node in a given parse tree.

Constituent	Grammar Rules	Examples
S	→ NP VP	Maria has left a note
NP	→ NNP	Maria
VP	→ VBN VP	has left a note
VP	→ VBN NP	left a note
NP	→ DT NN	a note

Table 3.2: Illustrating Constituency Rules for English on the sentence “Maria has left a note”.

We illustrate this with the parse tree of the sentence *Maria has left a note* (Jurafsky and Martin, 2000) in Figure 3.1 derived from the toy grammar defined in Table 3.2: it is made of the noun phrase (NP) “Maria” and the verb phrase (VP) “has left a note” which can be split more granularly as illustrated in the table.

As we can see, building a constituency grammar requires enumerating all the grammatical associations of word grammatical categories and constituents. This includes listing all the possible allowed word orders for languages that have relatively free word order (e.g. Russian, Czech, Hungarian, Latin). Of course, *syntactic sugar* additions to a chosen formalism can alleviate tedious enumerations.

Given a constituency grammar, for a given sentence, constituency parsing consists in predicting its associated constituency tree.

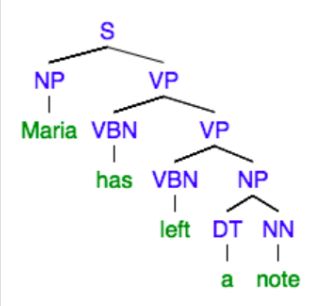


Figure 3.1: Constituency tree of the sentence "Maria has left a note" (Jurafsky and Martin, 2000).

Dependency Parsing

In contrast to constituency parsing, dependency parsing captures the syntactic structure of a sentence using dependencies relations between wordforms. This means that the dependency formalism does not explicitly define groups of words. It only characterizes relation between a *head* and a *dependent*.

Formally, dependency parsing consists of the following task:

$$\begin{aligned} DEP: \quad \mathcal{V}^{\mathcal{T}} \quad &\rightarrow \quad (\mathcal{V}, \mathcal{A}, \mathcal{L}) \\ (X_1, .., X_T) &\mapsto T = (V, A, L) \end{aligned}$$

In plain words, given a sequence of wordforms, dependency parsing associates a dependency tree T. T is defined as a triplet (V, A, L). V is simply the sequence of wordforms that, in this context, can be referred to as *vertices* or *nodes*. A is the set of directed relations between vertices referred to as *arcs* or *vertex*. L is the sequence of labels, one per relation. In dependency parsing, T must have a single node that does not have any incoming relations — the *root*. Additionally, T must be acyclic – i.e. no loop can be formed with a sequence of arcs. Finally, there must be a unique path from the root to every node.

We illustrate this in Figure 3.2 with the dependency tree of the sentence “Maria has left a note” from de Marneffe et al. (2021).

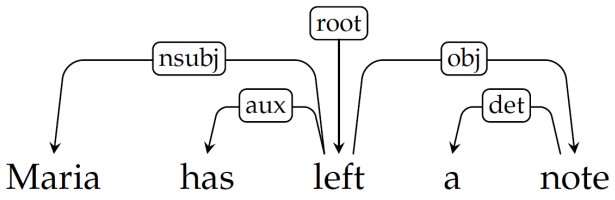


Figure 3.2: Dependency tree of the sentence "Maria has left a note" from de Marneffe et al. (2021).

Defining a dependency tree depends on the dependency grammar. Similar to constituency grammar, there are many ways to define it. In short, for a given language, defining a dependency grammar consists in defining the *arcs* and the type of arcs (the labels) between each pair of (head, dependent).

Arcs Based on the notion of constituent defined in the previous section, heads can be defined as the most important wordform in a given constituent. More specifically, to define head-dependent arcs, we start by identifying the heads of each constituent. The head of a constituent is the most important word. For a Noun-Phrase, it is the main noun. For a verb-phrase, it is the main verb. All the other wordforms in a constituent will depend directly or indirectly on the constituent head.

We illustrate in Figure 3.2 the dependency tree of the sentence *Maria has left a note* for which we already introduced the constituents in the previous section (cf. Table 3.2).[9] The root node is the main verb of the sentence (here *left*). Based on it, we can build the arcs within the verb-phrase (VP) "has left a note". "note" is the most direct dependent of "left" followed by "a" which depends on "note". Then we can look at the Noun-Phrase "Maria". There is no other word in this NP, so Maria is a *leaf* node (a node without dependence). Finally, we attach the auxiliary "has" to the verb "left".

Labels Each arc is labeled to characterize the type of grammatical relation between the head and the dependent. These relations are usually based on standard linguistic notions such as verb-subject relations, verb-object relations, and determiner-noun relations. Linguists have extended these basic notions to cover all types of grammatical relations and languages. How these relations are defined typically depends on the linguistic school of thought and the language.

Aiming at building multilingually consistent annotated datasets suitable for dependency parsing (*treebank*), Nivre et al. (2016) initiated the Universal Dependency Project (UD). The UD project consistently built a framework to annotate the largest number of natural languages with dependency trees, along with wordform segmentation, morphological analysis, and lemmatization. The UD project is a great success, with more than 122 language varieties currently having a UD treebank for 150+ treebanks. UD defines 63 types of head-dependent relations. We will use UD treebanks for a large variety of languages in the following chapters.

[9] We introduced conceptually dependency relations using the notion of constituent. However, when building a dependency parser, we usually do not rely on constituents explicitly.

Evaluation Metric The most standard metrics for dependency parsing are the Unlabeled Attachment Score (UAS) and Labeled Attachment Score (LAS).[10] Given a gold tree $G_{gold} = (V, A_{gold}, L_{gold})$ and a predicted tree $G_{pred} = (V, A_{pred}, L_{pred})$.

$$UAS = \frac{\#\{a, a \in A_{pred} \cap A_{gold}\}}{\#\{V\}} \tag{3.2}$$

$$LAS = \frac{\#\{l^a_{pred} = l^a_{gold}, a \in A_{pred} \cap A_{gold} \text{ with } l^a \text{ label of arc } a\}}{\#\{V\}} \tag{3.3}$$

UAS is the ratio of the number of correct arcs (i.e., head word and dependent word should agree with the gold tree) over the total number of arcs (which equals the total number of wordforms). LAS is the ratio of the number of correct arcs labeled correctly over the total number of arcs. Plank et al. (2015) showed that LAS correlates best with human judgment compared to other automatic evaluation methods. However, it shows that some human preferences are not captured by this metric (e.g., content POS are more important for humans than function POS in a predicted dependency tree).

3.3.3 Named-Entity-Recognition

Named-Entity Recognition (NER) (Chinchor and Robinson, 1998; Daelemans and Osborne, 2003) consists in predicting if a span of text is a named entity or not. How the named entities are defined and how granular there are may vary based on the use cases. In the NLP community, the named-entity labels are usually *organizations* (e.g. companies) tagged ORG, *locations* tagged LOC, *persons* tagged PERS. All other words are simply tagged as *others* (O). To perform NER of spans of text, it is standard to transform the task into a word-level sequence labeling task. To do so, we use the BIO framework (for beginning (B), inside (I), and outside (O)) to get a single label for each word. We show an annotated sample of the sentence "He formerly played for Almere City in the Netherlands" in Table 3.3.

[10] UAS and LAS were introduced by Eisner (1996).

He	O
formerly	O
played	O
for	O
Almere	B-ORG
City	I-ORG
in	O
the	O
Netherlands	B-LOC

Table 3.3: Annotation of "He formerly played for Almere City in the Netherlands" with Named-Entities using BIO (for beginning (B), inside (I), and outside (O)) labels.

Formally, given a sequence $(X_1, .., X_t)$ in a vocabulary $\mathcal{V}$, NER consists in assigning tags $(L_1, ..., L_t)$:

$$\begin{aligned} NER: \quad \mathcal{V}^{\mathcal{T}} \quad &\rightarrow \quad \mathcal{L}^{\mathcal{T}} \\ (X_1, .., X_T) &\mapsto (L_1, .., L_T). \end{aligned}$$

In this thesis, we mainly use the large-scale multilingual dataset collected by Rahimi et al. (2019) and the French Treebank (Abeillé et al., 2000) (annotated in NER by Sagot et al. (2012)) for our experiments in French.

Evaluation Metric The standard metric to evaluate NER is the F1-score (micro-F1) (Tjong Kim Sang and Meulder, 2003) which is defined as the harmonic mean of the precision and recall defined as:

$$precision = \frac{\#\{\text{Correct Predicted Named-Entities}\}}{\{\#\text{Predicted Named-Entities}\}} \tag{3.4}$$

$$recall = \frac{\#\{\text{Correct Predicted Named-Entities}\}}{\{\#\text{Observed Named-Entities}\}} \tag{3.5}$$

Intuitively, F1 is an adequate metric to account for how imbalanced a dataset may be toward a given class or how imbalanced some predictions may be toward a given type (in this case, non-entities (tagged as O) vs. named-entities). Indeed, let us assume that a model would over predict named-entities. This leads to increasing the recall, as more

named-entities are likely to be predicted by the model. However, the precision would fall as it would probably also decrease how good these predictions are.[11]

3.3.4 Lexical Normalization

Lexical normalization is the task of translating non-canonical words into canonical ones. We illustrate it with the following example (Table 3.4). In this thesis, we focus mainly on non-canonical data originating from User-Generated Content. Given a source sentence, our goal is to predict a canonical target sentence.

Non-Canonical	*yea... @beautifulloser8 im abt to type it uuup !!*
Canonical	*yeah... @beautifulloser8 i'm about to type it up !*

Table 3.4: Non-canonical UGC example and its canonical form

Formally, lexical normalization can be framed as a sequence labeling task from a source vocabulary of non-canonical words $\mathcal{V}_{\text{non-canonical}}$, to a target vocabulary of canonical ones $\mathcal{V}_{\text{canonical}}$.

$$\begin{aligned} NORM: \quad & \mathcal{V}^{\mathcal{T}}_{\text{non-canonical}} \quad \rightarrow \quad \mathcal{V}^{\mathcal{T}}_{\text{canonical}} \\ & (X_1, .., X_T) \mapsto (Y_1, .., Y_T). \end{aligned}$$

Evaluation Metric We define the three evaluation metrics on which we make our analysis. We distinguish between *need_norm* words, words that require to be normalized, and *need_no_norm* words that do not require normalization. The words normalized by our model (i.e., our model gave a prediction different from the source word) are noted *pred_need_norm*. We denote the words that require a normalization as *need_norm* words.

[11] We note that the micro-F1 metric does not distinguish between different named-entities classes (e.g., between LOC and ORG). For datasets with imbalanced types of entities, one may favor the macro-F1 score, which is defined as the arithmetic mean of the per-class F1 score.

Finally, the words that the model correctly normalizes are denoted *TP*. We then define recall and precision as:

$$recall = \frac{TP}{\#need\_norm} \tag{3.6}$$

$$precision = \frac{TP}{\#pred\_need\_norm} \tag{3.7}$$

F1 is simply the harmonic mean of the recall and precision (similarly to F1 defined in §3.3.3).

3.3.5 Transliteration

Transliteration consists of converting sequences of text written in one script (e.g., the Latin script) to another (e.g., the Cyrillic script). For some pairs of scripts, there is a 1 to 1 character level mapping between the source and the target script. Some other scripts require handling more complex and sometimes non-deterministic cases (e.g., Arabic script to Arabizi described in §4.7.1).

Evaluation Metric Transliteration systems are usually evaluated with the word-error rate (WER) as seen in (Ashby et al., 2021). WER measures the ratio of correctly transliterated words compared to a reference.

3.3.6 Natural Language Inference

Natural Language Inference (NLI) introduced by (Dagan et al., 2005a; MacCartney and Manning, 2008) aims at predicting if a sentence (a hypothesis) can be inferred by another sentence (a premise). It can be framed as a sequence classification task. In this thesis, we use the XNLI dataset (Conneau et al., 2018b), a multilingual NLI dataset derived from the SNLI dataset (Bowman et al., 2015a). Formally, given a premise $(p_1, ..., p_T)$ and a hypothesis $(h_1, .., h_{T'})$:

$$\begin{aligned} NLI: \quad & \mathcal{V}^T \quad \rightarrow \quad \mathcal{L} \\ & (p_1, ..., p_T, h_1, .., h_{T'}) \mapsto L \end{aligned}$$

In XNLI, there are three possible labels: (neutral, contradiction, entailment). NLI models are usually evaluated using a standard accuracy metric (Dagan et al., 2005a; MacCartney and Manning, 2008).

3.3.7 Question Answering

There are many ways to frame question answering (QA). We present here a currently popular framing of QA referred to as extractive question answering or reading comprehension introduced in (Hirschman et al., 1999) and (Voorhees and Tice, 2000). Given a question and a passage of text referred to as the *context*, Reading-Comprehension QA consists in extracting a span of text in the passage that answers the question.

Evaluation Metrics Reading-Comprehension QA is usually evaluated by comparing the predicted span of text with the references with exact-match or F1 score (Rajpurkar et al., 2016; Clark et al., 2020a).[12] In the case of the F1-score, for a given prediction, we first compute the recall and precision between the prediction and each reference — i.e. counting the number of tokens in the prediction that appear in the reference regardless of word order. Formally, given a prediction made of D_0 tokens $(\hat{x_1}, .., \hat{x_{D_0}})$ and K references $\{(x_1^1, .., x_{D^1}^1)_1, .., (x_1^K, .., x_{D^K}^K)\}$:

$$recall = \frac{\#\{\{\hat{x_1}, .., \hat{x_{D_0}}\} \cap \{x_1^k, .., x_{D^k}^k\}\}}{D^k}, precision = \frac{\#\{\{\hat{x_1}, .., \hat{x_{D_0}}\} \cap \{x_1^k, .., x_{D^k}^k\}\}}{D_0} \tag{3.8}$$

The next step is to compute the F1 score, the harmonic mean between the recall and the precision, and pick the maximum F1 score over all the K references. Finally, we aggregate the F1 score over all the (question, prediction) samples by simply computing the arithmetic mean. For instance, given the prediction *San Francisco* and the two references ["*San Francisco, CA*", "*San Francisco, California*"], the exact match would be 0 because the prediction matches exactly with no reference. The F1 score would be

[12] The predictions are lower-cased, and articles (e.g. "the", 'a") and punctuation marks are usually removed before evaluating.

$hmean(2/2, 2/3) = 0.8$ because *San Francisco* (2 tokens) appear in the reference *San Francisco CA* (3 tokens).[13]

[13] The result is the same if we pick the other reference *San Francisco California*) so taking the maximum does not impact the result for this specific sample.

4 Modeling Textual Data

4.1 Probabilistic Framework

We recall the modeling framework introduced in chapter 2.

In all our experiments, we assume we have a *corpus*. A corpus has the following structure:

- A *corpus* is defined as a collection of *documents* $\{D_1, .., D_C\}$.
- A *document* is defined as a sequence of *tokens* $(t_1, .., t_D)$. A document could be made of a few tokens, a sentence, a paragraph, or a collection of paragraphs.
- A *token* is a basic unit of discrete data. A token can be defined as a wordform (following a morpho-syntax framework § 2.2), a punctuation mark, a sequence of characters, or even a character itself. We refer to the set of all possible tokens as the *vocabulary* $\{t_1, ..., t_V\}$. Each token can be defined with a unique index in $[|1, V|]$. We do not assume anything about the language or the script from which each token originates.

We note that the choice of the corpus and how we define documents and tokens are use-case and task-specific. For some tasks and models, we may want to define tokens as characters and documents as entire web pages. In some other cases, tokens may be defined as wordforms, and documents may be defined as sentences.

The most basic model that we can define is referred to as a *language model* (Jelinek, 1976). Let X be the random variable that characterizes sequence of tokens (i.e. documents) $X = (t_1, ..., t_D)$

Broadly, speaking, language modeling consists in estimating the probability distribution of X.

$$p(X) = p(t_1, .., t_D) \tag{4.1}$$

This means that $p(t_1, .., t_D)$ gives us the probability of observing the sequence $(t_1, .., t_D)$.

Given tokens and documents, we may want to do more concrete and useful tasks, such as classifying a document or a token or even predicting another sequence to translate a sentence or answer a question. Let (X, Y) be a pair of random variables. X may characterize tokens or documents. Modeling an NLP task consists of estimating the conditional probability $Y|X$ to predict Y with X.

$$p(Y|X) \tag{4.2}$$

Here, X may characterize tokens or documents. Y might be a label i.e. $Y \in \{0, .., L\}$, or a sequence of labels.

Sequence Labeling For instance, let C be a corpus of sentences. Each sentence is a sequence of words $X_1, .., X_D$. Let $Y = (L_1, .., L_D)$ be the associated sequences of labels. Sequence Labeling consists in estimating:

$$p(Y|X) = p(L_1, .., L_D|X_1, .., X_D) \tag{4.3}$$

After estimating $p(Y|X)$, and given a sentence X, we can predict the sequences of labels Y. For instance, we may get the probability of having the sequence of POS (§ 3.3) tags *(PRON, VERB, NOUN)* given the sequence of words (*I, like, coffee*).

Sequence Classification Similarly, Let Y be the associated labels that characterize which topic a document $X_1, .., X_D$ is about. Then, $P(Y|X)$ may give us the probability of having the label *cooking* to the document *Heat a few tablespoons of oil in a skillet over medium-high heat. Add tofu to the pan in a single layer. Do not overcrowd the pan.*

$$p(Y|X) = p(Y|X_1, .., X_D) \tag{4.4}$$

Sequence Generation In the same framework, we can model tasks that output a sequence of tokens $Y = (Y_1, .., Y'_D)$ given a sequence of input tokens $X = (X_1, .., X_D)$ such as Machine Translation or Question Answering.

$$p(Y|X) = p(Y_1, .., Y'_D|X_1, .., X_D) \tag{4.5}$$

For instance, for machine translation given an input sentence $(X_1, .., X_D)$ in English and an output sentence $(Y_1, .., Y'_{D'})$ in French: $p(Y_1, .., Y'_{D'}|X_1, .., X_D)$ provides the probability of having, for instance *(Mangeons, du, tofu, !)* to the input *(Let, 's, eat, tofu, !)*.

Graph Prediction Given a sequence of tokens $(X_1, .., X_D)$, graph prediction consists in predicting the directed relations $A_{i,j} \in \{0, 1\}$ between each token X_i and X_j for all i, j. These relations are typically directed (i.e. $A_{i,j}$ usually differs from $A_{j,i}$). For some tasks (e.g. syntactic parsing), each relation is also labeled. This means that each $A_{i,j}$ is associated with a label $L_{i,j}$ in a predefined set of label $\mathcal{L}$.

Graph prediction consists of modeling the following joint distribution:

$$p(A, L|X) = p(\{(A_{i,j}, L_{i,j}), \forall (i, j) \in [|1, D|]^2\}|X_1, .., X_D) \tag{4.6}$$

In this thesis, we will perform graph prediction for dependency parsing (§3.3.2). Dependency parsing consists in predicting labeled trees. In dependency parsing (in the UD formalism (Nivre et al., 2016)), dependency trees are single-root (i.e., a single node has no incoming arcs), connected (i.e., there is a direct path from the root), acyclic (i.e., there is a unique path from one node to another) directed graph.

Based on the probability distribution of a graph, we can extract the most-likely tree using a maximum spanning tree like the Chu-Liu-Edmond algorithm from (Chu and Liu, 1965; Edmonds et al., 1967). In a nutshell, Chu-Liu-Edmond is a recursive algorithm that selects the arcs with maximum probability and removes recursively specific arcs to break the cycles in the graph to extract a tree.[1]

All the NLP tasks we work with in this thesis will be based on this modeling framework. Based on it, two modeling design questions must be answered for all NLP experiments.

1. How do we represent raw textual data? cf. §4.2-4.3.
2. Given a representation of our textual data, how do we parametrize and estimate our model to do prediction? cf. §4.5-4.6.

[1] Time complexity of the Chu-Liu-Edmond algorithm is $O(D^2)$.

4.2 Text in Computers

4.2.1 Encoding

Any NLP experiment starts with raw text. This text is usually stored in the memory of a computer. A critical question is how text stored in a computer is represented in the computer's memory.

Computers only work with *bits*. A bit corresponds to an electric impulse inside a computer. For this reason, a bit has only two possible values: 0 or 1. Hence, any data points in a computer must be stored as a sequence of bits. In computer science, it is usual to group bits together in groups of 8 bits that we refer to as *bytes*.

The way we represent written text in a computer's memory is referred to as *encoding*. In a few words, encoding is based on rules that we follow to "translate" a sequence of bits into readable symbols.

Since the invention of computers, many encoding standards have been proposed and used. Historically, the ASCII encoding developed in the US in the 60s was very popular in the early ages of computers. ASCII only supports 128 characters, including the English alphabet (upper case and lower case), punctuation marks used in English, and special characters such as white spaces. Since the 60s, many other encoding standards were developed to support other languages, non-Latin scripts, and newly introduced symbols such as emojis.

However, managing many different encoding standards is challenging and complex. To overcome these challenges, the Unicode Standard was created[2] in 1987. Unicode is not an encoding standard per se. The Unicode standard is essentially a database which associates *codepoints* to *characters*. For instance, the character Z:

$$Z \rightarrow \text{U+005A}$$

The Unicode standard includes, as of version 14.0[3], 144,697 characters. Each character is associated with a unique codepoint.

[2] https://home.unicode.org/basic-info/overview/

[3] http://www.unicode.org/versions/Unicode14.0.0/

As defined by the Unicode Consortium[4] which supports, maintains, and expands the Unicode standard, the Unicode standard is meant to be:

- Universal: i.e., represents all the characters of all human written language varieties,
- Stable,
- Unified across languages and scripts,
- Compositional: i.e., allow for composition with diacritics and accents.

CHARACTERS are defined in Unicode as "the abstract representations of the smallest components of written language that have semantic value". They represent letters, punctuation marks, logograms such as, for instance, *Hanzi* characters in Mandarin, emojis, and math symbols.[5]. They are associated with properties that provide information about the "semantics" of the characters such as the direction in which the character is meant to be displayed (e.g. right to left for an Arabic character).

CODEPOINTS are hexadecimal[6] numbers prefixed with U+. They are related to Unicode characters with a bijections, i.e. a codepoint refers to a single character in the Unicode character set and each character is associated with a single codepoint. For instance, Z is the character represented by 005A (i.e. the 90th character in the Unicode symbol).

ENCODING CODEPOINTS Based on Unicode representations, there are —again— many ways to encode Unicode into bytes. The most popular one is UTF-8 (Unicode Transformation Format - 8). UTF-8 encodes code-points using variable length encodings. For small codepoints, it will only use a singe byte, for larger it will use up to 4 bytes. As an example, the letter Z is encoded in UTF-8 as a single byte: 01011010 .

The Unicode Standard combined with the UTF-8 encoding have become a global success with more than 95% of website using it today.[7]

[4] https://home.unicode.org/

[5] The complete list can be found at https://www.unicode.org/charts/#symbols.

[6] Hexadecimal refers to the base 16: from 0-9 we use 0-9 symbols, from 10 to 15 we use A, B, C, D, E, F

[7] https://w3techs.com/technologies/cross/character_encoding/ranking

4.2.2 Tokenization

The first modeling decision that needs to be taken when we approach any NLP task is to choose what *unit* we will build our model on. Indeed, when we do NLP we are given textual data formatted as *strings*, i.e., raw sequences of characters.

We assume that we work with Unicode so each character is defined by Unicode. The first step is, therefore, to define groups of characters or *tokens* that we will model. The process of segmenting a raw sequence of characters into tokens is called *tokenization*.

At a high-level, tokenization can be done in two ways. On the one hand, we can use linguistic rules (e.g., special typographic characters, a word, a named entity) and segment sequences of characters based on these rules. On the other hand, we can perform tokenization by computing the frequency of sequences of characters based on a large corpus before tokenizing an input sentence by picking the most frequent sequences of characters.

4.2.3 Typographic Tokenization

The most straightforward segmentation method uses specific typographic characters. For instance, in many writing systems, white-spaces divide typographic units. White spaces are used in most alphabetic scripts (e.g., with the Latin script, the Cyrillic script), consonantal scripts (e.g., the Arabic script) or alphasylabic scripts (e.g., the Devanagari script). However, for several writing systems, white spaces are not used (e.g. Mandarin, Thai). We note that most of these writing systems are logographic scripts. For these scripts, simple tokenization usually relies on segmenting each character as a token.

4.2.4 Wordform Tokenization

For some tasks, it might be necessary to do tokenization at the *wordform* level. A wordform can be defined as a syntactic atomic unit (de Marneffe et al., 2021). It is, therefore, dependent on a theory of syntax. There are many ways to define what a wordform is in a given language based on the syntax framework we work in, the language. We present what wordform segmentation looks like in the Universal Dependency (UD) framework (McDonald et al., 2013a).

In the UD framework, wordform segmentation can be approached in two steps. The first step is to segment a raw sequence of characters into tokens. Second, when necessary, tokens are expanded into their multi-token wordforms.

As an example, the sentence "I haven't!" may be segmented in the following way (we use a CoNLL-like format (Hajic et al., 2009)) :

```
# text= I haven't!
1     I
2-3   haven't
2     have
3     not
4     !
```

Table 4.1: Wordform segmentation of "Ì haven't" in a CoNLL-like format (Hajic et al., 2009).

As we can see in table 4.1, "I haven't!" is segmented into 4 wordforms "I", "have", "not", "!" with the contraction "haven't" is expanded into two wordforms "have" and "not".

This wordform expansion process may be applied to diverse linguistic phenomenons. In English, as illustrated, it may be used for contractions. In French, it might be used to expand "du" and "au" into "de" "le" and "à" "le" respectively. For morphologically-rich languages such as Arabic, wordform tokenization is highly contextual and is, therefore, much more complex and ambiguous (Habash and Rambow, 2005). Finally, for languages without typographic separators, such as Chinese, wordform tokenization is also challenging (Han et al., 2013).

Doing wordform tokenization accurately in these cases is usually approached with trainable models. It requires a lot of annotated data, for instance, from the Universal Dependency treebanks (McDonald et al., 2013a). To develop accurate tokenizers, it is usual to frame tokenization as a character-level classification task as done in (de La Clergerie et al., 2017; Qi et al., 2020a).

In the UD framework, for morphological tasks or syntactic tasks such as POS tagging or dependency parsing, it is required to tokenize sequences of characters into wordforms.

4.2.5 Character n-gram Frequency-based Tokenization

Carefully segmenting wordforms may be necessary for syntactic-oriented tasks. However, modeling the segmentation specificities of each language is challenging and requires lots of linguistic knowledge. Additionally, due to the intrinsic Zipfian nature of language (Powers, 1998), it is simply impossible to encounter all the wordforms that will possibly be seen during inference in a given training corpus. From a modeling standpoint, this is a great challenge. Indeed, how can we expect a model to be "accurate" if it encounters multiple unseen wordforms? In NLP, we commonly refer to this challenge as the Out-of-Vocabulary problem (OOV). For these reasons, data-driven subword tokenization techniques were designed. We note that character n-gram tokenization is also helpful for languages with rich-morphology (Bojanowski et al., 2017).

BPE Tokenization Sennrich et al. (2016a) introduced the Byte-Pair-Encoding tokenization algorithm (inspired by a compression algorithm introduced in (Gage, 1994)). It is a data-driven tokenization technique. In practice, it is recommended to train the tokenizer using the same training data as the model it is supposed to be used for. It starts with a pre-tokenization step based on typographic segmentation described in section 4.2.3. For instance, in English, we pretokenize raw sequences of characters using white spaces.

BPE-tokenization is a *bottom-up* algorithm: it starts with characters and builds up BPE based on the frequency of sequences of characters.

We initialize a vocabulary V with all the unique characters in the training data. We refer to elements of the vocabulary as BPEs. For a predefined number of operations:

1. We compute the frequency of each pair of BPEs observed in the training data,
2. We add the most frequent BPE pair to the vocabulary V (i.e., merge the new BPE to the vocabulary).

Implemented naively, BPE-tokenization has a time complexity of $O(N^2)$ with N the length of the dataset in the number of pre-tokenized tokens.

With BPE-tokenization the size of the final vocabulary is equal to the number of unique characters (i.e. the initial size of the vocabulary) added with the number of merge operations. A close variant of BPE-tokenization is the WordPiece tokenization algorithm introduced by Schuster and Nakajima (2012) and used in the original BERT model

(Devlin et al., 2019a). Instead of selecting the most frequent BPE pair to compute the merge operations, it selects the BPE pairs that, once merged, maximize the log-likelihood over the entire training data.

UNIGRAM TOKENIZATION Similarly to BPE-tokenization, unigram tokenization (Kudo, 2018) is a data-driven subword segmentation algorithm. Unigram tokenization described by Kudo (2018) also starts with a pre-tokenization step that defines sequences of tokens $(\mathrm{x}_1, .., \mathrm{x}_T)$. For simplicity, we refer to tokens as words. We assume that we have a corpus of text C.

We assume that after training our tokenizer, we want to have a vocabulary (i.e., the set of unique tokens that can occur after tokenization of any data) of size M.[8]

The unigram tokenization algorithm aims to find the subword tokenization that maximizes the likelihood over our corpus. There are two unknowns: First, the set of subwords tokens that we are allowed to use (subwords could be any sequence of characters observed in the corpus). Second, the segmentation of each word given possible subwords is also unkwnon. Kudo (2018) proposed to estimate these two unknwons within a probabilistic framework.

The algorithm starts with initializing a seed vocabulary V by taking the most frequent sequences of characters observed in the corpus after pre-tokenization.[9] The idea of the algorithm is to iteratively remove tokens from this vocabulary until we reach the expected vocabulary size (i.e., when $|V|$=M).

To do so, we model sequences of subword tokens with a unigram language model. We assume that for a given word x and a subword tokenization of x, $\mathcal{S}(\mathrm{x}) = (x_1, .., x_L)$ with $x_i \in V$, we have:

$$p(\mathrm{x}) = \prod_{i=1}^{L} p(x_i) \tag{4.7}$$

[8] M is a hyperparameter of the unigram tokenization algorithm fixed manually.

[9] In practice, we can define it simply by taking all the characters observed in the corpus as well as the top $50\times M$ most frequent sequences of characters after pre-tokenization.

The underlying assumption is that the subwords occurrences are independent from each other. In practice, $p(x_i)$ are estimated by counting the occurrences of x_i in the corpus C over the total number of subwords in the corpus.

We then model the log-likelihood $\mathcal{L}$ over the entire corpus of text with:

$$\mathcal{L} = \sum_{\text{x} \in C} log(p(\text{x})) \tag{4.8}$$

Based on this, the unigram tokenization algorithm iterates through the two following steps:

1. Estimation Step: we compute the log-likelihood $\mathcal{L}$ over the entire corpus. To do so, for each word x, we find the subword tokenization $x_1, .., x_L$ that maximizes the unigram probability.[10],
2. Maximization Step: we remove the $\eta\%$ of subwords in V,[11] such that when removed, it maximizes the log-likelihood $\mathcal{L}$.[12]

We stop the algorithm when the subword vocabulary V reaches the desired size. We can then use the unigram tokenizer for each word by taking the most likely subword tokenization according to the unigram language model.

The key advantage of the unigram tokenization over BPE-tokenization is that it associates a probability $p(x)$ to a given segmentation (based on equation 4.7). This allowed Kudo (2018) to introduce *subword regularization*. During the training stage of a model (in their case, a Machine Translation sequence to sequence model), instead of using deterministic subword tokenization, we can sample over all the possible segmentation provided by the unigram tokenizer.[13] For machine translation, Kudo (2018) showed that subword regularization significantly impacts the translation accuracy (in terms of BLEU). Similarly, BPE-tokenization was further extended with BPE-dropout (Provilkov et al., 2020). By dropping with non-zero probability the merge operations between two tokens,

[10] In practice, this step is done efficiently with the Viterbi algorithm (Viterbi, 1967).

[11] $\eta\%$ is a hyperparameter typically set to 20%.

[12] In practice η is set to 20%.

[13] Given the subword tokenization of x $s_1, .., s_L$ Kudo (2018) defined the distribution over possible subword tokenization with $\frac{p(s_i)^\alpha}{\sum_i^l p(s_i)^\alpha}$. l controls the number of subword tokenization allowed, and α controls how "diverse" the sampling is.

it provides several possible tokenizations for each word. Provilkov et al. (2020) showed that BPE-dropout is on par with unigram tokenization for Machine Translation.

SentencePiece The main drawback of each of these techniques is that they require a pre-tokenization step. When we work with data specifically in scripts that do not rely on trivial typographic tokenization (Mandarin, Japanese), this pre-tokenization might be complex and language specific. To overcome this challenge, SentencePiece (Kudo and Richardson, 2018a) removed this requirement. It does so by simply applying unigram-tokenization or BPE-tokenization at the sentence-level. For languages with white-spaces, it replaces them with the special character "_". Additionally, Kudo and Richardson (2018a) integrated the NFKC character-level normalization process[14] that removes unicode ambiguities between characters that have the same glyphs (i.e. that look the same) in different languages but that are associated with different unicode points.

Character-Level Tokenization However, subword tokenization techniques inherently bias the model it is used for. Indeed, in some cases, subword tokenization may work very well for Latin-script written languages but much less for Chinese characters. For this reason, many approaches have segmented text at the character level. This alleviates the OOV problem as long as every character encountered at test time is seen during training. This was studied in neural language models in (Mikolov et al., 2011b; Kim et al., 2016), for machine translation in (Lee et al., 2017) and in document classification by (Conneau et al., 2017). Recently, in the context of multilingual modeling, Clark et al. (2021) showed that character-level segmentation leads to significant empirical progress.

4.2.6 Sub-Character Level Tokenization

BPE, unigram, Sentencepiece, and even character-level tokenization assume a fixed set of characters. If we encounter a new character or symbol not in the training data at test time, then this character will be unknown to the tokenizer and the model. This is a challenging limit when we work with real-world data and a large diversity of scripts. For instance, with character-rich languages such as Chinese, it is very likely to encounter an unknown character when we work with real-world data.

[14] http://unicode.org/reports/tr15/

It is possible to work at the byte level to overcome this challenge. This was studied for Question Answering by Kenter et al. (2018), and for sequence labeling by Gillick et al. (2016). However, working at the byte level leads to extending the sequence length. For instance, in UTF-8, each Unicode character requires 1 to 4 bytes.

To alleviate this, Wang et al. (2020) introduced Byte-level BPE-tokenization (BBPE) that extends BPE-tokenization with byte-level tokenization. BBPE does so by considering the raw text as sequences of bytes. More specifically, given a sequence of Unicode characters, BBPE applies the BPE algorithm at the level of UTF-8 bytes (cf. section 4.2.1). This means that based on their frequencies, it learns merge operations (cf. section 4.2.5) between bytes to form characters and sequences of characters. By design, every character defined in the Unicode database is "known" by the tokenizer.

We point to Mielke et al. (2021) for further discussions of linguistically-driven and data-driven tokenization algorithms.

4.3 Representing Text into Vectors

Performing any NLP task requires representing textual data into a data structure. This data structure can be as simple and unstructured as a string. However, modeling sequences of tokens typically requires representing text into vectors.

Embedding representation of text or simply *embedding* refers to any vector-based representation of text.

As extensively described in the previous chapter, natural languages are very diverse. For this reason, there is no unique and universal way of embedding textual data. Any embedding should be chosen on a case-by-case basis. Still, we will see that some approaches are more valuable than others.

Preliminary Example Before digging into specific embedding techniques, let us define what properties we look for in these embedding techniques. We want the embedding vectors to:

1. Capture linguistic information about each word. This information could be morphological, syntactical, or semantic information that characterizes each word we embed (Mikolov et al., 2013b; Pennington et al., 2014; Bojanowski et al., 2017),

2. Be useful for specific downstream NLP tasks. This means we expect the task-specific NLP models that use given embedding vectors to generalize better compared to if they were not using it (cf. (Dozat and Manning, 2016) for dependency parsing and (Lample et al., 2016) for NER as examples).[15]

In consequence of our first point, we expect two words that are morphologically, syntactically, or semantically similar to have embedding vectors close to each other according to a similarity metric. In NLP, one popular metric used to measure the similarity between word embedding vectors is the cosine similarity, defined as:

$$cos(x, y) = \frac{x \cdot y}{||x||_2||y||_2} \tag{4.9}$$

To illustrate this, we take the words *cat*, *dog* and *computer*. We note $\overrightarrow{cat}$, $\overrightarrow{dog}$ and $\overrightarrow{computer}$ the embedding vectors of *cat*, *dog*, and *computer* respectively. Let us assume that we would like to build an NLP model that uses word embeddings to classify if a word is an animal or not. Intuitively, we expect from a "good" word embedding technique to provide word vectors for $\overrightarrow{cat}$ and $\overrightarrow{dog}$ with a higher similarity relatively to $\overrightarrow{cat}$ and $\overrightarrow{computer}$, and $\overrightarrow{dog}$ and $\overrightarrow{computer}$.

We now describe several standard word embedding techniques. We start with presenting 1-hot encoding (§4.3.1), the most basic word embedding technique. Second, we describe briefly hand-crafted feature representations (§4.3.2). Finally, we present data-driven word embedding models (§4.3.3) such as count-based methods and prediction-based methods.

4.3.1 1-Hot Encoding

The first and most straightforward way to represent tokens into vectors is what is referred to as 1-hot encoding. Let t_i be a token in a given vocabulary indexed by i and x_{t_i} its embedding vector.

$$x_{t_i} = (0, 0, ..., 0, \underbrace{1}_{\text{index i}}, 0, .., 0) \tag{4.10}$$

[15] We will detail this point in section 4.6.

We derive straightforward properties of these representations:

$$x_{t_i} \perp x_{t_j} \quad i \neq j \tag{4.11}$$

$$cos(x_{t_i}, x_{t_j}) = cos(x_{t_i}, x_{t_k}) = 0 \quad \forall i, j, k \quad \text{s.t.} \quad i \neq j, j \neq k, k \neq i \tag{4.12}$$

Each token embedding is independent and equidistant from any other token embedding. Intuitively, this means that no linguistic information is integrated into 1-hot vectors.

1-hot encoding can be seen as the first approach to representing tokens into vectors. In addition, we can also use 1-hot encoding to represent linguistic information (Denis and Sagot, 2009; Sagot and Martínez Alonso, 2017).[16] We now describe linguistic information that can be useful in practice.

4.3.2 Hand-Crafted Feature Representation

A more refined approach to representing tokens into vectors is to collect linguistic information about each token and encode this information into vectors. We refer to such linguistic information as *feature*. The choice of these features is highly related to what task we want to use these features for. They range from low-level lexical or morphological information to high-level semantic or pragmatic information.

Lexical, Morphological and Syntactic Features We list several features of interest at the lexical and morphological levels. For a given token t:

- $f_1 \in \{0, 1\}$: Does the token start with a capital letter? **[Capitalization]**
- $f_2 \in \{0, 1\}$: Is the token mixed case (e.g. eBay) **[Capitalization]**
- $f_3 \in \{0, 1\}$: Is the token all capital letters? **[Capitalization]**
- $f_4 \in \{0, 1\}$: Does the token includes numbers? **[Digits]**
- $f_5 \in \{0, 1\}$: Does the token end with "ish"? **[Common Ending]**
- $f_6 \in \{0, 1\}$: Does the token end with "ist"? **[Common Ending]**

[16] It can represent any information that can be encoded with a fixed number of labels.

- $f_7 \in \{0, 1\}$: Does the token end with "an"? **[Common Ending]**
- $f_8 \in \{0, 1\}$: Does the token ends with *ing* **[Common Ending]**
- $f_9 \in \{0, 1\}$: Does the token ends with *ed* **[Common Ending]**
- f_{10}: Length of the token **[Length]**

The features $f_1, .., f_{10}$ can be computed from the string directly. They can be complemented by more complex linguistic features such as morphological or syntactic features. These features can be collected using external linguistic knowledge found in lexicon and grammars (Sagot and Martínez Alonso, 2017), or they can be computed using other machine learning models upstream.

- $f_{11} \in \{0, 1\}$: Is this token a VERB? **[POS tag]**
- $f_{12} \in \{0, 1\}$: Is this token a NOUN? **[POS tag]**
- $f_{13} \in \{0, 1\}$: Is this token an ADJ (Adjective) ? **[POS tag]**
- f_{14}: How many grammatical categories does this token belong to? **[POS tag]**
- f_{15}: Is the token in a noun-phrase (NP) ? **[Syntactic Constituent]**
- f_{16}: Is the token in a verb-phrase (NP) ? **[Syntactic Constituent]**

Based on these hand-crafted features, we can build , x_t, a vector representation of the token t:

$$x_t = (f_1, ..., f_{16}) \tag{4.13}$$

Such features have been used efficiently to perform downstream tasks such as POS tagging, NER, or Semantic Role Labeling (SRL). For instance, (Ratnaparkhi, 1996; Tseng et al., 2005; Denis and Sagot, 2009) showed that we could reach nearly human-level performance for POS tagging by combining lexical-level features and modeling them statistically. Gildea and Jurafsky (2002) showed that a probabilistic model using syntactic features could reach non-trivial performance in SRL.

Semantic Feature For more complex tasks, it might be helpful to use semantic features. One notable effort to build a lexical database that captures refined semantic information of words and semantic relations between them is WordNet (Miller, 1995).

4.3.3 Data-Driven Word Embedding Models

The Distributional Hypothesis

The limiting factor of most hand-crafted feature representations of words is that they are costly to collect, they usually require linguistic expertise, and they are hard to maintain and scale to new words, domains, concepts, and new languages. Additionally, specifically for complex semantic tasks, hand-crafted features do not always include all the information needed for the downstream task of interest. For this reason, computing these representations automatically in a *data-driven* way using a large corpus of text was shown to be a better solution for nearly all NLP tasks.

The foundation of nearly all embedding methods is the *distributional hypothesis*. Famously described by Firth (Firth, 1935, 1957) and Harris (Harris, 1954), the distributional hypothesis can be stated as:

> *You shall know a word by the company it keeps.* (Firth, 1957)

In simple words, the distributional hypothesis means that based on the context of a word (e.g., its surrounding words), we can infer its meaning.

Modeling Framework

In the most general manner, we can therefore define a word embedding model that makes use of the distributional hypothesis as a function E such that:

$$\begin{aligned} E: \quad \mathcal{V} \quad &\rightarrow \quad \mathrm{R}^D \\ w &\mapsto E(w, context(w)) \end{aligned}$$

A data-driven approach defines E using data. Different data-driven embedding models will define the context and the training procedure differently.

We cover traditional count-based methods and more recent prediction-based systems such as the Word2vec model (Mikolov et al., 2013b).

Count-Based Word Embeddings

Count-based approach computes word vector representation with co-occurrence statistics. Co-occurrence statistics are simply count-statistics of the occurrence of a pair of words. A word vector can be computed based on the co-occurrence statistics over an entire corpus.

For instance, let us assume a hypothetical corpus with the vocabulary: $V = \{leash, walk, run, owner, pet, barked, the, lion\}$.
To infer the embedding vector x_{lion} we simply count the (hypothetical) co-occurrence statistics of $\#\langle lion, leash\rangle = 0, \#\langle lion, walk\rangle = 15, \#\langle lion, run\rangle = 7, \#\langle lion, pet\rangle = 2, \#\langle lion, barked\rangle = 2, \#\langle lion, the\rangle = 25$. Based on them, we get $x_{lion} = (0, 15, 7, 1, 2, 25)$.

The problem with straightforward co-occurrence-based word embedding vectors is that frequent words impact a lot the similarity between word vectors while these words — usually determiners e.g., the, a — are the least informative.

$$PMI(w_1, w_2) = log\ \frac{P(w_1, w_2)}{P(w_1)P(w_2)} = log\ \frac{\frac{1}{n_{pairs}}\#\{(w_1, w_2)\}}{\frac{1}{n_{word}}\#\{w_1\}\frac{1}{n_{word}}\#\{w_2\}} \tag{4.14}$$

To refine this, the point-wise Mutual Information (PMI) was introduced in Church and Hanks (1990) to smooth and normalize co-occurrence-based statistics (cf. equation 4.14). To avoid relying on low frequency pairs it was further refined with the Positive-PMI (PPMI) which is equal to $PPMI(.,.) = max(0, PMI(.,.))$ (Dagan et al., 1993). As we will see in the following section, Levy and Goldberg (2014) showed that $PPMI$ is implicitly learned by prediction-based methods such as with the skip-gram negative sampling model (Mikolov et al., 2013b).

The limit of such models is the memory footprint. Indeed, the dimension of the embedding vectors is the size of the vocabulary V, possibly very large. They can be combined with dimension reduction matrix methods such as Singular Value Decomposition (SVD) (Stewart, 1993) to get dense embedding vectors of smaller dimensions.

Algorithm 1 Skip-Gram Word2vec Training

Given a corpus C, made of a set of unique tokens V.
Hyperparameters: number of negative samples K, a window size l, dimension of word vectors d, learning rate (α_t)
Initialize Randomly: $\mathbf{W} \in \mathbb{R}^{(V,d)}$ and $\mathbf{C} \in \mathbb{R}^{(V,d)}$
# Training loop for a single epoch
for w *in* C *(indexed by* i*)* **do**
  # sample window size R
  Sample $R \in [|1, l|]$
  for c *in* $[|w_{i-R}, w_{i+R}|] \setminus \{w\}$ **do**
    # Negative samples
    Sample $N_K = \{v_1, .., v_K\} \subset V$ represented by $\{\mathbf{v}_1, .., \mathbf{v}_K\}$ in $\mathbf{C}$
    # Compute loss
    $l(\mathbf{W}, \mathbf{C}) = -\sigma(\mathbf{w}, \mathbf{c}) - \frac{1}{K} \sum_{v \in N_K} log\, \sigma$**(-w, v)**
    # Parameter update with SGD
    $\mathbf{W}_t = \mathbf{W}_{t-1} - \alpha_t . \nabla\, l(\mathbf{W}_{t-1}, \mathbf{C}_{t-1})$
    $\mathbf{C}_t = \mathbf{C}_{t-1} - \alpha_t . \nabla\, l(\mathbf{W}_{t-1}, \mathbf{C}_{t-1})$
  end
end

Prediction-Based Word Embeddings

Instead of doing co-occurrence estimation (on a matrix of size V^2) before reducing it to a dense matrix of smaller dimensions, learning a continuous matrix representation in one step is possible.

These techniques are usually based on training a model that predicts a word given its context (or the other way around). Implicitly, the parameter of such a model captures linguistic information about words. These techniques are commonly called prediction-based techniques (Baroni et al., 2014).

The most notable and successful prediction-based (static) word embedding technique is the word2vec model (Mikolov et al., 2013b). The Skip-Gram Word2vec (detailed in the Algorithm 1) implements four key ideas. First, each *context* word and *focus* word in a vocabulary V is parametrized with a dense vector (of dimension d). These vectors are randomly initialized. Second, the word vector parameters are trained on the classification task of predicting whether or not words are in the surrounding context of the focus word.

Third, the objective function is defined with Negative Sampling (NS), a computationally efficient simplification of the softmax-cross-entropy loss. Fourth, the parameters are trained by minimizing the NS loss with Stochastic Gradient Descent. Consequently, the word2vec model scales to corpora of billions of tokens without any memory bottleneck.[17]

The Skip-Gram Word2vec model was shown to capture very rich syntactic and semantic representations (Mikolov et al., 2013b). In addition, from a theoretical perspective, Levy and Goldberg (2014) demonstrated the link between the Skip-Gram model trained with Negative Sampling (SGNS) and the Point-wise mutual information (presented in §4.3.3). They showed that the SGNS model implicitly learns a shifted Positive Point-wise Mutual Information matrix.

A large number of variants of the Word2vec model were introduced. For instance, the *polyglot* embeddings are word2vec embeddings trained for a large number of languages (Al-Rfou' et al., 2013), the fasttext embeddings integrated sub-words embedding to alleviate the Out-of-Vocabulary drawback and integrate morphological information (Bojanowski et al., 2017).

Finally, prediction-based word embedding vectors were shown to work very well with task-specific deep learning architectures. We present how in the following sections.

4.4 Deep Learning Methods for NLP

Deep learning is nowadays the most popular modeling framework for NLP. We introduce the different modeling approaches of deep learning that we will use in the rest of this thesis. This section is mainly based mainly on (Rumelhart et al., 1985; Chauvin and Rumelhart, 1995; Collobert et al., 2011; Radford et al., 2018)

4.4.1 Framework

In this section, we assume we have a sequence of vectors $(X_1, .., X_T)$ in $\mathbb{R}^{d,T}$. Our goal is to find a function noted dnn parametrized by θ such that dnn_θ gives good predictions

[17] The time complexity of the skip-gram negative sampling is $O(E * T * (K + 1) * d)$ with E the number of epochs, T the number of words in the corpus, K the number of negative samples, and d the word vector dimension (assuming constant time operation for multiplication, addition, sigmoid and random sampling of the negative samples). Its memory complexity is $O(V * d)$.

compared to the observed sample $Y \in \Omega$. Ω is usually a multi-dimensional Euclidean space. We will specify it below depending on the task we are considering.

Formally, given a loss function l, we would like to find θ such that $\mathbb{E}(l(\hat{Y}, Y))$ is minimal with $f_\theta(X_1, .., X_T) = \hat{Y}$ and dnn_θ:

$$\begin{aligned} dnn_\theta : \quad & \mathbb{R}^{d.T} \quad \rightarrow \quad \Omega \\ & (X_1, .., X_T) \mapsto \hat{Y} \end{aligned}$$

We defined our loss function as any differentiable function:

$$\begin{aligned} l : \quad & \Omega^2 \quad \rightarrow \quad \mathbb{R} \\ & (y, y') \mapsto l(y, y') \end{aligned}$$

For regression tasks, l is usually the Euclidean distance. For classification tasks, which are the type of tasks we study in this thesis, the cross-entropy (CE) is the most standard loss used in practice.[18]

All the deep learning models we study in this thesis are parametric i.e., $\theta \in \mathbb{R}^D$ with $D \in \mathbb{N}$ fixed. In deep learning, the parametrization of the model — i.e., the space in which the function dnn_θ is being learned — is called the *architecture*.

4.4.2 Designing a Deep Learning Model

Before any training takes place, building a deep learning model for a given task requires answering the following *design* questions:

1. How to represent the input sequence into vectors?
2. What architecture do we want to use?
3. What output activation function and what loss function should we use?

[18] Given a label vector $y \in \{0, 1\}^V$ and an estimated probability vector $\hat{y} \in [0, 1]^V$, $l(y, \hat{y}) = CE(y, \hat{y}) = \sum_i y_i \, log(\hat{y}_i)$

Embedding Input Layer

Deep learning models take as inputs real-number vectors or sequences of real-number vectors. When we work with discrete symbols like in NLP, we need to represent our discrete symbols in vectors. Therefore, it is possible, in theory, to use all the token-level representation techniques described above in section 4.3.1-4.3.3. In practice, deep learning models were shown to work much better with continuous and trainable *embedding layers* (Bengio et al., 2001).

We define a continuous embedding layer as $Emb \in \mathbb{R}^{\delta_e \times |V|}$. This means that for each token $t \in V$ indexed by j in the vocabulary $V = \{t_1, .., t_{|V|}\}$ we have t_j *embedded* by the vector $Emb_{.j}$ (i.e. column of the matrix Emb indexed by j) of dimension δ_e (the dimension of the embedding vectors). In the following sections, given a token t, we will note $Emb(t)$ the embedding representation of the token t in the embedding matrix Emb.

This technique was introduced in early deep learning research on discrete data. For instance, Riis and Krogh (1996) used an embedding technique to encode amino acids to predict the structure of proteins with a deep learning model. Jensen and Riis (2000) used an embedding layer to perform text-to-phoneme transliteration. A few years later, Bengio et al. (2001) showed that a simple 1-hidden layer deep learning model using a continuous and trainable embedding representation layer of words outperforms state-of-the-art n-gram models for language modeling.

The term *embedding* was first introduced by Collobert and Weston (2008), who successfully trained a deep learning model on multiple NLP tasks using a word-level embedding layer. Nowadays, embedding layers are used for nearly all deep learning models that handle symbolic input data.

We now present the main deep learning architectures for NLP.

Multi-Layer Perceptron (MLP)

Deep learning models are made of the composition of simple transformations. We start by presenting in detail a Multi-Layer-Perceptron (MLP) (Rumelhart et al., 1985). MLP is, conceptually, the most simple deep learning architecture.

We start with a 2-layer MLP. MLP takes as input uni-dimensional variable. In NLP, we usually work with sequences of input tokens. To fall back to a uni-dimensional input

sequence, we can concatenate all the input vectors. In consequence, we assume that we want to model an input variable $X \in \mathbb{R}^{\delta_i}$. Additionally, we assume that we want to predict a variable $Y \in \mathbb{R}^{\delta_o}$

A 2-layer MLP can be seen as a function dnn_θ such that:

$$\begin{aligned} dnn_\theta : \quad & \mathbb{R}^{\delta_i} \quad \rightarrow \quad \mathbb{R}^{\delta_o} \\ & X \mapsto W_2\varphi_1(W_1X + b_1) + b_2 \end{aligned}$$

φ_1 is a fixed (i.e. non-trainable) non-linear function ($\varphi_1 : \mathbb{R}^d \rightarrow \mathbb{R}^\delta$). The model is parametrized by W_1, b_1, W_2 and b_2 (trainable parameters) with $W_1 \in \mathbb{R}^{\delta \times \delta_i}$, $b_1 \in \mathbb{R}^\delta$, $W_2 \in \mathbb{R}^{\delta_o \times \delta}$ and $b_2 \in \mathbb{R}^{\delta_o}$

δ is the dimension of the hidden layer of the model. It is a hyper-parameter fixed prior to training the model. It can be selected in practice with hyper-parameter search (Hastie et al., 2001) or by following the best practices for a given task (Chollet, 2021).

From a 2-layer MLP (also called a 1-hidden-layer MLP), deriving a L-layer MLP is straightforward: To do so, we need to compose L times the transformation following:

$$dnn_{(W_i\ b_i, i \in [|1,L|])}(X) = W_L\varphi_{L-1}(...\varphi_2 \circ W_2\varphi_1(W_1X + b_1) + b_2)...) + b_L \tag{4.15}$$

In a more readable way, we can introduce the variable h_i and describe the L-layer MLP as:

$$\begin{aligned} & h_{i+1} = \varphi_i(W_ih_i + b_i), \forall\, i \in [|1, L-1|] \\ & \text{with } \ h_1 = X \ \text{ and } \ \hat{Y} = dnn(X) = h_L \end{aligned} \tag{4.16}$$

h_i are called hidden states of the model ($h_i \in \mathbb{R}^{\delta_i}$). φ_l are fixed non-linear functions, $\varphi_l : \mathbb{R}^{\delta_{l-1}} \rightarrow \mathbb{R}^{\delta_l}, \forall\, l \in [|1, L-1|]$. W_l and b_l are trainable parameters. $W_l \in \mathbb{R}^{\delta_{l-1} \times \delta_l}$, $b_l \in \mathbb{R}^{\delta_l}$, with $\delta_l \in \mathbb{N}^*, \forall\, l \in [|1, L|]$. And δ_l is the dimension of the hidden layer l of the model.

Output Dimension and Output Activation Function We note that the output dimension (δ_L) and the output activation function (φ_L) have a critical role in the model. Indeed, it defines what output space the output variable will be living in. For this reason, they must be chosen carefully.

In practice, for a regression task (i.e. $Y \in \mathbb{R}^{\delta_o}$), we simply use the identity function and we set δ_L to be equal to δ_o.

For a classification task, the output variable is a label (e.g., in $[|1, L|]$). To model such an output variable, we predict the probability distribution over the labels. More precisely, we want to predict a vector $\hat{p} \in [0,1]^L$ such that $p_i \in [0,1]$ and $\sum_i p_i = 1 \ \forall\, i$. To do so, we set the output dimension δ_o equal to L. For the activation function, we usually define φ_L with the softmax function, given $s \in \mathbb{R}^L$, defined as:

$$softmax(s) = \left(\frac{e^{s_i}}{\sum_{k=1}^{L} e^{s_k}} \right)_{i \in [|1,L|]} \tag{4.17}$$

Multi-Layer Perceptrons are also called Feed-Forward neural networks (FNN) (as opposed to Recurrent Neural Networks that we will describe next). They inherit from the early work of McCulloch and Pitts on artificial neurons. McCulloch and Pitts (1943) describes a simplification of biological neurons that could compute, in theory, logical operations. Several years later, Rosenblatt (1958) introduced the $Perceptron$ — the equivalent of a single layer neural network with an "all-or-none" activation function[19] — with a useable algorithm to train it. However, the perceptron was highly limited. Indeed, Minsky and Seymour (1969) showed that the perceptron could only solve discriminative tasks that are linearly separable.

About 30 years later, by introducing the backpropagation algorithm and using differentiable activation functions, Rumelhart et al. (1985) showed how to use and train the multi-layer perceptron.

Recurrent Neural Network

A critical limit of MLP is that they only can take fixed vectors as input. For NLP, this means that we can only model as a fixed context, i.e., a predefined number of tokens.

[19] Defined with $\mathbb{1}_{w.x+b \geq 0}$

For many NLP tasks, this approach is not ideal. Indeed, some tasks require long-term dependencies — e.g., Language Modeling (Le et al., 2012). Deep learning provides an architecture to solve this: the Recurrent Neural Network (RNN) Minsky and Seymour (1969); Rumelhart et al. (1985). We present here RNNs for sequence classification or sequence labeling.

We start by presenting a L-layer *vanilla*-RNN. Given a sequence of tokens $(X_1, .., X_T) \in V^T$, the vanilla-RNN can be seen as the function rnn_θ such that:[20]

$$\begin{aligned} rnn_\theta : \quad [|0,1|]^{V \times T} &\rightarrow \Omega \\ (X_1, .., X_T) &\mapsto (\hat{Y_1}, .., \hat{Y_T}) \end{aligned}$$

Such that:

$$\begin{aligned} h_{i+1,t+1} = \varphi_i(W_i h_{i,t} + U_i h_{i+1,t} + b_i), \forall\, i \in [|1,L|] \,\forall\, t \in [|0,T|] \\ \text{with } h_{1,t} = Emb(X_t) \text{ and } \hat{p_{t+1}} = h_{L+1,t+1} \\ \text{with } \varphi_L = softmax \end{aligned} \tag{4.18}$$

W_l, U_l and b_l are trainable parameters. W_l are the weights of the feed-forward component of the layer l. U_l are the weights of the recurrent component of the layer l. $W_l \in \mathbb{R}^{\delta_{l-1} \times \delta_l}$, $b_l \in \mathbb{R}^{\delta_l}$, with $\delta_l \in \mathbb{N}^*, \forall\, l \in [|1,L|]$.

In a more synthetic way, we can define the vanilla RNN *cell* of layer i, given the activation function φ_i, with $RNN_i = \varphi_i(W_i h_{i,t} + U_i h_{i+1,t} + b_i)$. We get:

$$\begin{aligned} h_{i+1,t+1} = RNN_i(h_{i,t}, h_{i+1,t}), \forall\, i \in [|1,L|] \,\forall\, t \in [|0,T|] \\ \text{with } h_{1,t} = Emb(x_t) \text{ and } \hat{p_{t+1}} = h_{L+1,t+1} \\ \text{with } \varphi_L = softmax \end{aligned} \tag{4.19}$$

[20] We describe the RNN with a continuous embedding input layer.

The idea of the vanilla-RNN was discussed in (Minsky and Seymour, 1969). It was then described in detail by Rumelhart et al. (1985), who noted that a vanilla-RNN could be seen as an MLP for which the recurrent matrix weights are shared across each step of the forward pass.

LSTM and GRU Vanilla RNNs are limited by the vanishing gradient phenomenon described by Hochreiter (1998). Indeed, long-term dependencies in a sequence can only be learned by the model's parameters after going through multiple gradient computation steps — possibly vanishing. To address this problem, more complex architectures were introduced. The Long-Short-Term model (LSTM) (Hochreiter and Schmidhuber, 1997) defies explicitly a *memory* vector that is supposed to store long-term information. The Gated Recurrent Unit (GRU) (Cho et al., 2014) was introduced as a more parameter-intensive version of the LSTM (and, therefore, less costly to train). LSTM and recurrent neural networks, in general, are designed to model a sequence in a unidirectional way (e.g., from left to right). To tackle tasks that need to access both the left and right context, it is possible to combine two Recurrent Modules each in one direction (Graves and Schmidhuber, 2005).[21]

Attention

The limit of recurrent neural networks for sequence modeling is that they build a fixed vector representation of the input sequence. To overcome this limit, the attention mechanism was introduced by Bahdanau et al. (2015). The attention mechanism was initially introduced for machine translation to combine the hidden vectors of recurrent neural networks to build a prediction-specific representation of an input sequence. It was then adapted to sequence labeling and sequence classification (Yang et al., 2016).

This approach was shown to deliver better performance in practice for sequence classification (Yang et al., 2016) and machine translation (Bahdanau et al., 2015). The attention mechanism also enables a convenient way to interpret a model's prediction, as seen in (Bahdanau et al., 2015; Yang et al., 2016).

[21] For LSTM, this approach would be referred to as a Bidirectional LSTM (or Bi-LSTM))

In order to improve even further the modeling and computational abilities of neural networks, Vaswani et al. (2017) showed that the recurrent layers were not necessary to build accurate sequence models. They designed the transformer architecture as a feed-forward neural network based on attention mechanisms. We now present in detail this architecture.

Self-Attention: The Transformer Architecture

Until recently, recurrent neural networks combined with attention mechanisms delivered the best empirical performance for many NLP tasks. However, these models are computationally slower than feed-forward neural networks. Indeed, by definition, they require sequential operations (as described in equation 4.18). In practice, this means that recurrent neural networks cannot be fully parallelized: any implementation requires waiting for the computation of step $t-1$ to be completed before starting the computation of step t. This inherent limit was overcome with a new kind of architecture introduced by Vaswani et al. (2017): the Transformer. The Transformer is a feed-forward neural network that combines simple Feed-Forward transformation (or layers) described in section 4.4.2 and *Self-Attention* layers.

Intuitively, self-attention layers build a new *contextual* representation of each input vectors. By contextual, we mean a representation aggregating information from the entire input sequence. This new contextual representation consists of:

- For a given vector h_t and its query vector q_t we want to build the new representation vector $\tilde{h}_t$,
- This is done using a ponderation of the information encoded in the intermediate representation $(v_1, .., v_T)$ (the *value* vectors),
- This ponderation is computed by computing the similarity between the intermediate representations q_t (the *query*) and the *keys* $(k_1, .., k_T)$.

We now detail this intuitive explanation of the mechanism of a self-attention layer.

Given a sequence of input vectors $X = (x_1, .., x_T) \in V^T$ (noted $H = (h_{0,1}, .., h_{0,T})$), a L-layer Transformer Architecture can be described as:

$$H_{i+1} = FeedForward(A_{i+1}) \text{ and } A_{i+1} = SelfAttention(H_i) \quad \forall\, i \in [|1, L|]$$
$$\text{with} \quad \text{Self-Attention}(H_i) = softmax(\frac{Q\, K^T}{\sqrt{\delta_K}})V \tag{4.20}$$
$$H_0 = (Emb(x_1), ...Emb(x_T))$$

For each layer i, given the input sequence $H = (h_1, .., h_T)_i$ — which is the output of the layer $i-1$ — each self-attention transforms H into a new matrix A. For this purpose, the self-attention builds three intermediate vectorial representations of the sequence H: the *query* $Q = \begin{pmatrix} q_1 \\ \vdots \\ q_T \end{pmatrix}$, the *key* $K = \begin{pmatrix} k_1 \\ \vdots \\ k_T \end{pmatrix}$ and the *value* $V = \begin{pmatrix} v_1 \\ \vdots \\ v_T \end{pmatrix}$ vectors.

The key, query, and value representation vectors are computed based on the parameters W_Q, W_K, and W_V as follows:

$$q_t = W_Q h_t \ , \ \forall\, t \in [|1, T|] \text{ with } W_Q \in \mathrm{R}^{\delta_q \times \delta} \tag{4.21}$$

$$k_t = W_K h_t \ , \ \forall\, t \in [|1, T|] \text{ with } W_K \in \mathrm{R}^{\delta_k \times \delta} \tag{4.22}$$

$$v_t = W_V h_t \ , \ \forall\, t \in [|1, T|] \text{ with } W_V \in \mathrm{R}^{\delta_v \times \delta} \tag{4.23}$$

From these intermediate representations of H, the self-attention layer builds A with:

$$A = softmax\left(\frac{Q\, K^T}{\sqrt{\delta_k}}\right)V \tag{4.24}$$

$$\text{i.e.} \ \ \forall\, t, a_t = softmax\left(\frac{q_t\, K^T}{\sqrt{\delta_k}}\right)V = \sum_{t'=1}^{T} \frac{s_{t'}}{\sqrt{\delta_k}} v_{t'} \text{ with } \ s_{t'} = \frac{e^{q_t k_{t'}^T}}{\sum_{i=1}^{T} e^{q_t k_i^T}} \tag{4.25}$$

In an even more compact way, given input vectors $H = (h_1, .., h_T) \in \mathrm{R}^{\delta \times T}$, and self-attention parameters $W_Q \in \mathrm{R}^{\delta_q \times \delta}$, $W_K \in \mathrm{R}^{\delta_k \times \delta}$, $W_V \in \mathrm{R}^{\delta_v \times \delta}$, we have:

$$A = softmax\left(\frac{(H^T W_Q)\,(H^T W_K)^T}{\sqrt{\delta_k}}\right)(H^T W_V) \tag{4.26}$$

In practice, it was shown by Vaswani et al. (2017) that projecting input hidden states multiple times and running multiple self-attention — multi-head self-attention — entails better empirical performance.

A notable refinement of the transformer self-attention architecture was introduced with the DeBERTa model (He et al., 2021) that disentangles the relative position of tokens and the token embedding itself, leading to significant performance progress.

Positional Encoding In contrast with recurrent models, the Transformer does not implicitly model the input tokens' position.[22]

A solution is to inject the position signal into the model as an embedding vector. Many approaches have been proposed for this. In the original Transformer, Vaswani et al. (2017) proposed to use a fixed (i.e., non-trainable) positional embedding vector with the same dimension as the token embedding vectors (in order to allow summation between the two types of embedding).

For a sequence of input tokens $(X_1, ..., X_T)$, the positional embedding of token X_{pos} is a vector in R^δ defined it as:

$$PosEmb(X_{pos})(k) = \begin{cases} sin\left(\frac{pos}{10000^{\frac{k}{\delta}}}\right) & \forall\, k \in [|0, \delta - 1|], \text{ if } k \text{ is even} \\ cos\left(\frac{pos}{10000^{\frac{k-1}{\delta}}}\right) & \forall\, k \in [|0, \delta - 1|], \text{ if } k \text{ is odd} \end{cases} \tag{4.27}$$

We illustrate it in Figure 4.1 [23] with the positional embedding vectors of dimension $\delta = 768$ and a sequence length of 512. We note that $X_{pos} \in [-1, 1]^\delta$. Additionally, we note that for all pos, and all offset $K \in \mathbb{N}^*$, X_{pos+K} can be inferred by a linear

[22] RNNs models do so based on the recurrence relationship described in equation 4.18.

[23] The figure was generated using this notebook from Jay Alamar.

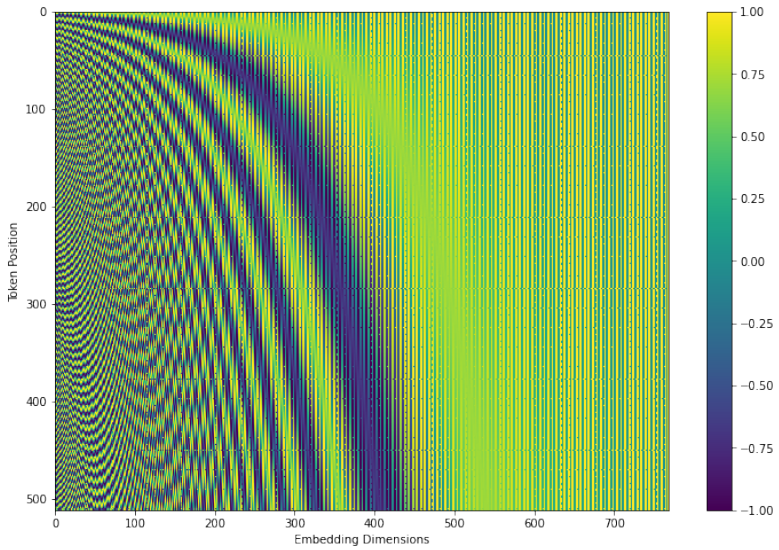


Figure 4.1: Absolute Positional Encoding proposed by (Vaswani et al., 2017). Each position in $[|1, 512|]$ is encoded in a vector of size 768 defined with equation 4.27.

transformation of X_{pos+K}, independent of pos.[24] Based on this, the intuition from Vaswani et al. (2017) was that this would help the model to generalize to positions not seen during the model's training (or seen infrequently), for instance, very large pos for long sequences.

Following the same idea of using embedding for positions, (Radford et al., 2018; Devlin et al., 2019a; Delobelle et al., 2020) designed BERT, RoBERTa, and GPT using trainable position embedding vectors initialized randomly (similarly to token embeddings). In all these approaches, each position receives a unique embedding vector which means that the positions are encoded in an *absolute* manner.

The limit of absolute position encoding techniques is that the model may struggle to generalize to long sequences that were unseen during training. To overcome this, *relative* position encoding were designed. In such approaches, only the distance between tokens is encoded into the model as opposed to the absolute position of each token in the sequence. (Shaw et al., 2018; Huang et al., 2019; Raffel et al., 2019) did so by

[24] We can derive the linear transformation by simply relying on the trigonometric formulas $cos(pos + k) = cos(K)cos(pos) - sin(K)sin(pos)$ and $sin(pos + k) = cos(K)sin(pos) + sin(K)cos(pos)$.

modifying the self-attention mechanism to model the relative distance between tokens. For instance, Raffel et al. (2019) included a trainable scalar for each distance between query and key position into the self-attention softmax (cf. definition of the self-attention layer in equation 4.25).[25] This means that the intermediate self-attention representation of a token is biased based on its distance from the surrounding tokens. Even more simply, Press et al. (2022) showed that biasing the self-attention weights in a non-trainable way based on the relative distance of queries and keys leads to much better performance on long sequences, even when the length was not seen during training.

To combine the best of both worlds, Su et al. (2021) proposed the Rotary Position Encoding (RoPE) technique. In a nutshell, RoPE is based on integrating rotation matrices in the self-attention transformation (based on the absolute positions of keys and queries). These rotations have the properties that they are only sensitive to the relative positions of keys and queries. This technique competes with or outperforms other positional encoding techniques (Andonian et al., 2021).

How to Model variable Sequence Length? Like any Feed-Forward network, Transformer models can only model fixed-length sequences. In practice, Transformers have a maximum-sequence length as small as 512 tokens (Devlin et al., 2018a) up to 2048 (Brown et al., 2020) in some cases. However, NLP tasks usually require sequences of various lengths (phrase, sentence, paragraph, document, web pages).

We can cut the sequence for very long sequences and ignore the tokens beyond the maximum-length positions. Still, for some tasks, it is required to have the full context. In these cases, a simple trick is to use a sliding window of size S (that can be as long as the maximum-sequence length) and process the sequences by blocks of length S.

In practice, we append the sequence with a special padding token for sequences shorter than the maximum-sequence length. In the case of the transformer, we make the model ignore these tokens by setting the self-attention weights to 0.

[25] This scalar is shared across layers.

Encoder-Decoder

Many NLP tasks are based on predicting a sequence of variable length (the output sequence) given another sequence of variable length (the input sequence). A notable and very studied sequence-to-sequence task is probably Machine Translation. To approach this type of task the encoder-decoder framework was introduced. The encoder-decoder is not an architecture *per se* but a family of architecture based on combining two modules — an encoder and a decoder — together. It was introduced in (Cho et al., 2014) for machine translation. Schematically, two deep learning architectures are defined: On the one hand, the encoder takes as input the input sequence. On the other hand, the decoder takes as input both the output of the encoder — that encodes the input sequence — and the output sequence.

Encoder-Decoder can be based on Recurrent Neural Network (Cho et al., 2014; Bahdanau et al., 2015) and on the Transformer architecture (Vaswani et al., 2017). The transformer architecture was introduced for machine translation using an encoder-decoder transformer.

4.4.3 Training Deep Learning Models

Nearly all modern deep learning models are trained with the Stochastic Gradient Descent (SGD) algorithm (Robbins and Monro, 1951) or a variant of it. In short, SGD is based on the gradient descent algorithm (Cauchy et al., 1847), which uses the gradient direction to minimize a given function. The SGD uses an estimation of the gradient on a few samples to optimize the objective function (i.e., in practice, to minimize a loss function §4.4.1). An efficient way to implement SGD with deep learning models is to use the chain rule. This implementation is known as the backpropagation algorithm, and it was first described by Rumelhart et al. (1985). We describe the backpropagation algorithm schematically in Figure 2.

In practice, the Adam optimizer (Kingma and Ba, 2015; Loshchilov and Hutter, 2019) is one of the most popular optimizers. It is a more refined version of the SGD that integrates first and second-order momentum estimation making convergence faster and more stable.

Algorithm 2 Backpropagation with SGD

Given observations $((x_i), (y_i))$ of two variables (X, Y)
Given a loss function l. An architecture dnn_θ
The goal is to find the best θ **s.t.** $E(l(Y, dnn_\theta(X))$ **is small**. Given a learning rate α
for *step* $<$ max **do**
    Sample (x, y)
    # Forward pass:
    $\hat{y} = dnn_\theta(x)$ and $l(y, \hat{y})$
    # Backward pass:
    $\nabla_\theta \, l(y, \hat{y})$ # compute gradients
    $\theta := \theta - \alpha \nabla_\theta \, l(y, \hat{y})$ # parameter update
end

Regularization Techniques for Deep Learning Models

Deep and Large deep learning models are tricky to train with backpropagation. They suffer from vanishing gradients (Hochreiter, 1998), exploding gradients (Pascanu et al., 2013), they can easily be stuck in saddle points (Dauphin et al., 2014).

To tackle these challenges, many architectural improvements were introduced, such as layer normalization (Ba et al., 2016) to tackle gradient explosion, residual connection (Wu et al., 2018) to address gradient vanishing, as well as best practices to randomly initialize parameters. The most widely used initialization for linear transformation is the He initialization introduced by He et al. (2015).[26] For many tasks, deep learning models are highly over parametrized[27].

4.5 Language Model

4.5.1 Modeling Framework

Causal Language Model

As seen earlier, language modeling estimates the probability distribution of sequences of tokens.

[26] It is the default initialization for linear layers in PyTorch https://github.com/pytorch/pytorch/blob/master/torch/nn/modules/linear.py#L44-L48.

[27] i.e., there are more parameters than training data points

$$p(X) = p(t_1, .., t_D) \tag{4.28}$$

Estimating the probability distribution of arbitrarily long text sequences is intractable. For this reason, we can decompose $P(X)$ using the Bayes' Rule of conditional probability (Bayes, 1763).

$$\begin{aligned} P(t_1, .., t_D) &= P(t_D|t_1, .., t_{D-1})P(t_1, .., t_{D-1}) \\ P(t_1, .., t_D) &= P(t_D|t_1, .., t_{D-1})P(t_{D-1}|t_1, .., t_{D-2})P(t_1, .., t_{D-2}) \\ P(t_1, .., t_D) &= \prod_{k=1}^{D} P(t_k|t_{1:k-1}) \end{aligned} \tag{4.29}$$

In consequence, estimating the probability distribution of sequences of tokens in a corpus is equivalent to estimating the *transition probability* $P(t_k|t_{1:k-1})$.[28] A model that estimates the transition probability of sequences of tokens is usually called *causal* language model in the sense that they model the language using its natural sequence order.

Denoising Language Modeling Causal Language Models can be seen as part of a more general family of language modeling approaches called *Denoising Language Modeling* (Hill et al., 2016; Devlin et al., 2018a; Lewis et al., 2020).

Given a sequence of tokens $(t_1, ..t_D) \in V^D$, a noise function $\phi : V^D \rightarrow (V \cup \{\varsigma_1, .., \varsigma_s\})^D$, a denoising language model aims at estimating:

$$p((t_1, .., t_D)|\phi(t_1, .., t_D)) \tag{4.30}$$

ϕ outputs a noise sequence of tokens by possibly integrating special tokens $\{\varsigma_1, .., \varsigma_s\}$. Recently, the most widely used $noise$ function has been the masking procedure introduced with the BERT model (Devlin et al., 2019a). More specifically, 15% of tokens are sampled for each sequence of tokens. Within these 15%, 80% are replaced by the special

[28] As a notation convention, we assumed $P(t_1|t_0) = P(t_1)$ (t_0 empty token).

[MASK] token, 10% are replaced by a random token in the vocabulary V, and 10% are unchanged. Many variant models have been introduced. Some models mask multiple tokens, such as the T5 model (Raffel et al., 2019) or SpanBERT (Joshi et al., 2019). With similar results, the BART language model (Lewis et al., 2020) was trained by integrating a noise function that randomly shuffles tokens.

4.5.2 Estimating Language Models

N-grams Language Models

The most straightforward approach to estimating causal language models is to compute the frequency of n-grams (Jelinek, 1980). The challenge is that count-based statistics of very long sequences are very poor estimators, even with large datasets. For this reason, the solution is to use a fixed window of a few n-gram (3 or 4 words, for instance) as the left context. After integrating smoothing techniques (Chen and Goodman, 1996), the n-gram causal language model can reach good empirical performance when estimated on a very large corpus. They also have the advantage of being very fast at test times. Indeed, predicting with a n-gram language model consists of a look-up in a large table of n-grams to compute the distribution over the vocabulary.

Feature-Based Models

To refine n-gram count-based language models, Kuhn et al. (1994) applied hidden Markov models (HMMs) to language modeling and Lau et al. (1993) used entropy-based models for it. These more complex models also allowed for the integration of linguistic features to help language modeling (Berger et al., 1996).

Deep Learning-based Language Models

Deep learning techniques became a powerful solution to overcome n-gram-based systems' limits in language modeling. Early work from Schmidhuber and Heil (1996) showed that character-level deep learning-based language modeling was promising for data compression. Bengio et al. (2001) showed that a 1-hidden layer MLP combined with a dense

representation of words outperforms a state-of-the-art n-gram language model. This work showed for the first time that if trained on enough data with long enough context, even simple deep learning architecture could outperform statistical ones.

Recurrent Neural Networks for Language Modeling In the years that followed, deep learning-based causal language models have been extended to model longer context sizes, larger vocabulary. These approaches led to scaling the number of parameters in the architecture and training these models with more data. Additionally, intense experiments on the type of architecture led to the success of recurrent neural networks for language modeling, introducing the so-called RNNLM. For instance, Mikolov et al. (2011a) showed that recurrent neural networks outperform standard MLP models and are competitive with n-gram state-of-the-art models trained on much more data. Sutskever et al. (2011) integrated gated connections in the RNNLM, speeding up training and reaching state-of-the-art performance for character-level training. With the advance of more powerful hardware and the adaptation of Graphic-Processing Units (GPU) to train deep learning models, it became possible to train more complex architectures such as LSTM models (Hochreiter and Schmidhuber, 1997). The main challenge when training large architecture is overfitting (Sundermeyer et al., 2012). Indeed, large-scale deep-learning models are over-parametrized and are keen to memorize their data which can hurt generalization. By applying carefully regularization techniques such as drop-out (Srivastava, 2013) specifically to the non-recurrent connections, Zaremba et al. (2014) managed to train efficiently large LSTM models leading state-of-the-art performance in causal language modeling. Several modeling refinements were then introduced to improve the performance and make training and inference cheaper and faster (Grave et al., 2017a; Gal and Ghahramani, 2016) as well as to scale the number of parameters to reach better performance (Merity et al., 2018).

Successes of the Transformer Architecture A key factor in the success of deep learning models in language modeling is the number of parameters they can efficiently be trained with and the amount of data they can be trained on. In the case of LSTM models, Gal and Ghahramani (2016) were able to train a 66M parameters model. However, as described in § 4.4.2, recurrent neural network are based on sequential op-

erations. This inherently makes the training process slower compared to feed-forward neural networks. This gave an intrinsic advantage to the Transformer architecture. As we described in §4.4.2, it was introduced by Vaswani et al. (2017) who showed, with an encoder-decoder framework, that it is a powerful architecture for Machine Translation. Radford and Narasimhan (2018) adapted the Transformer to a decoder-only model and trained it with a causal language model objective. Al-Rfou et al. (2019) ran a detailed comparison with LSTM-based language models showing that transformers models are easier to scale and better causal language models compared to LSTM trained with the same amount of parameters and the same training data. This approach was then further extended and scaled with the Transformer-XL model (Dai et al., 2019), GPT-2 (Radford et al., 2019) and GPT-3 (Brown et al., 2020). Similarly, the flexibility of the transformer architecture and its computational efficiency allowed Devlin et al. (2019a) to train a large-scale mask language model with great task-specific success after fine-tuning (cf. § 4.6.3).

4.6 Task-Specific Modeling in NLP

$$\underbrace{p_{\theta_1}(Y|X)}_{training} \rightarrow \underbrace{\hat{Y} \sim p_{\theta_1}(Y|X)}_{prediction} \tag{4.31}$$

Following what we described in the introduction (cf. figure 1.1), NLP models can be split schematically into two parts:

First is a representation module, which takes raw signal sequences of bytes and encodes it into fixed feature vectors. These vectors can be as simple as 1-hot encoding vectors (cf. §4.3.1), more complex hand-crafted morphological features (cf. §4.3.2) or prediction-based continuous word embeddings (cf. §4.3.3).

Second, a trainable model that takes the vectorized representation to perform prediction for a specific task (e.g., POS tagging, Machine Translation, etc.).

Historically, models were based on probabilistic estimation combining complex hand-crafted features (cf. §4.6.1). In the past ten years, deep learning models outperformed these approaches (cf. §4.6.2 and §4.6.3) (Cho et al., 2014; Sutskever et al., 2014; Lample

et al., 2016; Yang et al., 2016; Dozat and Manning, 2016; Peters et al., 2018c; Devlin et al., 2018b) showing that very little features engineering were needed for most NLP tasks.

4.6.1 Hand-Crafted Feature-Based Modeling

Given sequences of tokens $(t_1, .., t_L) \in V^L$ and sequences of labels $(y_1, .., y_L) \in \mathcal{L}^L$, the goal is to estimate $p(y|t_1, .., t_L)$ to perform prediction. If we do sequence classification, we can simply consider y_L as the only label of interest in the sequences of labels. We focus our review on the different POS tagging and NER modeling approaches.

Rule-Based Systems

Early approaches were based on rules. The predictions relied on a combination of hand-crafted features with hand-crafted rules to disambiguate the labels of tokens or sequences of tokens. These rules were defined by hand or learned using computational models on Finite State Machine or Context-Free Grammars (McCulloch and Pitts, 1943; Hopcroft et al., 2006).

Sequence Labeling For POS tagging, Klein and Simmons (1963) were able to reach more than 90% accuracy using a computational grammar coder (CGC). Stolz et al. (1965) extended this approach by adding a probability-based disambiguation step. For ambiguous cases, their approach uses observed probability to disambiguate between plausible POS tags, defined using lexicons and extracted morphological features. For NER, Appelt et al. (1993) developed a non-deterministic Finite State Machine (FSM) [29] to extract named entities and verb groups. In the same spirit, the transformation-based approach (Brill, 1995) starts with rules computed on the (word, tag) observations and morphological lexicon. These rules are then refined iteratively based on the error made by the tagger on some held-out data (Brill, 1992, 1995). Vilain and Day (1996) adapted this approach to NER.

[29] Originally introduced in (McCulloch and Pitts, 1943).

Parsing For syntactic parsing, given a constituency grammar described in § 3.3.2, and POS tags, a wide range of algorithms have been developed in order to parse sentences. For instance, the Cocke-Kasami-Younger (CKY) approach (Kasami, 1966; Aho et al., 1986) is a dynamic-programming algorithm that recursively predicts a parse tree given a constituency context-free grammar. The challenge in grammar-based parsing is that several parse Trees may be generated. To disambiguate between the predicted trees, the first approaches were based on rules using morphological and contextual features to choose the most likely tree.

QA We finish our literature review of rule-based systems for NLP by taking Question-Answering (QA). Early QA systems were based on pipelines composed of several modules. Woods and WA (1977) designed a QA system designed for lunar geologists. It was based on several modules. In the first step, a parser takes the user query and translates it into a database readable query. This step usually involves a lexicon to identify the keywords and a grammar to extract their semantic relationship. Based on it, a formal representation of the user query is generated. Finally, this formal representation is used to query a database. Similarly, Green et al. (1961) built a QA system for baseball. Notably, Lehnert (1977) conceptualized Question Answering, starting from the human thought process. This led to the making of the QUALM rule-based QA system.

Based on the progress in Information-Retrieval (IR) (Kupiec, 1993) and low-level tasks such as NER, coreference resolution, and stemming, Question Answering systems became more and more complex. For instance, (Hirschman et al., 1999) is one of the pioneering works building a QA system that can predict short-answer spans. It did so by combining refined preprocessing approaches such as rule-based stemming (Abney, 1997) to reduce the sparsity on the query side and document side, as well as named entity recognition and coreference resolution on the answer side to enrich the query and document representation.

Feature-Based Probabilistic Modeling

Based on these early rule-based systems, the empirical progress and modeling novelties came from accessing larger annotated datasets, computing more informative features, and more accurate statistical estimation techniques.

For instance, Bahl and Mercer (1976) approached POS tagging with a statistical decision algorithm. Schmidt (1994) used a trainable decision tree.

Sequence Labeling & Parsing Ratnaparkhi (1996) proposed to use the maximum-entropy principle (Jaynes, 1957; Berger et al., 1996) to train a feature-based model for POS tagging. In short, words are characterized by a list of linguistic features based on their morphology (prefix, suffix, etc.), POS tags of words seen in the training data, and contextual features (i.e., linguistic features and POS tags of surrounding words). Based on this, the model is parametrized with a simple exponential multinomial model. Following the maximum entropy principle (Jaynes, 1957),[30] the model is then estimated by finding the parameters that maximize the entropy of the model. In such a model, the challenges are the sparsity of the features. Indeed, at test time, if the words are not in the training data, the model only relies on hand-crafted features. These features are usually language-specific, not robust to non-standard wordforms (e.g., spelling errors), and sparse. Bender et al. (2003) successfully used the same approach for NER. We also note that maximum entropy-based models were widely used for the CoNLL-2003 shared task for NER (Daelemans and Osborne, 2003), and were a key module of the winning system (Florian et al., 2003).

For syntactic parsing, methods were developed to estimate the probability of a given parse tree provided by a grammar-based algorithm. By factorizing the probability of a tree by using the head-dependent structure, and estimating the probability of head-dependent patterns using treebanks, (Charniak, 1997; Collins, 1997) were able to reach very good performance. These approaches were then refined and extended using better-suited hand-crafted features and better probabilistic models (Johnson, 2001; Petrov and Klein, 2007; Finkel et al., 2008).

[30] The intuition is that given a parametrization and some observation, the best parameters are the ones that distribute the most evenly the probability mass between the different labels, i.e., the parameters that lead to the maximum entropy.

In the 90s, Hidden Markov Models (HMM) (Baum and Petrie, 1966) brought significant performance improvement to NLP and sequence labeling. In a nutshell, HMM aims at estimating transition probabilities of a hidden (i.e., unobserved) random process y — for POS tagging or NER, a sequence of tags $y_1, .., y_L$ — given an observed sequence $x_1, .., x_L$. A transition probability is simply the probability of getting y_t given previous states $y_{1:t-1}$ and the observed variable x_t. We note that HMM are a specific case of Graphical Generative Probabilistic Models.[31] Charniak et al. (1993) showed that HMM could reach very competitive performance on POS tagging.[32] Based on standard Markov assumptions,[33] Merialdo (1994) reached similar conclusions. On this line of research, Toutanova et al. (2003) reached state-of-the-art performance (above 97% on the Penn Tree Bank (Marcus et al., 1993)) using dependency networks (Heckerman et al., 2001) — a generalization of HMM to a cyclic graphical probabilistic model — based on rich morphological and contextual hand-crafted features. HMMs have also been extensively studied for NER. Zhou and Su (2002) reached state-of-the-art performance with a feature-based HMM model. HMMs were also widely used for the ConLL-2003 shared task for NER (Daelemans and Osborne, 2003). For NER, we note that the feature selection usually differs from POS tagging. To identify Named Entities, the morphological features used are usually based on the type of entities we are aiming to extract. For instance, to identify dates, Zhou and Su (2002) relies on date template matching features based on the list of days of the week and month. Identifying location and person relies on knowledge features (referred to as *Gazetteer* features) such as the list of locations and well-known people (e.g. "Bill Gates").

Finally, following the same feature-based approach, early works on deep learning models reached non-trivial performance on POS tagging (Benello et al., 1989; Nakamura and Shikano, 1988). In these studies, the deep learning models are usually simple feed-forward neural networks (§ 4.4) modeling 1-hot encoded features of each word as inputs.

[31] They are Directed Acyclic Probabilistic Graphical Model that characterizes the dependencies between two random processes with two simplifying assumptions: $p(x_t|y_{1:t}, x_{1:t-1}) = p(x_t|y_{1:t})$ and $p(y_t|y_{1:t-1}) = p(y_t|y_{t-1})$.

[32] Above 96% on the Brown corpus (Francis and Kucera, 1979).

[33] For sequence labeling, the Markov assumption can be framed as $p(y_t|y_{1:t-1}) = p(y_t|y_{t-1})$.

QA Based on the progress of search engines and the beginning of the mainstream internet at the end of the 90s, the Question Answering task became a critical piece for search engines to provide specific sentences, spans, and entities to the end-user. The task was formalized with the TREC Question Answering track and benchmark (Voorhees and Tice, 2000). (Ng et al., 2000) is one of the first machine learning models to approach the Reading Comprehension task (Hirschman et al., 1999). It models the task with multiple semantic features based on pattern matching between question and candidate answers, with keywords, coreference resolution, and named entity recognition. These features are fed to a decision tree based on (Quinlan, 1993)'s work [34]. (Echihabi and Marcu, 2003) adapted the noisy channel approach for Question Answering by parsing syntactically and semantically queries and answers to identify the correct spans. (Wang et al., 2007) approached Question Answering as an Answer Selection Task (AS2). Answers and queries are structured using POS tags and dependency parse trees. The model is then trained with maximum-log-likely-hood. Wang et al. (2007) outperformed state-of-the-art baselines with this approach.

Limits We note that in this paradigm, most of the progress done in NLP — approximately between the 1950s and the 2000s — usually struggled to be adapted for other types of tasks. Indeed, nearly all the models were based on a high level of expertise for each task to define, select and refine the features necessary to solve the task. Notably, most of this work was language-specific, task-specific, and domain-specific and, therefore, generalized poorly.

4.6.2 Distributional Representation as Input of Deep-Learning Architectures

With the progress of modern hardware and the increase of FLOP/s,[35] in the mid-2000s it became feasible to train larger deep learning architectures on larger quantity of textual data (Bengio et al., 2001; Collobert and Weston, 2007, 2008). Additionally, at the beginning

[34] A notable related work of decision tree models is the Classification and Regression Trees approach from Breiman et al. (1983)

[35] FLOP/s stands for Floating-point operation per second and is a standard metric to measure the computing power across different hardware.

of the 2010s, prediction-based distributional continuous word representation techniques (Mikolov et al., 2013b; Pennington et al., 2014) provided word vectors that capture refined morphological (Mikolov et al., 2018), syntactic, and, semantic information. Based on this fundamental progress, more available resources (FLOP/s and Datasets) on the one hand and better representation learning techniques on the other, deep learning models brought significant empirical progress to NLP.

Collobert and Weston (2008) trained a fixed-window neural network on multiple tasks such as POS tagging, NER, and Semantic Role Labeling. They used a multi-layer feed-forward neural network composed of a trainable — randomly initialized — word embedding layer (§ 4.4.2), a fixed window size for the context (of around five words) modeled with a convolution layer and multiple feed-forward neural network[36]. A single architecture is trained on multiple tasks (including a language modeling task). Collobert and Weston (2008) reached near state-of-the-art performance for all the tasks studied.

Collobert et al. (2011) improved this approach by using a pretrained word embedding layer. This layer was trained using a fixed-window MLP similar to (Bengio et al., 2001) model.

After scaling the number of parameters, training the models on a larger quantity of data, improving the generalization of models with various training regularization techniques (§4.4.3), recurrent neural networks and, more specifically, LSTM-based models outperformed significantly other modeling approaches in sequence labeling and structured prediction. Lample et al. (2016) combined a bidirectional LSTM with a CRF using a pretrained word2vec-based input layer to reach state-of-the-art performance in NER. We note that the most critical piece in this model is the pretrained word2vec layer, bringing more than 5 points of improvement compared to comparable baseline models that do not use a pretrained word2vec layer. Similarly, Wang et al. (2015) outperformed all non-deep learning-based systems for POS tagging. Plank et al. (2016) extended and improved this approach by introducing character-level Bi-LSTM plugged into a word-level Bi-LSTM and reached state-of-the-art for large number of languages[37]. In Dependency Parsing, Kiperwasser and Goldberg (2016) successfully adapted a Bi-LSTM model to

[36] If we exclude the word embedding layer, the number of trainable parameters is of the order of magnitude of 100,000.

[37] They used polyglot word embedding vectors (Al-Rfou' et al., 2013) trained based on a denoising language model objective.

both transition-based and graph-based parsing. Each input wordform is represented in both cases with a POS embedding input vector (randomly initialized) and a word embedding vector (possibly pretrained and fine-tuned during training). Dyer et al. (2016) jointly model sequences of words and syntactic trees using an LSTM-based model. Finally, Dozat and Manning (2016) refined (Kiperwasser and Goldberg, 2016)'s approach by adding a Bi-Affine output layer to perform graph-based dependency parsing. This approach established new state-of-the-art performance on dependency parsing for 50+ languages (Dozat et al., 2017; Qi et al., 2018).

Similarly, combining LSTM-based deep learning models with word embeddings improved the performance of QA systems. Working with large datasets such as the SQUAD dataset from (Rajpurkar et al., 2016) and the CNN/Daily mail from (Hermann et al., 2015), Seo et al. (2016) introduced the Bidirectional Attention-Flow to perform span-prediction and reached state-of-the-art. Again, a key element in this model is the use of pretrained prediction-based word embeddings. Other related work in span-based QA use the same modeling ingredients (Hermann et al., 2015; Chen et al., 2016; Hu et al., 2017; Clark and Gardner, 2018).

Given this significant progress made using a single type of model — a pretrained word embedding layer combined with an LSTM-based deep learning model — across a wide diversity of tasks, it became even more crucial to build NLP benchmarks (McCann et al., 2018; Wang et al., 2018, 2019a). Simply stated, a benchmark is a group of tasks, training, and evaluation dataset allowing practitioners to compare the performance of modeling techniques on a wide variety of tasks. GLUE introduced by Wang et al. (2018) is one of the most notable recent benchmarks. It combines nine sentence understanding tasks such as Natural Language Inference (Dagan et al., 2005b; Bowman et al., 2015b), sentiment analysis, or grammatical acceptability task. Wang et al. (2018) showed that a BI-LSTM model combined with word embeddings was a very competitive approach for the GLUE benchmark.

Limits A key ingredient in this modeling approach is the transfer occurring between continuous vectors, pretrained with prediction-based word embedding methods (§ 4.3.3) and randomly initialized deep-learning architectures. An inherent constraint in this

approach is that the pretrained representations of words are static. Indeed, each input tokens get a fixed vector.

4.6.3 Language Model as Transferable Contextual Representation for Specific Tasks

$$\underbrace{p_{\theta_0}(X)}_{pretraining} \rightarrow \underbrace{p_{\theta_1}(Y|X,\theta_0)}_{fine-tuning} \rightarrow \underbrace{\hat{Y} \sim p_{\theta_1}(Y|X)}_{prediction} \tag{4.32}$$

There are many ways to transfer a pretrained deep learning-based language model for downstream tasks. One of the first attempts is Collobert and Weston (2008), which trained a single MLP architecture simultaneously with a denoising language model objective (§4.5.1) along with downstream tasks such as Semantic Role Labeling, POS tagging, and NER. However, language modeling did not improve the performance of the model.

More recently, Howard and Ruder (2018) improved the state-of-the-art performance on multiple text classification tasks by pretraining an LSTM-based language model and fine-tuning the entire architecture — after appending a task-specific feed-forward module — on the classification task. Similarly, Peters et al. (2018b) released the ELMo model, which provides contextual word embedding as the output of a Bidirectional-LSTM pretrained language model. The difference is that at fine-tuning time, only weights over the LSTM language model layer are learned — as opposed to fine-tuning the entire architecture. Finally, Devlin et al. (2019b) introduced the revolutionary BERT model. By framing the language model task as Masked Language Modeling (MLM § 4.5.1), by using large pretraining a transformer encoder with it on a large quantity of data (about 3.2 Billion words), and by fine-tuning the entire architecture on downstream sequence labeling tasks, Devlin et al. (2019b) improved from a substantial margin the state-of-the-art performance on a wide variety of tasks. By sharing it through the popular transformers library (Wolf et al., 2020),[38] BERT impacted the entire field of Natural Language processing by providing an easy-to-use framework for nearly any NLP task, extending the state-of-the-art performance in most of them.

[38] It was first released as `pytorch-pretrained-bert` https://github.com/huggingface/transformers/releases/tag/v0.1.2.

Following the success of BERT, many variants were released, changing one or several of its core design parameters, and improving the downstream performance of BERT. Liu et al. (2019) showed with the RoBERTa model that the next-sentence prediction of BERT is not useful for downstream performance and removed it from the training objective. They also implemented dynamic masking, refined the optimization process by training on larger batch sizes (8,000 sequences of length 512 tokens) and for a much larger number of steps (1M steps), and trained on more data (160GB). Joshi et al. (2020a) used whole-word masking and mask spans of several words instead of single bpe tokens and trained for long sequences in place of pairs of sentences. Zhang et al. (2019) integrated an entity masking and prediction of related entities' objectives. Lan et al. (2020) introduced shared layers. Yang et al. (2019) replaced the mask language model objective with an autoregressive objective with permutation to keep bidirectional context. Finally, One of the most notable contributions compared to BERT is the ELECTRA model (Clark et al., 2020b) which introduces a more sample-efficient method by replacing the mask language modeling task with a discrimination task (between gold tokens and plausible tokens generated by another MLM Transformer based model). All these approaches reduced the cost of pretraining (data, parameter, or sample efficiency) or downstream task performance. By adapting the mask-language denoising objective to encoder-decoder architectures (§ 4.4.2), (Raffel et al., 2019; Lewis et al., 2020) showed that pretraining large transformer could also lead to impressive improvement for sequence generation tasks.

Finally, a wide number of models were released for non-English languages. In chapter 5, we present our contribution with the CamemBERT model. We list — non-exhaustively — multilingual mask language models such as mBERT, XLM (Conneau and Lample, 2019), XLM-R (Conneau et al., 2020a), mT5 (Xue et al., 2021), mBART (Liu et al., 2020) and monolingual models in Arabic (Antoun et al., 2020), Dutch BERT (de Vries et al., 2019), Chinese (Cui et al., 2021, 2020), or Spanish (Cañete et al., 2020).

As of March 2022, more than 5000 unique model checkpoints have been uploaded to the Hugging-Face hub based on denoising language modeling.[39]

[39] cf. https://huggingface.co/models?pipeline_tag=fill-mask&sort=downloads.

4.6.4 Prompting Language Models for Zero-Shot and Few-Shot Generalization

$$\underbrace{p_{\theta_0}(X)}_{pretraining} \rightarrow \underbrace{\hat{X} \sim p_{\theta_1}(X|prompt)}_{prediction} \tag{4.33}$$

By scaling the number of parameters of generative pretrained language models (Andonian et al., 2021; Brown et al., 2020; Rae et al., 2021; Smith et al., 2022; Chowdhery et al., 2022), new learning behavior have been observed. Most interestingly, (Radford et al., 2019; Brown et al., 2020) showed that language models could be used to perform specific NLP tasks such as sequence classification or sequence generation by prompting them with natural language text without the need for extra parameter updates through fine-tuning.

In the zero-shot setting — i.e., trained without any annotated data in the given task —, (Brown et al., 2020)'s generative model was shown to compete with accurate NLP systems trained on large amounts of annotated data (e.g., SuperGLUE (Wang et al., 2019a)). After prompting the model with a few demonstrations (i.e., training samples framed with natural language), the 540 Billion parameters PALM model (Chowdhery et al., 2022) even outperforms average human performance[40] for some tasks (e.g., on the Big Bench benchmark[41]).

However, the way natural language prompts (also referred to as hard or discrete prompts) are framed, specifically, the task demonstration prompts, highly impacts the performance of the model (Jiang et al., 2020; Liu et al., 2021). This challenge led to designing techniques to search for the most optimal hard prompts (Reynolds and McDonell, 2021) and to collecting large-scale datasets of natural prompts (Sanh et al., 2021b; Bach et al., 2022).

Concurrently, soft-prompting was introduced (Lester et al., 2021; Qin and Eisner, 2021) as a middle ground between fine-tuning large-scale generative language models and prompting them. Soft-prompting consists in inserting new parameters into the model that are fine-tuned on a specific task. (Li and Liang, 2021) further extended it for generation tasks.

[40] Average human performance reflects the score of non-experts human in performing a given task averaged across all generated answers.

[41] https://github.com/google/BIG-bench

The generalization abilities of language models with prompting emerged from models trained with billion or hundred of billion of parameters. However, these large-scale language models are inherently costly to use and train, posing many ethical and environmental challenges that should be addressed as discussed in (Bender et al., 2021; Weidinger et al., 2021b).

4.7 Modeling Out-of-Domain and Out-of-Language Text

So far, following a standard Machine Learning framework, we assumed that the evaluation data originates from the same domain and language as the data the model is trained on. However, as described in chapter 2, languages are very diverse and highly vary across speakers, writers, genres, topics, mediums, essentially across domains (§2). Additionally, collecting annotated data for all tasks of interest across all languages and domains is costly and challenging. For these reasons, it is often necessary for NLP to use a model trained on a dataset that originates from a given language or domain referred to as the source, on another language or domain referred to as the target. In the case of domains, we refer to this process as cross-domain transfer. In the case of languages, we refer to it as cross-lingual transfer (§ 4.7.2).

This section covers techniques used to achieve cross-domain and cross-lingual transfer.

4.7.1 Domain and Language Gap Reduction

$$\widetilde{X} \rightarrow X \tag{4.34}$$

To make cross-lingual or cross-domain transfer easier, it can be useful to make the target domain or language, noted $\widetilde{X}$, more similar to the source language or domain X (illustrated in figure 4.34). There are many ways to achieve this depending on the experimental framework and the linguistic properties we would like to make "more similar". For non-standard text such as User Generated Content, lexical normalization can be used to make the spelling errors and jargon specific to this data type more similar to standard edited text such as Wikipedia. To build models in the multilingual setting,

translation and transliteration can be used to get training data in the target language and script or to use models trained in another language and script for the target language.

Lexical Normalization for non-standard data

As described in section 3.3.4, lexical normalization consists in transforming a non-standard word into its standard form. In this thesis, we will focus mainly on User-Generated-Content (UGC), i.e., data found on social media data like Twitter. As described in (Foster, 2010b; Seddah et al., 2012; Eisenstein, 2013b; Baldwin et al., 2013), UGC is often characterized by the extensive use of abbreviations, slang, internet jargon, emojis, embedded metadata (such as hashtags, URLs or *at* mentions), and non-standard syntactic constructions and spelling errors. For this reason, lexical normalization has traditionally been seen as a promising way to cope with non-standard UGC data.

The Lexical Normalization task is close to Grammar Error Correction and Spelling correction. Research on these topics was originally on spelling error identification. These approaches were based on lists of correct spelling forms such as (McIlroy, 1982). By integrating string comparison to it (e.g., using the edit distance), Henry (1986) ranks which correction is the most suited given an identified spelling error. Traditionally, lexical normalization has been tackled probabilistically using Shannon's noisy channel approach (Shannon, 1948). In essence, this method decomposes the probability of getting a "standard" form given a noisy form by factorizing it between a language model on the standard data and an error model (between the non-standard data and the standard one). Brill and Moore (2000) successfully applied this method to spelling correction by splitting strings into substrings. Toutanova and Moore (2002) extended it by integrating phonological features. Choudhury et al. (2007) used similar features in an HMM model to achieve better performance in spelling correction. (Han and Baldwin, 2011) is one of the pioneering studies to extend lexical normalization to Social Media data (Twitter data). Lexical normalization is seen in two modeling steps. The first step is a non-standard word identification step. The second is a normalization step. In terms of modeling, they used syntactic features (extracted using a dependency parser) with an SVM model (Boser et al., 1992) to detect non-standard forms. Then, based on string-based and phonemic distances and language modeling, they predict the normalized form based on a list of

candidates and assume. This work illustrates the complexity of tackling non-standard textual data with feature-based systems. Indeed, non-standard textual data requires a wide range of features (such as syntactic and phonological features) and rich contextual information to be normalized. Li and Liu (2015) used a maximum entropy model to rank standard wordforms using multiple morphological and contextual features.

(van der Goot and van Noord, 2017; van der Goot, 2019) significantly improved the state-of-the-art performance with the MoNoise model that combines several modules for generating standard candidates and ranking them. The candidate generator relies on several sub-modules such as a rule-based spell-checker,[42] the similarity in a word2vec pretrained embedding space (Mikolov et al., 2013b), look-up tables computed from the training data, and morphological features. Based on the candidate list, a Random Forest classifier (Breiman, 2001) ranks a feature-based representation of the candidates.[43] In English, van der Goot and van Noord (2017) showcased the performance of MoNoise on Lexical Normalization of Twitter datasets, namely LexNorm (Han and Baldwin, 2011), LexNorm1.2 (Eisenstein, 2013a), several normalization datasets aggregated by Li and Liu (2014), and the LexNorm2015 dataset (Baldwin et al., 2015) of newly collected and normalized tweets.

Overall, these feature-rich techniques are very promising but are challenging to scale to more languages than end-to-end models. In the chapter 6, we will cover our contribution to this task and also discuss the most recent state-of-the-art in multilingual lexical normalization.

Transliteration

As described in section 3.3.5, transliteration consists of converting data written in one script to another. For many NLP tasks such as machine translation, transliteration can be used as an intermediary step to reduce the gap between the source data and the target data (Nakov and Tiedemann, 2012; Birch et al., 2013; Durrani et al., 2014a).

Transliteration has traditionally been approached using rule-based systems that rely on phonology. In a few words, the transliteration system uses how a sequence of characters

[42] They use aspell `http://aspell.net/`.

[43] The features are similar to the candidate generation step, to the exception of an n-gram language model trained on both standard and non-standard text.

is pronounced to convert it into the target script by ensuring that the transliteration preserves the pronunciation.

Grapheme-to-phoneme (G2P) is a specific type of transliteration for which the output sequence encodes directly the phonology of the input text (3.3.5). The IPA (Jespersen, 2013) is commonly used for this purpose. G2P has been approached by collecting heuristics that associate characters of the source scripts (e.g., Latin, Cyrillic, Arabic, Japanese) to the IPA alphabet (Daelemans and van den Bosch, 1996; Black et al., 1998). It can also be modeled in a supervised way using sequential models (Rao et al., 2015). We note that for most languages, building a mapping table between graphemes and phonemes leads to excellent performance (Mortensen et al., 2018). Romanization refers to transliteration with Latin as the output script. Similarly to IPA-transliteration, it can be approached using heuristics. For instance, Hermjakob et al. (2018) use Unicode character description corrected with hand-crafted rules to get phonetic representations. These representations are then transliterated to the Latin script.

For some pairs of scripts, transliteration may be more ambiguous. For instance, transliterating Urdu from the Perso-Arabic script and the Devanagari script requires many steps of normalization and conversion. (Lehal and Saini, 2012; Ray et al., 2022) approached it with multi-step pipelines that normalize, segment, and use language models to address ambiguous cases.

We note that transliteration from the Arabic script is usually ambiguous, mainly because of its consonantal nature (§2.3.2). Indeed, vowels are encoded with diacritics which are usually not written, so the transliteration model has to implicitly guess the diacritics based on the context to predict the character in the target script.

Arabic written in the Latin script is usually referred to as *Arabizi*. It is used online by millions of people worldwide who do not have access to an efficient Arabic keyboard or prefer to use Latin keyboards. Transliterating text from Arabizi to its Arabic script form is challenging. Mainly because Arabizi is non-standard and is usually code-mixed with other languages such as French or English (Seddah et al., 2020). Younes et al. (2018); Shazal et al. (2020) addressed it with sequence to sequence RNN-based models.

Translation

Finally, a straightforward way to do cross-lingual transfer is to use machine translation. Indeed, assuming we want to perform a sequence classification task (e.g., NLI) in a target language $\hat{X}$ (e.g., English), and we have annotated data in a language X (e.g., French), we can train a model in English, translate the data from French to English and perform the prediction in English itself — known as the Translate-Test setting (Conneau et al., 2018b; Hu et al., 2020). We can also translate the source language to the target to train a model directly in the target language (we refer to it as Translate-Train). We note that this cannot be used for sequence labeling tasks (as it would require token-level alignment between source and target).

Translation can also be used for data augmentation techniques for word-level tasks such as POS tagging and dependency parsing. For instance, Agic et al. (2016) developed an annotation projection technique using parallel biblical text (Christodoulopoulos and Steedman, 2015) and word aligners to build dependency parsers for low-resource languages.

4.7.2 Representation Transfer

$$\underbrace{p_\theta(Y|X)}_{training} \xrightarrow{\text{zero-shot cross-lingual transfer}} \underbrace{p_\theta(\widetilde{Y}|\widetilde{X})}_{evaluation} \tag{4.35}$$

Cross-lingual transfer is a general term that refers to the process of using a model trained on a language or a set of languages for another language possibly unseen during training. Zero-shot Cross-Lingual Transfer, illustrated in figure 4.35, refers to the process of training a model (parametrized by θ) in one source language noted X (e.g. English) to use it for another *target* language noted $\widetilde{X}$ (e.g. French).

The basic idea of performing cross-lingual transfer is to build a multilingual representation of data shared between the source and target languages. They are many ways to build shared multilingual representations. We list three approaches, from the least to the most successful empirically.

Feature-Based Multilingual Representation

Historically, cross-lingual transfer modeling has been approached with lexical features (§ 4.3.2). The idea is to represent words with morphological, syntactic, or semantic features shared across languages. Then, we can train them using only the lexical features instead of training task-specific models on the words. This method is referred to as the delexicalized approach and has been studied, among others, by (Petrov and Klein, 2007; Zeman and Resnik, 2008; Søgaard, 2011).

Static Multilingual Word Embedding

Based on the success of prediction-based word embedding techniques like the word2vec model (cf. § 4.3.3), many studies designed alignment techniques to build multilingual word embedding shared between several languages. These techniques are based on learning a projection function that maps one embedding space onto another. This projection is usually trained using a word-level translation dictionary (Mikolov et al., 2013a). Smith et al. (2017) showed that for related languages like English, German and French, it was possible to learn accurate rotations between languages by relying on anchor words only – i.e., words that both exist in the source language and the target language. Artetxe et al. (2018) refined this approach by adding normalization and a re-weighting step into the projection process leading to state-of-the-art performance in zero-shot bilingual lexicon extraction (Ruder et al., 2019).

Based on these shared word-embedding spaces, similarly to the delexicalized approach, the idea is to train a model using input word embedding from the shared multilingual space in the source language and, at test time to feed the model with input tokens in the target language represented with word embedding from the same multilingual embedding space (Klementiev et al., 2012; Guo et al., 2015).

Multilingual Language Models for Zero-Shot Cross-Lingual Transfer

Based on the success of the pretraining-fine-tuning (§ 4.6.3) paradigm based on large scale language models, and on the release of large scale multilingual language models (Devlin

et al., 2018b; Conneau and Lample, 2019; Conneau et al., 2020a; Xue et al., 2021; Liu et al., 2020).

Pires et al. (2019a) was the first work to analyze the performance of the multilingual version of BERT in the zero-shot cross-lingual transfer setting. They showed that mBERT could perform well in languages mBERT was trained on, even between languages written in different scripts. The more typologically similar, the better the zero-shot cross-lingual transfer is. Multilingual language model-based zero-shot cross-lingual transfer was shown to work across a wide variety of languages and for a wide variety of tasks (Conneau and Lample, 2019; Conneau et al., 2020a; Xue et al., 2021; Liu et al., 2020; Clark et al., 2020a). Even though these approaches exhibit interesting cross-lingual transfer abilities, the reasons of this transfer is largely unexplained. In chapter 7, we present structural and behavioral analyses of mBERT to help understand how multilingual language models perform cross-lingual transfer.

Part II

Pretraining and Fine-tuning a Transformer Masked Language Model

5 Language Modeling For French: More Data Is Not Always Needed

5.1 Motivations

As we described in the previous chapters, language modeling has become a fundamental component in the making of state-of-the-art task-specific models in NLP (Peters et al., 2018b; Devlin et al., 2019b) . More specifically, large-scale transformer (Vaswani et al., 2017) Masked Language Models trained on large raw text corpora extended impressively the performance of NLP models on most tasks.

However, for almost a year and a half after the first release of the BERT model, large-scale mask monolingual language models were only available for English.

This chapter is an adaptation of the publications (Martin et al., 2019; Martin et al., 2020) which was a close collaboration with Louis Martin and Pedro Ortiz. As part of this work, I was personally in charge of the fine-tuning experiments for NER, POS tagging and Dependency parsing. I also contributed with research ideas, specifically the analysis of large-scale models trained on smaller amount of data, and trained on fewer number of training steps.

We present our work on pretraining, fine-tuning, and analyzing the CamemBERT model, a large BERT-like model for French. In summary, we find that if trained on diverse enough data, it is possible to pretrain a state-of-the-art mask language model with as little as 4GB of data.

5.2 Modeling

5.2.1 Transformer

Similar to RoBERTa and BERT, CamemBERT is a multi-layer Transformer encoder (Vaswani et al., 2017). We point the reader to §4.4.2 for a detailed definition of the Transformer. CamemBERT uses the original architectures of $\text{BERT}_{\text{BASE}}$ (12 layers, 768 hidden dimensions, 12 attention heads, 110M parameters) and $\text{BERT}_{\text{LARGE}}$ (24 layers, 1024 hidden dimensions, 16 attention heads, 335M parameters). CamemBERT is derived from RoBERTa (Liu et al., 2019) (presented in §4.6.3), the main differences being the use of whole-word masking and the use of SentencePiece (Kudo and Richardson, 2018a) tokenization (cf. § 4.2.5).

5.2.2 Training on the OSCAR Corpus

Pretrained language models benefits from being trained on large datasets (Devlin et al., 2018b; Liu et al., 2019; Raffel et al., 2019). We therefore use the French part of the OSCAR corpus (Ortiz Suárez et al., 2019; Ortiz Suarez, 2022), a pre-filtered and pre-classified version of Common Crawl[1].

OSCAR is a set of monolingual corpora extracted from Common Crawl snapshots. It follows the same approach as (Grave et al., 2018) by using a language classification model based on the fastText linear classifier (Grave et al., 2017b; Joulin et al., 2016) pretrained on Wikipedia, Tatoeba and SETimes, which supports 176 languages. No other filtering is done. We use a non-shuffled version of the French data, which amounts to 138GB of raw text and 32.7B tokens after subword tokenization. We use the OSCAR 2019 version. [2]

5.2.3 Pre-processing

We segment the input text data into subword units using SentencePiece with Byte-Pair encoding (Kudo and Richardson, 2018a). SentencePiece is an extension of Byte-Pair encoding (BPE) from Sennrich et al. (2016b) and WordPiece (Kudo, 2018) that does not require pre-tokenization (at the word or token level), thus removing the need for

[1] https://commoncrawl.org/about/

[2] Available at https://oscar-corpus.com/post/oscar-2019/.

language-specific tokenisers. We point the reader to §4.2.5 for a detailed introduction and discussion on subword-level tokenization technique. We set the vocabulary size to 32k subword tokens. These subwords are learned on 10^7 sentences sampled randomly from the pretraining dataset. We do not use subword regularisation (i.e. sampling from multiple possible segmentations) for the sake of simplicity.

5.2.4 Pretraining Objective

We train our model on the Masked Language Modeling (MLM) task. Given an input text sequence composed of N tokens $(x_1, ..., x_N)$, we select 15% of tokens for possible replacement. Among those selected tokens, 80% are replaced with the special <MASK> token, 10% are left unchanged and 10% are replaced by a random token. The model is then trained to predict the initial masked tokens using cross-entropy loss. We point the reader to § 4.5.1 for a more detailed definition of MLM.

Following the RoBERTa approach, we dynamically mask tokens instead of fixing them statically for the whole dataset during preprocessing. This improves variability and makes the model more robust when training for multiple epochs.

Since we use SentencePiece to tokenize our corpus, the input tokens to the model are a mix of whole words and subwords. An upgraded version of BERT[3] and Joshi et al. (2019) have shown that masking whole words instead of individual subwords leads to improved performance. Whole-word Masking (WWM) makes the training task more difficult because the model has to predict a whole word — possibly made of a several tokens — rather than predicting only part of the word given the rest. We train our models using WWM by using whitespaces in the initial untokenized text as word delimiters.

WWM is implemented by first randomly sampling 15% of the words in the sequence and then considering all subword tokens in each of this 15% for candidate replacement. This amounts to a proportion of selected tokens that is close to the original 15%. These tokens are then either replaced by <MASK> tokens (80%), left unchanged (10%) or replaced by a random token.

[3] https://github.com/google-research/bert/blob/master/README.md

Subsequent work has shown that the next sentence prediction (NSP) task originally used in BERT does not improve downstream task performance (Lample and Conneau, 2019; Liu et al., 2019), thus we also remove it.

5.2.5 Optimisation

Following (Liu et al., 2019), we optimize the model using Adam (Kingma and Ba, 2015) ($\beta_1 = 0.9$, $\beta_2 = 0.98$) for 100k steps with large batch sizes of 8192 sequences, each sequence containing at most 512 tokens. We enforce each sequence to only contain complete paragraphs (which correspond to lines in the our pretraining dataset).

5.2.6 Pretraining

We use the RoBERTa implementation in the fairseq library (Ott et al., 2019). Our learning rate is warmed up for 10k steps up to a peak value of 0.0007 instead of the original 0.0001 given our large batch size, and then fades to zero with polynomial decay. Unless otherwise specified, our models use the BASE architecture, and are pretrained for 100k backpropagation steps on 256 Nvidia V100 GPUs (32GB each) for a day. We did not train our models for longer due to practical considerations, even though the performance was still increasing (cf. Figure 5.1).

5.3 Using CamemBERT for downstream tasks

We use the pretrained CamemBERT in two ways. In the first one, which we refer to as *fine-tuning*, we fine-tune the model on a specific task in an end-to-end manner. In the second one, referred to as *feature-based embeddings* or simply *embeddings*, we extract frozen contextual embedding vectors from CamemBERT. These two complementary approaches shed light on the quality of the pretrained hidden representations captured by CamemBERT.

5.3.1 Fine-tuning

For each task, we append the relevant predictive layer on top of CamemBERT's architecture. Following the work done on BERT (Devlin et al., 2019a), for sequence tagging and sequence labeling we append a linear layer that respectively takes as input the last hidden representation of the <s> special token and the last hidden representation of the first subword token of each word. For dependency parsing, we plug a bi-affine graph predictor head as introduced by Dozat and Manning (2017). We refer the reader to this article for more details on this module. We fine-tune on XNLI by adding a classification head composed of one hidden layer with a non-linearity and one linear projection layer, with input dropout for both.

We fine-tune CamemBERT independently for each task and each dataset. We optimize the model using the Adam optimiser (Kingma and Ba, 2015) with a fixed learning rate. We run a grid search on a combination of learning rates and batch sizes. We select the best model on the validation set out of the 30 first epochs. For NLI we use the default hyper-parameters provided by the authors of RoBERTa on the MNLI task.[4] Although this might have pushed the performances even further, we do not apply any regularisation techniques such as weight decay, learning rate warm-up or discriminative fine-tuning, except for NLI. We show that fine-tuning CamemBERT in a straightforward manner leads to state-of-the-art results on all tasks and outperforms the existing BERT-based models in all cases. The POS tagging, dependency parsing, and NER experiments are run using Hugging Face's Transformer library extended to support CamemBERT and dependency parsing (Wolf et al., 2019). The NLI experiments use the fairseq library following the RoBERTa implementation.

5.3.2 Embeddings

Following Straková et al. (2019) and Straka et al. (2019) for mBERT and the English BERT, we use CamemBERT in a feature-based embeddings setting. In order to obtain a representation for a given token, we first compute the average of each sub-word's

[4] More details at https://github.com/pytorch/fairseq/blob/master/examples/roberta/README.glue.md.

representations in the last four layers of the Transformer, and then average the resulting sub-word vectors.

We evaluate CamemBERT in the embeddings setting for POS tagging, dependency parsing and NER; using the open-source implementations of Straka et al. (2019) and Straková et al. (2019).[5]

5.4 Evaluation of CamemBERT

In this section, we measure the performance of our models by evaluating them on the four aforementioned tasks: POS tagging, dependency parsing, NER and NLI.

5.4.1 POS tagging and dependency parsing

Model	GSD		Sequoia		Spoken		ParTUT	
	UPOS	LAS	UPOS	LAS	UPOS	LAS	UPOS	LAS
mBERT (fine-tuned)	97.48	89.73	98.41	91.24	96.02	78.63	97.35	91.37
$XLM_{MLM-TLM}$ (fine-tuned)	98.13	90.03	98.51	91.62	96.18	80.89	97.39	89.43
UDify (Kondratyuk, 2019)	97.83	91.45	97.89	90.05	96.23	80.01	96.12	88.06
UDPipe (Straka, 2018)	97.63	88.06	98.79	90.73	95.91	77.53	96.93	89.63
+ mBERT + Flair	97.98	90.31	**99.32**	93.81	**97.23**	81.40	97.64	92.47
CamemBERT (fine-tuned)	**98.18**	**92.57**	99.29	**94.20**	96.99	81.37	**97.65**	**93.43**
UDPipe + CamemBERT (embeddings)	97.96	90.57	99.25	93.89	97.09	**81.81**	97.50	92.32

Table 5.1: **POS** and **dependency parsing** scores on 4 French treebanks, reported on test sets assuming gold tokenization and segmentation (best model selected on validation out of 4). Best scores in bold, second best underlined.

For POS tagging and dependency parsing, we compare CamemBERT with other models in the two settings: *fine-tuning* and as *feature-based embeddings*. We report the results in Table 5.1.

CamemBERT reaches state-of-the-art scores on all treebanks and metrics in both scenarios. The two approaches achieve similar scores, with a slight advantage for the

[5] UDPipe Future is available at https://github.com/CoNLL-UD-2018/UDPipe-Future, and the code for nested NER is available at https://github.com/ufal/acl2019_nested_ner.

fine-tuned version of CamemBERT, thus questioning the need for complex task-specific architectures such as UDPipe Future.

Despite a much simpler optimisation process and no task specific architecture, fine-tuning CamemBERT outperforms UDify on all treebanks and sometimes by a large margin (e.g. +4.15% LAS on Sequoia and +5.37 LAS on ParTUT). CamemBERT also reaches better performance than other multilingual pretrained models such as mBERT and $\text{XLM}_{\text{MLM-TLM}}$ on all treebanks.

CamemBERT achieves overall slightly better results than the previous state-of-the-art and task-specific architecture UDPipe Future+mBERT +Flair, except for POS tagging on Sequoia and POS tagging on Spoken, where CamemBERT lags by 0.03% and 0.14% UPOS respectively. UDPipe Future+mBERT +Flair uses the contextualized string embeddings Flair (Akbik et al., 2018), which are in fact pretrained contextualized character-level word embeddings specifically designed to handle misspelled words as well as subword structures such as prefixes and suffixes. This design choice might explain the difference in score for POS tagging with CamemBERT, especially for the Spoken treebank where words are not capitalized, a factor that might pose a problem for CamemBERT which was trained on capitalized data, but that might be properly handle by Flair and the UDPipe Future+mBERT +Flair model.

5.4.2 Named-Entity Recognition

Model	F1
SEM (CRF) (Dupont, 2017)	85.02
LSTM-CRF (Dupont, 2017)	85.57
mBERT (fine-tuned)	87.35
CamemBERT (fine-tuned)	<u>89.08</u>
LSTM+CRF+CamemBERT (embeddings)	**89.55**

Table 5.2: **NER** scores on the FTB (best model selected on validation out of 4). Best scores in bold, second best underlined.

For NER, we similarly evaluate CamemBERT in the fine-tuning setting and as input embeddings to the task specific architecture LSTM+CRF. We report these scores in Table 5.2.

In both scenarios, CamemBERT achieves higher F1 scores than the traditional CRF-based architectures, both non-neural and neural, and than fine-tuned multilingual BERT models.[6]

Using CamemBERT as embeddings to the traditional LSTM+CRF architecture gives slightly higher scores than by fine-tuning the model (89.08 vs. 89.55). This demonstrates that although CamemBERT can be used successfully without any task-specific architecture, it can still produce high quality contextualized embeddings that might be useful in scenarios where powerful downstream architectures exist.

5.4.3 Natural Language Inference

Model	Acc.	#Params
mBERT (Devlin et al., 2019a)	76.9	175M
$\text{XLM}_{\text{MLM-TLM}}$ (Lample and Conneau, 2019)	<u>80.2</u>	250M
$\text{XLM-R}_{\text{BASE}}$ (Conneau et al., 2020a)	80.1	270M
CamemBERT (fine-tuned)	**82.5**	110M
Supplement: LARGE models		
$\text{XLM-R}_{\text{LARGE}}$ (Conneau et al., 2020a)	<u>85.2</u>	550M
$\text{CamemBERT}_{\text{LARGE}}$ (fine-tuned)	**85.7**	335M

Table 5.3: **NLI** accuracy on the French XNLI test set (best model selected on validation out of 10). Best scores in bold, second best underlined.

On the XNLI benchmark, we compare CamemBERT to previous state-of-the-art multilingual models in the fine-tuning setting. In addition to the standard model with a BASE architecture, we train another model with the LARGE architecture, referred to as $\text{CamemBERT}_{\text{LARGE}}$, for a fair comparison with $\text{XLM-R}_{\text{LARGE}}$. This model is trained

[6] $\text{XLM}_{\text{MLM-TLM}}$ is a lower-case model. Case is crucial for NER, therefore we do not report its low performance (84.37%)

with the CCNet corpus, described in Sec. 5.5, for 100k steps.[7] We expect that training the model for longer would yield even better performance.

Our model reaches higher accuracy than its BASE counterparts reaching +5.6% over mBERT, +2.3 over $\text{XLM}_{\text{MLM-TLM}}$, and +2.4 over $\text{XLM-R}_{\text{BASE}}$. CamemBERT also uses as few as half as many parameters (110M vs. 270M for $\text{XLM-R}_{\text{BASE}}$).

$\text{CamemBERT}_{\text{LARGE}}$ achieves a state-of-the-art accuracy of 85.7% on the XNLI benchmark, as opposed to 85.2, for the recent $\text{XLM-R}_{\text{LARGE}}$.

Our model uses fewer parameters than multilingual models, mostly because of its smaller vocabulary size (e.g. 32k vs. 250k for XLM-R). Two elements might explain the better performance of CamemBERT over XLM-R. Even though XLM-R was trained on an impressive amount of data (2.5TB), only 57GB of this data is in French, whereas we used 138GB of French data. Additionally XLM-R also handles 100 languages, and the authors show that when reducing the number of languages to 7, they can reach 82.5% accuracy for French XNLI with their BASE architecture.

5.4.4 Summary of Camembert's results

CamemBERT improves the state of the art for the 4 downstream tasks considered, thereby confirming on French the usefulness of Transformer-based models. We obtain these results when using our model as a fine-tuned model and when we use it as as contextual embeddings with task-specific architectures. This questions the need for more complex downstream architectures, similar to what was shown for English (Devlin et al., 2019a). Additionally, this suggests that CamemBERT is also able to produce high-quality representations out-of-the-box without further tuning.

5.5 Impact of corpus origin and size

In this section we investigate the influence of the homogeneity and size of the pretraining corpus on downstream task performance. With this aim, we train alternative version of CamemBERT by varying the pretraining datasets. For this experiment, we fix the

[7] We train our LARGE model with the CCNet corpus for practical reasons. Given that BASE models reach similar performance when using OSCAR or CCNet as pretraining corpus (Appendix Table 5.6), we expect an OSCAR LARGE model to reach comparable scores.

Data	Size	Parsing		NER	NLI
		POS	LAS	F1	Acc.
Fine-tuning					
Wiki	4GB	97.45	88.75	89.86	78.32
CCNet	4GB	97.67	**90.04**	90.46	**82.06**
OSCAR	4GB	97.71	89.87	90.65	81.88
OSCAR	138GB	**97.79**	89.88	**91.55**	81.55
Embeddings					
Wiki	4GB	97.21	88.64	91.23	-
CCNet	4GB	97.81	90.04	**92.30**	-
OSCAR	4GB	**97.82**	**90.05**	91.90	-
OSCAR	138GB	97.77	89.84	91.83	-

Table 5.4: Results on the four tasks using language models pre-trained on data sets of varying homogeneity and size, reported on validation sets (average of 4 runs for POS tagging, parsing and NER, average of 10 runs for NLI). Parsing results are the macro-averaged score across four treebanks, namely the GSD, Sequoia, French Spoken, and Partut treebanks (cf. Table 9.1 in the Appendix for full results.). *Embeddings* is done using CamemBERT embeddings plugged to UDPipe Future for tagging and parsing; and an LSTM+CRF model for NER)

.

number of pretraining steps to 100k, and allow the number of epochs to vary accordingly (more epochs for smaller dataset sizes). All models use the BASE architecture.

In order to investigate the need for homogeneous clean data versus more diverse and possibly noisier data, we use alternative sources of pretraining data in addition to OSCAR:

- **Wikipedia**, which is homogeneous in terms of genre and style. We use the official 2019 French Wikipedia dumps[8]. We remove HTML tags and tables using Giuseppe Attardi's *WikiExtractor*.[9]
- **CCNet** (Wenzek et al., 2019), a dataset extracted from Common Crawl with a different filtering process than for OSCAR. It was built using a language model trained on Wikipedia, in order to filter out bad quality texts such as code or tables.[10] As this filtering step biases the noisy data from Common Crawl to more Wikipedia-like text, we expect CCNet to act as a middle ground between the unfiltered "noisy"

[8] https://dumps.wikimedia.org/backup-index.html.

[9] https://github.com/attardi/wikiextractor.

[10] We use the head split, which corresponds to the top 33% of documents in terms of filtering perplexity.

OSCAR dataset, and the "clean" Wikipedia dataset. As a result of the different filtering processes, CCNet contains longer documents on average compared to OSCAR with smaller—and often noisier—documents weeded out.

Table 5.5 summarizes statistics of these different corpora.

Corpus	Size	#tokens	#docs	Tokens/doc Percentiles: 5%	50%	95%
Wikipedia	4GB	990M	1.4M	102	363	2530
CCNet	135GB	31.9B	33.1M	128	414	2869
OSCAR	138GB	32.7B	59.4M	28	201	1946

Table 5.5: Statistics on the pretraining datasets used.

In order to make the comparison between these three sources of pretraining data, we randomly sample 4GB of text (at the document level) from OSCAR and CCNet, thereby creating samples of both Common-Crawl-based corpora of the same size as the French Wikipedia. These smaller 4GB samples also provides us a way to investigate the impact of pretraining data size. Downstream task performance for our alternative versions of CamemBERT are provided in Table 9.1. The upper section reports scores in the fine-tuning setting while the lower section reports scores for the embeddings.

5.5.1 Common Crawl vs. Wikipedia?

Table 9.1 clearly shows that models trained on the 4GB versions of OSCAR and CCNet (Common Crawl) perform consistently better than the the one trained on the French Wikipedia. This is true both in the fine-tuning and embeddings setting. Unsurprisingly, the gap is larger on tasks involving texts whose genre and style are more divergent from those of Wikipedia, such as tagging and parsing on the Spoken treebank. The performance gap is also very large on the XNLI task, probably as a consequence of the larger diversity of Common-Crawl-based corpora in terms of genres and topics. XNLI is indeed based on multiNLI which covers a range of genres of spoken and written text.

The downstream task performances of the models trained on the 4GB version of CCNet and OSCAR are much more similar.[11]

5.5.2 How many data do you need?

An unexpected outcome of our experiments is that the model trained "only" on the 4GB sample of OSCAR performs similarly to the standard CamemBERT trained on the whole 138GB OSCAR. The only task with a large performance gap is NER, where "138GB" models are better by 0.9 F1 points. This could be due to the higher number of named entities present in the larger corpora, which is beneficial for this task. On the contrary, other tasks don't seem to gain from the additional data.

In other words, when trained on corpora such as OSCAR and CCNet, which are heterogeneous in terms of genre and style, 4GB of uncompressed text is large enough as pretraining corpus to reach state-of-the-art results with the BASE architecure, better than those obtained with mBERT (pretrained on 60GB of text).[12] This calls into question the need to use a very large corpus such as OSCAR or CCNet when training a monolingual Transformer-based language model such as BERT or RoBERTa. Not only does this mean that the computational (and therefore environmental) cost of training a state-of-the-art language model can be reduced, but it also means that CamemBERT-like models can be trained for all languages for which a Common-Crawl-based corpus of 4GB or more can be created. OSCAR is available in 166 languages, and provides such a corpus for 38 languages. Moreover, it is possible that slightly smaller corpora (e.g. down to 1GB) could also prove sufficient to train high-performing language models. In addition, further research is needed to confirm the validity of our findings on larger architectures and other more complex natural language understanding tasks. However, even with a BASE architecture and 4GB of training data, the validation loss is still decreasing beyond 100k steps (and 400 epochs). This suggests that we are still under-fitting the 4GB pretraining dataset, training longer might increase downstream performance.

[11] We provide the results of a model trained on the whole CCNet corpus in the Appendix. The conclusions are similar when comparing models trained on the full corpora: downstream results are similar when using OSCAR or CCNet.

[12] The OSCAR-4GB model gets slightly better XNLI accuracy than the full OSCAR-138GB model (81.88 vs. 81.55). This might be due to the random seed used for pretraining, as each model is pretrained only once.

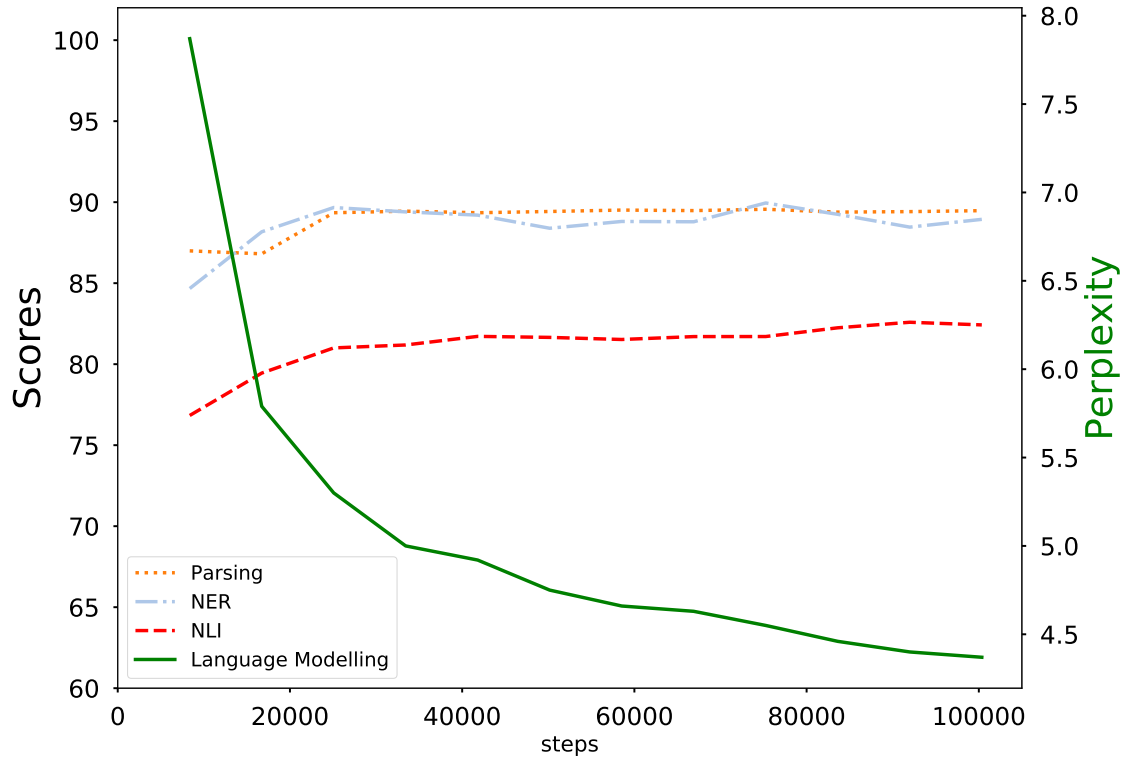


Figure 5.1: Impact of number of pretraining steps on downstream performance for CamemBERT.

.

5.6 How much training steps do you need?

Figure 5.1 displays the evolution of downstream task performance with respect to the number of steps. All scores in this section are averages from at least 4 runs with different random seeds. For POS tagging and dependency parsing, we also average the scores on the 4 treebanks.

We evaluate our model at every epoch (1 epoch equals 8360 steps). We report the masked language modelling perplexity along with downstream performances. Figure 5.1, suggests that the more complex the task the more impactful the number of steps is. We observe an early plateau for dependency parsing and NER at around 22k steps, while for NLI, even if the marginal improvement with regard to pretraining steps becomes smaller, the performance is still slowly increasing at 100k steps.

In Table 5.6, we compare two models trained on CCNet, one for 100k steps and the other for 500k steps to evaluate the influence of the total number of steps. The model

trained for 500k steps does not increase the scores much from just training for 100k steps in POS tagging and parsing. The increase is slightly higher for XNLI (+0.84).

Those results suggest that low-level syntactic representations are captured early in the language model training process while it needs more steps to extract complex semantic information as needed for NLI.

Dataset	Masking	Arch.	#Param.	#Steps	UPOS	LAS	NER	XNLI
Masking Strategy								
OSCAR	Subword	Base	110M	100k	97.78	89.80	**91.55**	81.04
OSCAR	Whole-word	Base	110M	100k	**97.79**	**89.88**	91.44	**81.55**
Model Size								
CCNet	Whole-word	Base	110M	100k	97.67	89.46	90.13	82.22
CCNet	Whole-word	Large	335M	100k	**97.74**	**89.82**	**92.47**	**85.73**
Dataset								
CCNet	Whole-word	Base	110M	100k	97.67	89.46	90.13	**82.22**
OSCAR	Whole-word	Base	110M	100k	**97.79**	**89.88**	**91.44**	81.55
Number of Steps								
CCNet	Whole-word	Base	110M	100k	**98.04**	89.85	90.13	82.20
CCNet	Whole-word	Base	110M	500k	97.95	**90.12**	91.30	**83.04**

Table 5.6: Comparing scores on the **Validation sets** of different design choices. POS tagging and parsing datasets are averaged. (average over multiple fine-tuning seeds).

5.7 Impact of Model Size

Table 5.6 compares models trained with the BASE and LARGE architectures. These models were trained with the CCNet corpus (135GB) for practical reasons. We confirm the positive influence of larger models on the NLI and NER tasks. The LARGE architecture leads to respectively 19.7% error reduction and 23.7%. To our surprise, on POS tagging and dependency parsing, having three time more parameters doesn't lead to a significant difference compared to the BASE model. Tenney et al. (2019) and Jawahar et al. (2019) have shown that low-level syntactic capabilities are learnt in lower layers of BERT while higher level semantic representations are found in upper layers of BERT.

POS tagging and dependency parsing probably do not benefit from adding more layers as the lower layers of the BASE architecture already capture what is necessary to complete these tasks.

5.8 Impact of Whole-Word Masking

In Table 5.6, we compare models trained using the traditional subword masking with whole-word masking. Whole-Word Masking positively impacts downstream performances for NLI (although only by 0.5 points of accuracy). To our surprise, this Whole-Word Masking scheme does not benefit much lower level task such as Name Entity Recognition, POS tagging and Dependency Parsing.

5.9 Discussion

Pretrained Masked Language Models have led to significant empirical progress in NLP for various tasks. After the first release of the BERT model (Devlin et al., 2019b), (Liu et al., 2019) supported the hypothesis that using hundreds of Gigabytes of data was required to reach state-of-the-art downstream performance. Indeed, they claimed that training their model on up to 160GB of data improves the performance even with a fixed computing budget (i.e., for the same number of training steps with the same batch size and number of parameters). With CamemBERT, we were the first to challenge this claim. We show that if the pretraining data originates in various sources (such as with OSCAR and CCNet), pretraining on as little as 4GB of data matches the performance of a model trained on up to 138GB of data.

Our finding was concurrently supported by Raffel et al. (2019) who showed that an encoder-decoder transformer, pretrained on about 0.5B tokens was competitive with a model pretrained on about 17B tokens for a large variety of tasks such as Machine Translation and the GLUE benchmark's tasks (Wang et al., 2018). It was further extended by Micheli et al. (2020) who showed that pretraining a model on about 100M of data leads to competitive downstream performance.

In consequence of our findings, listing the languages with at least 1 GB of data in the OSCAR corpus (Ortiz Suárez et al., 2019), our results show that about 63 languages

could potentially get an accurate large-scale monolingual language model. This includes a large number of languages that are considered low-resource, such as Punjabi (1.1GB available in OSCAR), Mongolian (2.8GB available), and Georgian (7.8GB available). For languages that do not have that much data, such as Yoruba or Maltese, we will see in part IV how cross-lingual transfer techniques based on large-scale multilingual language models can enable us to build accurate language models.

Part III

Addressing the Variability of Natural Languages

6 Lexical Normalization of User Generated Content

6.1 Motivations

In the last chapter, we assumed that the evaluation text originates from the same language and same set of domains as the training data. In practice in NLP, for most of the cases this hypothesis does not hold. Indeed, as discussed in chapter 2 language varies across domains, communities and time, specifically online (Jurafsky, 2018). This chapter focuses on non-canonical data originating from User Generated Content (UGC). UGC exhibits many linguistic phenomenons that make it different from domains for which we have large quantity of training data — mainly edited text such as Wikipedia data and News data — and on which we usually train our NLP systems (Foster, 2010b; Seddah et al., 2012; Eisenstein, 2013b; Baldwin et al., 2013; Plank, 2016).

Indeed, in NLP, most available training data originates from a limited set of domains. These domains are typically news and encyclopedic data (Plank, 2016) and referred to as canonical. By contrast, UGC includes linguistic phenomena usually not observed in canonical domains and are therefore referred to as non-canonical. As described by (Foster, 2010b; Seddah et al., 2012; Eisenstein, 2013b; Baldwin et al., 2013), these phenomena can be frequent spelling errors, simplification, specific jargons and use of phonetics.

To build systems that are more robust, there are two approaches one may take:

1. Collect UGC training data, annotate it for the task of interest and train or adapt a model directly on this data. We will study this in the chapter 8 in the multilingual setting,

2. Reduce the domain gap (cf. section 4.7.1) between the source training data (in our case edited standard text) and the target data (in our case non-standard UGC data).

In this chapter, we focus on the second approach. For UGC, one way to do this is to perform lexical normalization (introduced formally in §3.3.4 and discussed in §4.7.1). Our goal is to build a model to do lexical normalization of UGC data in English. As defined in section 3.3.4), lexical normalization is the task of translating non canonical words into canonical ones.

The type of linguistic phenomena that lexical normalization of UGC need to handle can be:

- spelling errors : *makeing* in *making*
- internet Slang : *lmfao* in *laughing my f.cking ass off* [1]
- contraction : *lil* for *little*
- abbreviation : *2nite* for *tonight*
- phonetics : *dat* for *that*

It also involves detecting that the following forms should be untouched: *':)', @Khalil-Brown, #Beyonce*

This chapter is an adapted version of (Muller et al., 2019). In summary, we present an enhancement of BERT's tokenization and architecture to fine-tune it efficiently for lexical normalization.

Data

We focus on lexical normalization in English. We base all our experiments on the WNUT data released by Baldwin et al. (2015). This dataset includes 2950 noisy tweets for training and 1967 for test. Out of the 44,385 training tokens, 3,928 require normalization leading to an unbalanced data set. Among those 3,928 noisy tokens, 1043 are 1-to-N (i.e. single noisy words that are normalized as several words) and 10 are N-to-1 cases (i.e. several noisy words that are normalized as single canonical words).

As highlighted before, our framework is more challenging than the standard approach to normalization, illustrated by the 2015 shared task, that usually authorizes external UGC

[1] Normalization found in the lexnorm 2015 dataset.

resources. As our goal is to test the ability of BERT, a model trained on canonical data only, we restrain ourselves to only using the training data as examples of normalization and nothing more.

Our work is therefore to build a domain transfer model in a low-resource setting.

6.2 Normalization with BERT

Until this work, lexical normalization was mainly approached with feature-rich modular systems. As discussed in 4.7.1, the best approach (van der Goot and van Noord, 2017; van der Goot, 2019) relied on a candidate generator module combined with a feature-based random-forest (Breiman, 2001) that ranks the candidates to find the best normalization form. In this chapter, we present an adaptation of the BERT model to perform lexical normalization end-to-end. If a word is considered to be canonical, the model will simply predict the same word. If it is considered to be non-canonical, the model predicts its normalized form.

6.2.1 BERT

We start by presenting the components of BERT that are relevant for our normalization model. All our work is done on the released *base* version.

WordPiece Tokenization

BERT takes as input sub-word units in the form of WordPiece tokens introduced in §4.2.5. We recall that the WordPiece vocabulary is computed based on the observed frequency of each sequence of characters of the corpus BERT is pre-trained on: Wikipedia and the BookCorpus. It results in a 32 thousand tokens vocabulary.

Reusing BERT, in any way, requires to use its original WordPiece vocabulary. In the context of handling non-canonical data, this is of primary importance. Indeed, frequent tokens in our non-canonical data set might not appear in the vocabulary of BERT and therefore will have to be split. For example, the word *lol* is a non-canonical word (it appears more than 222 times in the original lexnorm15 dataset). Still, it is not in BERT-base WordPiece vocabulary and will have to be split in two tokens. For

tokenization of WofrdPieces, we follow the implementation found in the *huggingface pytorch-pretrained-BERT* project.[2] It is implemented as a greedy matching algorithm. We write it in pseudo-code in Algorithm 3.

Algorithm 3 Greedy WordPiece tokenization

```
Vocabulary = Bert WordPiece Vocabulary;
init start=0, string=word,
wordPieceList = list()
while string not empty do
    substring:=string[start:]
    while substring not empty do
        if substring in Vocabulary then
            wordPieceList := wordPieceList U [substring]
            break loop
        else
            substring := substring[:-1]
        end
    end
    start := start + length(substring)
end
Result: wordPieceList
Note : Tokenizing words into wordpiece tokens, by matching in an iterative way from left to right, the
longest sub-string belonging to the wordpiece vocabulary
```

6.2.2 Fine-Tuning BERT for normalization

We now present the core of our contribution. How to make BERT a competitive normalization model? In a nutshell, there are many ways to do lexical normalization. Neural models have established the state-of-the-art in the related Grammatical Error Correction task using the sequence to sequence paradigm (Sutskever et al., 2014) at the character level. Still, this framework requires a large amount of parallel data. Our preliminary experiments showed that this was unusable for UGC normalization. Even the use a powerful pre-trained model such as BERT for initializing an encoder-decoder requires the decoder to learn an implicit mapping between noisy words and canonical ones. This is not reachable with only three thousand sentences.

[2] https://github.com/huggingface/pytorch-pretrained-BERT

Noisy	Gold	#next mask
ye	ye	0
##a	##ah	0
im	i	2
[MASK]	’	-
[MASK]	m	-
already	already	0
knowing	knowing	0
wa	wh	0
##t	##at	0

Table 6.1: Parallel Alignment of *yea im already knowing wat u sayin* normalized as *yeah i’m already knowing what you saying* with gold number of next masks for each source token

We therefore adapted BERT in a direct way for normalization. As described in section 6.2, BERT Masked Language Model ability allows token prediction. Simply feeding the model with noisy tokens on the input and fine-tuning on canonical token labels transforms BERT into a normalization model. There are two critical points in doing so successfully. The first is that it requires WordPiece alignment (cf. section 6.2.2). The second is that it requires careful fine-tuning (cf. section 6.2.2).

Wordpiece Alignment

We have in a majority of cases, as described in section 6.1, word-level alignment between non-canonical and canonical text. Still, the dataset also includes words that are not aligned. For 1-to-N cases, we simply remove the spaces. As we work at the WordPiece level this does not bring any issue. For N-to-1 cases (only 10 observations), by considering the special token "|" of the lexnorm15 dataset as any other token, we simply handle source multi-words as a single one, and let the wordpiece tokenization splitting them.

We frame normalization as a 1-to-1 WordPiece token mapping. Based on the word level alignment, we present two methods to get WordPiece alignment: an *Independent Alignment* approach and a *Parallel Alignment* one.

Independent Alignment We tokenize noisy words and non-noisy ones independently (cf. algorithm 3). By doing so, for each word, we get non-aligned WordPiece tokens. We handle it in three ways :

noisy	canonical
ye	yeah
##a	[SPACE]
im	i
[MASK]	’
[MASK]	m
already	already
knowing	knowing
wa	what
##t	[SPACE]

Table 6.2: Independent Alignment of *yea im already knowing wat u sayin* normalized as *yeah i’m already knowing what you saying*

- If we get as many noisy tokens as standard tokens, we keep the alignment as such,
- If there are more tokens on the target side, we append the special token [MASK] on the source side. This means that we force the model to predict a token at training time.
- If there are more tokens on the source side, we introduce a new special token [SPACE].

An alignment example extracted from lexnorm15 can be found in table 6.2. As we can see, this simple token alignment algorithm leads to introducing multiple [MASK] and [SPACE] tokens that will have to be handled by the model.

Parallel Alignment We enhance this first approach with a *parallel alignment* method, described in Algorithm 4.

Our goal is to minimize the number of [MASK] and [SPACE] appended into the source and gold sequences. Therefore, for each word, we start by tokenizing into WordPieces the noisy source word. For each WordPiece, we start the tokenization on the gold side, starting and ending from the same character positions. As soon as we tokenized the entire gold sub-string, we switch to the next noisy sub-string and so on. By doing so, we ensure a closer alignment at the WordPiece level. We illustrate on the same example this enhanced parallel alignment in Table 6.3.

We highlight two aspects of our alignment techniques. First, introducing the special token [SPACE] induces an architecture change in the MLM head. We detail this in

Algorithm 4 Parallel WordPiece tokenization

```
Vocabulary = Bert WordPiece Vocabulary;
Init start=0; string=canonical word;
string_noisy = noisy word; end_gold=0; wordPListNoisy=list(); wordPieceListGold=list();
while string_noisy not empty do
    string_noisy:=string_noisy[start:]
    substr_noisy:=string_noisy
    while substr_noisy not empty do
        breaking:=False
        if substr_noisy in Vocabulary then
            wordPListNoisy := wordPListNoisy U [substr_noisy]
            if start equals length string_noisy then
                end_gold:=length(string)
            else
                end_gold:= start+length(substr_noisy)
            end
            while substr_gold not empty do
                substr_gold:= string[start:end_gold]

                if substr_gold in Vocabulary then
                    wordPieceListGold:= wordPieceListGold U [substr_gold]
                    break loop
                else
                    end_gold := end_gold -1
                end
            end
        else
            substr_noisy:=substr_noisy[:-1]
        end
        if breaking then
            break loop
        end
    end
    start := start + length(substr_noisy)
end
```

Result: wordPListNoisy

Note : Tokenizing noisy tokens and canonical tokens in wordpieces in parrallel to minimize the number of appended [MASK] and [SPACE]

section 6.2.2-(A). Second, appending the extra token [MASK] on the source side based on the gold sequence induces a discrepancy between training and testing. Indeed, at test time, we do not have the information about whether we need to add an extra token or not. We describe in section 6.2.2-(B) how we extend BERT's architecture with the addition of an extra classification module to handle this discrepancy.

Architecture Enhancements

(A) Enhancing BERT MLM with [SPACE]

In order to formalize lexical normalization as a token prediction we introduced in section 6.2.2 the need for a new special token [SPACE]. We want our normalization model to predict it. We therefore introduce a new label in our output WordPiece vocabulary as well as a new vector in the last softmax layer. We do so in a straightforward way by appending to the output matrix a vector sampled from a normal distribution[3].

(B) #Next [MASK] predictor

As we have described, alignment requires in some cases the introduction of [MASK] tokens within the source sequence based on the gold sequence. We handle the discrepancy introduced between training and testing in the following way. We add an extra token classification module to BERT architecture. This module takes as input BERT last hidden state of each WordPiece tokens and predict the number of [MASK] to append next

In table 6.1, we illustrate the training signal of the overall architecture. It takes noisy WordPiece tokens as input. As gold labels, it takes on the one side the gold WordPiece tokens and on the other side the number of [MASK] to append next to each source WordPiece tokens.

At test time, we first predict the number of next masks to introduce in the noisy sequence. We then predict normalized tokens using the full sequence.

This *#next mask* prediction module exceeds the context of normalization. Indeed, it provides a straightforward way of performing data augmentation on any Masked Language Model architecture. We leave to future work the investigation of its impact beyond lexical normalization.

Fine-Tuning

We describe here how we fine-tune our architecture for normalization. Our goal is to learn lexical normalization in a general manner. To do so, intuitively, our model needs to: on the one hand, preserve its language model ability that will allow generalization. On

[3]Each dimension $v_d \sim \mathcal{N}(mean_i(x_d), \sigma_i^2(x_d))$ (i indexing the WordPiece vocabulary and d the dense dimension of BERT output layer), $mean_i$ (resp. σ_i^2) means mean (resp. variance) along the i dimension

Noisy	Canonical
ye	ye
##a	##ah
im	i
[MASK]	’
[MASK]	m
already	already
knowing	knowing
wa	wh
##t	##at

Table 6.3: Parallel Alignment of *yea im already knowing wat u sayin* normalized as *yeah i’m already knowing what you saying*

the other hand, the MLM needs to adjust itself to learn alignment between noisy tokens and canonical tokens.

Based on those intuitions, we performe fine-tuning in the following way:

(i) Our first approach is to back-propagate on all tokens at each iteration. We also dropout 10% of input tokens by replacing them with the [MASK] as done during BERT pre-training. In this setting, all tokens are considered indifferently whether they require normalization or not .

(ii) The second approach that happens to perform the best is our *Noise-focus* fine-tuning. The intuition is that it should be much easier for the model to learn to predict already normalized tokens than the ones that require normalization. For this reason, we design the following strategy: For a specific portion of batches noted p_{noise} we only back-propagate through noisy tokens. We found that having an increasing number of noise-specific batch while training provides the best results.

Formally we describe our strategy as follows. For each mini-batch, we sample b following the distribution $b \sim \text{Bernoulli}(p_{noise})$, with $p_{noise} = \min\left(\frac{epoch}{n\_epoch}, 0.5\right)$, $epoch$ being the current number of epoch and n_epoch the total number of epochs.

If b equals 1 we back-propagate through noisy tokens, otherwise we back-propagate in the standard way on all the tokens. In other words, while training, for an increasing portion of batches, we train on tokens that require normalization. We found that this dynamic strategy was much more efficient than applying a static p_{noise}. Moreover, we highlight that the portion of noise specific update is capped at 50% (0.5 in the equation).

Above this value, we observed that the performances degraded in predicting non-noisy tokens.

Optimization Details

Note that, excluding the fine-tuning strategy and the alignment algorithm, the optimization hyper-parameters are shared to all the experiments we present next. Generally speaking, we found that optimizing BERT for lexical normalization with WordPiece alignment is extremely sensitive to hyper-parameters. We managed to reach values that work in all our following experiments. For the optimization, we use the Adam algorithm (Kingma and Ba, 2015). We found that 1e-5 provides the most stable and consistent convergence across all experiments as evaluated on validation set. We found that a mini-batch of dimension 4 brings the best performance also across all experiments. Finally, we kept a dropout value of 0.1 within the entire BERT model. We train the model for up to 10 epochs and used performance as measured with the F1-score (detailed in the next section) on the validation set as our early-stopping metric.

6.3 Experiments

All our experiments are run on the lexnorm15 dataset. We do not use any other resources making our problem falling under a low-resource domain transfer framework. As only pre-processing, we lower-case all tokens whether they are on the noisy source side or on the canonical side.

We first present our analysis on the validation set that corresponds to the last 450 sentences of the original training set of lexnorm15.

The evaluation metrics on which we make our analysis are defined in §3.3.4. Following previous works, we focus on the F1 score as our main evaluation metric. F1 is simply the harmonic mean of the recall and precision. For more fine grained analysis we also report the recall on sub-sample of the evaluated dataset. Particularly, we distinguish between Out-of-Vocabulary (OOV) and In-Vocabulary words (InV) and report the recall on those subsets. We define it formally as:

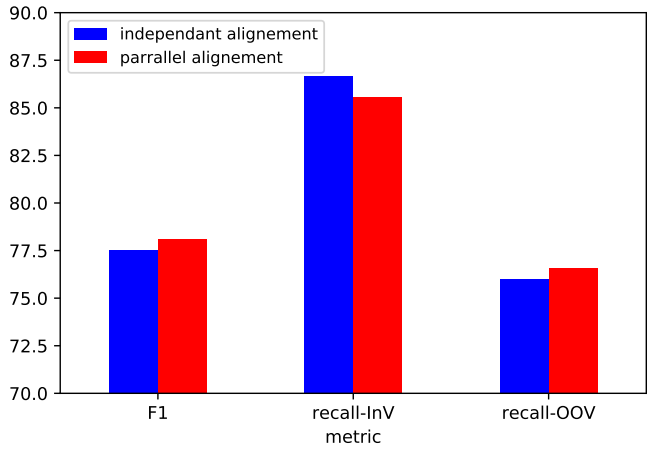


Figure 6.1: Impact of noisy/canonical alignment method with a focus on generalization by comparing Out-of-Vocabulary (OOV) and In-Vocabulary (InV) performance (development set)

$$recall\_sample = \frac{TP \cap sample}{\#need\_norm \cap sample}$$

6.3.1 Alignment algorithm

Does enforcing alignment in a greedy way as described in Algorithm 4 help normalization?

As we compare in figure 6.1, our parallel alignment method provides a +0.5 F1 improvement (78.1 vs 77.6 F1). We also compare the performance of our two models on OOV and InV words. Indeed, normalising a seen word is much easier than a word unseen during training. As we observe, the gain coming from our our alignment technique come from a better generalization. We gain +0.6 in recall on OOV thanks to this parallel alignment.

6.3.2 Fine-Tuning Strategy

As observed in table 6.4, our fine-tuning strategy focused on noisy tokens improves with a large margin the performance of our system. We interpret it in the following way: lexical normalization is imbalanced. As seen in 6.1 there are around 9 times more *need_no_norm* than *need_norm* tokens. By specifically training on noisy tokens we successfully manage to alleviate this aspect of the data.

Standard	Noise-focused	Gain
78.1	79.28	+1.18

Table 6.4: Impact of our noise-specific strategy on the F1 score (development set) reported with best alignment setting

Model	Accuracy
BERT noise-focused	97.5
MoNoise	97.6

Table 6.5: Comparing our systems to the State-of-the-art system MoNoise (we report on same development dataset reported in MoNoise original paper (last 950 sentences))

In conclusion, our best model is BERT trained on parallel tokenized data with the noise-focus fine-tuning strategy. We reach 79.28 in F1 score. The following table illustrates how our model performs normalization on a typical example:

Noisy	*@aijaee i hear you u knw betta to cross mine tho*
Norm	*@aijaee i hear you you know better to cross mine though*

6.4 Discussion

We now compare our system to previous works. As we see in Table 6.7, our non-UGC system is far from the State-of-the-Art model MoNoise (van der Goot and van Noord, 2017) in terms of F1 score. In order to take into account detection in our metric, we also report the overall accuracy of the system in table 6.5. We are therefore 6.7 points below in terms of F1 score and 0.2 point below in terms of overall accuracy on lexnorm15 dataset.

However, we note that MoNoise is a feature-based Random Forest based on external modules. Among others, it makes use of a skip-gram model trained on 5 millions tweets, the Aspell tool and a n-gram model trained on more than 700 millions tweets.

In order to have a more balanced comparison, we compare our system to the MoNoise model after removing the feature that has the most impact, according to the original paper: the n-gram module (referred as *MoNoise no n-gram*). In this setting, we significantly outperform the MoNoise model (+1.78 improvement) (Table 6.7).

Model	F1
Supranovich and Patsepnia, 2015	82.72
Berend and Tasnádi, 2015	80.52
our best model	**79.28**
Beckley, 2015	75.71
GIGO	72.64
Ruiz et al., 2014	53.1

Table 6.6: Comparing our systems to WNUT 2015 shared task that allowed UGC resources

Model	F1	UGC resources	speed
MoNoise	86.39	lex15+700Mtweets	57s
our best model	**79.28**	lexnorm15	9.5s
MoNoise NNG	77.5	lex15+5Mtweets	-

Table 6.7: Comparing our systems to the State-of-the-art system MoNoise on lexnorm15 test. Speed is reported as time to predict 1000 tokens (includes model loading). *MoNoise No-Ngrams* or MoNoise NNG is the score reported in the original paper without the use of UGC-n-grams but with a UGC word2vec

Moreover, we based all our work on the lexnorm15 dataset released for the W-NUT 2015 shared task (Baldwin et al., 2015). We compare our model to the competing systems (cf. table 6.6). Briefly, the second best model (Berend and Tasnádi, 2015) use a n-gram model trained on a English tweet corpus. The best competing system (Supranovich and Patsepnia, 2015) is based on a lexicon extracted from tweets. Still, we see that our model is able to outperform models ranked 3, 4 and 5 that are all built using UGC resources.

Finally, the state-of-the-art models we presented are modular. They require features from external modules. This makes them extremely slow at test time. We compare it in Table 6.7, demonstrating another practical interest for our approach. Our model is 6 times faster than MoNoise at prediction time.

Following those observations, we claim that BERT, enhanced to handle token introduction and token removal, fine-tuned in a precise way toward noisy words, is a competitive lexical normalization model.

This result exceeds the context of lexical normalization of noisy User Generated Content. Indeed, the success of BERT in improving NLP models on a diversity of tasks was, until now, restricted to canonical edited texts. In our work, we showed that it was possible to adapt such a general model to the extreme case of normalizing non-canonical UGC in

a low-resource setting. We let for future work the adaptation of BERT to other tasks in a non-canonical context.

6.5 Conclusion

Two years after this work, van der Goot et al. (2021) organized the MultiLexNorm shared task. I contributed to the organization of the shared task by participating in the discussions and setting up the evaluation platform that can be found at https://competitions.codalab.org/competitions/34355. It offered 9 participants the opportunity to compete in the lexical normalization task across 12 language varieties. Based on the ByT5 model (Xue et al., 2022) — a byte-level (cf. §4.2.6) multilingual pretrained T5 model, fine-tuned on word-level lexical normalization, Samuel and Straka (2021) outperformed the former state-of-the-art (van der Goot and van Noord, 2017) by about 17 points in average across the 12 languages. This showed that pretrained sequence to sequence models like T5 is inherently better at performing lexical normalization than masked language models.

However, the downstream impact of better normalization did not lead to important performance progress. Indeed, van der Goot et al. (2021) showed that performing dependency parsing on predicted normalized forms led to only 0.8 improvements in LAS score in average.

Overall, these results suggest that even if done accurately, lexical normalization has only a limited impact on downstream performance. Indeed, language variability and domain gap is only partially related to lexical differences. Stylistic, syntactic, type of sentences, and sentence length are also at play when we model UGC data. Reducing those differences in practice is impossible due to the lack of parallel data between non-canonical and canonical domains. Additionally, subword-level (§4.2.5) language models like BERT are surprisingly robust to lexical variability (that lexical normalization aims at reducing). As shown by Itzhak and Levy (2022), the subword-token of those models encodes rich character-level information. This could explain their robustness to lexical variability and the fact that they are not outperformed by character-level models (Riabi et al., 2021).

As described extensively in chapter 5, with the emergence of the pretraining-fine-tuning approach (§4.6.3), it became possible to use a large quantity of data to train NLP systems. In many cases, though, due to the cost of training and the lack of large quantity of data, it is not possible to pretrain from scratch a model on a specific language and domain (cf. §4.6.3). The following chapters will study how we can overcome this limit. We will see how we can use pretrained models and adapt them to specific domains and languages.

Part IV

Addressing the Diversity of Natural Languages

7 Understanding Zero-Shot Cross-Lingual Transfer

7.1 Motivations

Before moving to adapting large-scale language models to low-resource languages, we first look at how we could use them directly on data different from our training data. We focus on the zero-shot cross-lingual transfer setting (introduced in §4.7.2). In this setting, we assume that we fine-tune a model on a given language – referred to as the source language. We then evaluate it on another language referred to as the target language.

This chapter is an adapted version of (Muller et al., 2021b) done in collaboration with Yanai Elazar. Based on many fruitful meetings and discussions with Yanai over several months, I was mainly responsible for designing and running the behavioral and structural analysis.[1]

Remarkably, as illustrated in table 7.1 for dependency parsing, large-scale multilingual language models such as mBERT reach non-trivial performance on nearly all language pairs studied. In some cases, the performance are quite high compared to non-deep learning based baselines (e.g. when we transfer from English to French for instance).

The source of such a successful transfer is still largely unexplained. Pires et al. (2019a) hypothesize that these models learn shared multilingual representations during pretraining. Focusing on syntax, Chi et al. (2020) recently showed that mBERT (Devlin et al., 2019b), encodes linguistic properties in shared multilingual sub-spaces. Gonen et al. (2020) suggest that mBERT learns a language encoding component and an abstract cross-lingual component. In this work, we are interested in understanding the mechanism that

[1]The code to run the analysis is available at https://github.com/benjamin-mlr/first-align-then-predict-w-RANDOM-INIT

Target - Source	English	Arabic	Russian
English	89.0	25.5	61.4
Arabic	35.9	59.5	59.4
Russian	62.5	42.3	85.2
Chinese	27.5	11.5	28.8
Czech	60.4	40.0	72.8
Finnish	48.4	28.2	52.9
French	74.0	28.9	65.8
German	70.3	27.1	65.9
Hindi	28.9	10.2	28.7
Indonesian	44.1	36.2	47.6
Italian	74.5	28.9	65.3
Japanese	12.0	16.7	15.1
Polish	55.2	41.2	66.1
Portuguese	68.6	34.7	66.4
Slovenian	73.1	35.8	62.9
Spanish	70.0	32 .0	66.7
Turkish	34.0	19.6	32.2
Mean	53.2	28.0	55.4

Table 7.1: Dependency Parsing Zero-Shot Cross-lingual transfer performance (measured with the LAS score) of mBERT fine-tuned on a Source language and evaluated on a Target language.

leads mBERT to perform zero-shot cross-lingual transfer. More specifically, we ask **what parts of the model and what mechanisms support cross-lingual transfer?**

By combining behavioral and structural analyses (Belinkov et al., 2020), we show that mBERT operates as the stacking of two modules: (1) A multilingual encoder, located in the lower part of the model, critical for cross-lingual transfer, is in charge of aligning multilingual representations; and (2) a task-specific, language-agnostic predictor which has little importance for cross-lingual transfer and is dedicated to performing the downstream task. This mechanism that emerges out-of-the-box, without any explicit supervision, suggests that mBERT behaves like the standard cross-lingual pipeline.

7.2 Analysis Techniques

We study mBERT with a novel behavioral test that disentangles the task fine-tuning influence from the pretraining step (§7.2.1), and a structural analysis on the intermediate

representations (§7.2.2). Combining the results from these analyses allows us to locate the cross-lingual transfer and gain insights into the mechanisms that enable it.

7.2.1 Locating Transfer with Random-init

In order to disentangle the impact of the pretraining step from the fine-tuning, we propose a new behavioral technique: Random-init. First, we randomly initialize a set of parameters (e.g. all the parameters of a given layer) instead of using the parameters learned during the pretraining step. Then, we fine-tune the modified pretrained model and measure the downstream performance.[2]

By replacing a given set of pretrained parameters and fine-tuning the model, *all other factors being equal*, Random-init enables us to quantify the contribution of a given set of pretrained parameters on downstream performance and therefore to locate which pretrained parameters contribute to the cross-lingual transfer.

If the cross-lingual performance is significantly lower than same-language performance, we conclude that these layers are more important to cross-lingual performance than they are for same-language performance. If the cross-lingual score does not change, it indicates that cross-lingual transfer does not rely on these layers.

This technique is reminiscent of the recent *Amnesic Probing* method (Elazar et al., 2020), that removes from the representation a specific feature, e.g. Part-of-Speech, and then measures the outcome on the downstream task. In contrast, Random-init allows to study a specific architecture component, instead of specific features.

7.2.2 Hidden State Similarities across Languages

To strengthen the behavioral evidence brought by Random-init, and provide finer analyses that focus on individual layers, we study how the textual representations differ between parallel sentences in different languages. We hypothesize that an efficient fine-tuned model should be able to represent similar sentences in the source and target languages similarly, even-though it was fine-tuned only on the source language.

[2]Note that we perform the same optimization procedure for the model with and w/o Random-init (optimal learning rate and batch size are chosen with grid-search).

To measure the similarities of the representation across languages, we use the Central Kernel Alignment metric (CKA), introduced by Kornblith et al. (2019). The CKA is invariant to isotropic scaling and invariant to orthonormal transformation (i.e. rotations), two properties that are needed to preserve the structure of deep-learning models.[3] We follow Conneau et al. (2020c) who use the CKA as a similarity metric to compare the representations of monolingual and bilingual pretrained models across languages.

For a given source language l and a target language l', we collect a 1000 pairs of aligned sentences from the UD-PUD treebanks (Zeman et al., 2017). For a given model and for each layer, we get a single sentence embedding by averaging token-level embeddings (after excluding special tokens). We then concatenate the 1000 sentence embedding vectors and get the matrices X_l and X'_l. Based on these two matrices, the CKA between the language l and the language l' is defined as:

$$CKA(X_l, X_{l'}) = \frac{||X_l^T X_{l'}||_F^2}{||X_l^T X_l||_F \, ||X_{l'}^T X_{l'}||_F} \tag{7.1}$$

with $||.||_F$ defining the Frobenius norm.

We use the CKA to study the representation difference between source and target languages in pretrained and fine-tuned multilingual models. For every layer, we average all contextualized tokens in a sentence to get a single vector.[4] Then we compute the similarity between target and source representations and compare it across layers in the pretrained and fine-tuned models. We call this metric the *cross-lingual similarity*.

7.3 Experimental Setting

7.3.1 Data Sources

We base our experiments on data originated from three sources: the Universal Dependency project (McDonald et al., 2013b) and the WikiNER dataset (Pan et al., 2017). We also

[3] As described in (Kornblith et al., 2019), orthogonal transformation of the input data do not affect the training process of deep-learning models (LeCun et al., 1990) so a similarity metric should be invariant to it.

[4] After removing [CLS] and [SEP] special tokens.

make use of the CoNLL-2003 shared task NER English dataset[5] (Tjong Kim Sang and De Meulder, 2003).

7.3.2 Languages

For all our experiments, we use English, Russian and Arabic as source languages in addition to Chinese, Czech, Finish, French, Indonesian, Italian, Japanese, German, Hindi, Polish, Portuguese, Slovenian, Spanish, and Turkish as target languages.

7.3.3 Fine-tuning Data

For all the cross-lingual experiments, we use English, Russian and Arabic as source languages on which we fine-tune mBERT. For English, we take the English-EWT treebank (Silveira et al., 2014) for fine-tuning, for Russian the Russian-GSD[6] treebank and for Arabic the Arabic-PADT treebank (Hajič et al., 2009).

7.3.4 Evaluation Data

Cross-Lingual Transfer Experiments

For all our experiments, we perform the evaluation on all the 17 languages. For Parsing and POS tagging we use the test set from the Parallel UD (PUD) treebanks released for the CoNLL Shared Task 2017 (Zeman et al., 2017). For NER, we use the corresponding annotated datasets in the wikiner dataset.

Domain Analysis Datasets

We list here the datasets for completing our domain analysis experiment in Section 7.4 reported in Table 7.3. To have a full control on the source domains, we use for fine-tuning the English Partut treebank for POS tagging and parsing (Svizzera, 2014). It is a mix of legal, news and wikipedia text. For NER, we keep the WikiANN dataset (Pan et al., 2017). For the same-language and out-of-domain experiments, we use the English-EWT, English-Lines and English Lexnorm (van der Goot and van Noord, 2018)

[5] https://www.clips.uantwerpen.be/conll2003/

[6] https://github.com/UniversalDependencies/UD_Russian-GSD

treebanks for Web Media data, Literature data and Noisy tweets respectively. For the cross-lingual French evaluation, we use the translation of the English test set,[7] as well as the French-GSD treebank. For NER, we take the CoNLL-2003 shared task English data as our out-of-domain evaluation extracted from the *News* domain. We note that the absolute performance on this dataset is not directly comparable to the one on the source wikiner. Indeed, the CoNLL-2003 dataset uses an extra MISC class. In our work, we only interpret the relative performance of different models on this test set.

7.4 Disentangling the Pretraining Effect

For each experiment, we measure the impact of randomly-initializing specific layers as the difference between the model performance without any random-initialization (Ref) and with random-initialization (Random-init). Results for two consecutive layers are shown in table 7.2. The rest of the results, which exhibit similar trends, can be found in the Appendix (table 9.2). [8]

For all tasks, we observe sharp drops in the cross-lingual performance at the lower layers of the model but only moderate drops in the same-language performance. For instance, the parsing experiment with English as the source language, results in a performance drop on English of only 0.96 points (En-En), when randomly-initializing layers 1 and 2. However, it leads to an average drop of 15.77 points on other languages (En-X). Furthermore, we show that applying Random-init to the upper layers does not harm same-language and cross-lingual performances (e.g. when training on parsing for English, the performance slightly decreases by 0.09 points in the same-language while it increases by 1.00 in the cross-lingual case). This suggests that the upper layers are *task-specific* and *language-agnostic*, since re-initializing them have minimal change on performance. We conclude that mBERT's upper layers do not contribute to cross-lingual transfer.

[7] We do so by taking the French-ParTUT test set that overlaps with the English-ParTUT, which is made of 110 sentences.

[8] The detailed results for POS tagging, parsing and NER can be found in the Appendix in tables 9.3, 9.4 and 9.5.

Does the Target Domain Matter?

In order to test whether this behavior is specific to the cross-lingual setting and is not general to any out-of-distribution transfer, we repeat the same Random-init experiment by evaluating on same-language setting while varying the evaluated domain.[9] If the drop is similar to cross-lingual performance, it means that lower layers are important for out-of-distribution transfer in general. Otherwise, it would confirm that these layers play a specific role for cross-lingual transfer.

We report the results in table 7.3. For all the analyzed domains (Web, News, Literature, etc.) applying Random-init to the two first layers of the models leads to very moderate drops (e.g. -0.91 when the target domain is English Literature for parsing), while it leads to large drops when the evaluation is done on a distinct language (e.g. -5.82 when evaluated on French). The trends are similar for all the domains and tasks we tested on. We conclude that the pretrained parameters at the lower layers are consistently more critical for cross-lingual transfer than for same-language transfer, and cannot be explained by the possibly different domain of the evaluated datasets.

7.5 Cross-Lingual Similarity in mBERT

The results from the previous sections suggest that the lower layers of the model are responsible for the cross lingual transfer, whereas the upper layers are language-agnostic. In this section, we assess the transfer by directly analyzing the intermediate representations and measuring the similarities of the hidden state representations between source and target languages. We compute the CKA metric (cf. equation 7.1 in §7.2.2) between the source and the target representations for pretrained and fine-tuned models using parallel sentences from the PUD dataset (Zeman et al., 2017). In Figure 7.1, we present the similarities between Russian and English with mBERT pretrained and fine-tuned on the three tasks.[10]

[9] Although other factors might play a part in out-of-distribution, we suspect that domains plays a crucial part in transfer. Moreover, it was shown that BERT encodes out-of-the-box domain information (Aharoni and Goldberg, 2020)

[10] We report the comparisons for 5 other languages in Figure 9.1 in the Appendix.

Src-Trg	Ref	*Random-init of layers* Δ1-2	Δ3-4	Δ5-6	Δ7-8	Δ9-10	Δ11-12
			Parsing				
En - En	88.98	-0.96	-0.66	-0.93	-0.55	0.04	-0.09
Ru - Ru	85.15	-0.82	-1.38	-1.51	-0.86	-0.29	0.18
Ar - Ar	59.54	-0.78	-2.14	-1.20	-0.67	-0.27	0.08
En - X	53.23	-15.77	-6.51	-3.39	-1.47	0.29	1.00
Ru - X	55.41	-7.69	-3.71	-3.13	-1.70	0.92	0.94
Ar - X	27.97	-4.91	-3.17	-1.48	-1.68	-0.36	-0.14
			POS				
En - En	96.51	-0.30	-0.25	-0.40	-0.00	0.05	0.02
Ru - Ru	96.90	-0.52	-0.55	-0.40	-0.07	0.02	-0.03
Ar - Ar	79.28	-0.35	-0.49	-0.36	-0.19	-0.05	-0.00
En - X	79.37	-8.94	-2.49	-1.66	-0.88	0.20	-0.14
Ru - X	79.25	-10.08	-2.83	-1.65	-2.74	0.01	-0.45
Ar - X	64.81	-6.73	-3.50	-1.63	-1.56	-0.73	-1.29
			NER				
En - En	83.30	-2.66	-2.14	-1.43	-0.63	-0.23	-0.12
Ru - Ru	88.20	-2.08	-2.13	-1.52	-0.64	-0.33	-0.13
Ar - Ar	87.97	-2.37	-2.11	-0.96	-0.39	-0.15	0.21
En - X	64.17	-8.28	-5.09	-3.07	-0.79	-0.47	-0.13
Ru - X	62.13	-15.85	-9.36	-5.50	-2.44	-1.16	-0.06
Ar - X	65.59	-16.10	-8.42	-3.73	-1.40	-0.25	0.67

Table 7.2: Relative Zero shot Cross-Lingual performance of mBERT with Random-init (§7.2.1) on pairs of consecutive layers compared to mBERT without any random-initialization (Ref). In Src-Trg, Src indicates the source language on which we fine-tune mBERT, and Trg the target language on which we evaluate it. Src-X is the average across all 17 target language with X $\neq$ Src. Detailed results per target language are reported in tables 9.4, 9.3 and 9.5 in the Appendix. Coloring is computed based on how mBERT with Random-init performs compared to the Ref model. $\geq$ Ref | $<$ Ref | $\leq$ -2 points | $\leq$ -5 points

The cross-lingual similarity between the representations constantly increases up to layer 5 for all the three tasks (reaching 78.1%, 78.1% and 78.2% for parsing, POS tagging and NER respectively). From these layers forward, the similarity decreases. We observe the same trends across all languages as reported in the Appendix in figure 9.1. This demonstrates that the fine-tuned model creates similar representations regardless of the language and task, and hints on an alignment that occurs in the lower part of the model. Interestingly, the same trend is also observed in the pretrained model, suggesting that the fine-tuning step preserves the multilingual alignment.

		Random-init of layers					
Src - Trg	Ref	Δ0-1	Δ2-3	Δ4-5	Δ6-7	Δ8-9	Δ10-11
Domain Analyses				*Parsing*			
En - En	90.40	-1.41	-2.33	-1.57	-1.43	-0.60	-0.46
En - En Lit.	77.91	-0.91	-1.38	-1.85	-0.83	-0.23	-0.17
En - En Web	75.77	-2.14	-2.42	-2.54	-1.42	-0.71	-0.69
En - En UGC	45.90	-1.97	-2.75	-2.10	-1.04	-0.39	-0.25
Cross-Language							
En - Fr tran.	83.25	-5.82	-2.69	-2.42	-0.44	0.25	0.94
En - Fr Wiki	71.29	-7.86	-4.33	-4.64	-0.92	-0.11	0.33
Domain Analyses				*POS*			
En - En	96.83	-1.35	-0.98	-0.70	-0.40	-0.28	-0.24
En - En Lit.	93.09	-0.58	-0.65	-0.28	-0.04	-0.06	0.12
En - En Web	89.67	-1.07	-1.21	-0.41	-0.10	0.03	0.21
En - En UGC	68.93	-2.38	-1.07	-0.14	0.54	-0.04	0.63
Cross-Language							
En - Fr Tran.	93.43	-3.59	-0.88	-1.31	-0.56	0.46	0.25
En - Fr.	91.13	-5.10	-0.93	-1.16	-0.74	0.15	-0.07
Domain Analyses				*NER*			
En - En	83.22	-2.45	-2.15	-1.28	-0.49	-0.15	-0.06
En - News	51.72	-1.32	-1.05	-0.80	-0.14	-0.31	-0.33
Cross-Language							
En - Fr	76.16	-5.14	-2.82	-1.97	-0.33	0.52	0.34

Table 7.3: Relative Zero shot Cross-Lingual performance of mBERT with Random-init (§7.2.1) on pairs of consecutive layers compared to mBERT without any random-initialization (Ref). We present experiments with English as the source language and evaluate across various target domains in English in comparison with the cross-lingual setting when we evaluate on French.
EN-Lit. refers to the Literature Domain. UGC refers to User-Generated Content. FR-Tran. refers to sentences translated from the English *In-Domain* test set, hence reducing the domain-gap to its minimum.
≥ Ref | < Ref | ≤ -2 points | ≤ -5 points

These results do not match the findings of Singh et al. (2019), who found no language alignment across layers, although they inspected Natural Language Inference, a more "high-level task" (Dagan et al., 2005a; Bowman et al., 2015a). This difference could be due to the different choice of similarity metric. Indeed, in contrast with the Canonical Correlation Analysis (Hotelling, 1992) used by Singh et al. (2019), the linear CKA used here, was shown to be robust to noise in the training procedure of deep-learning models

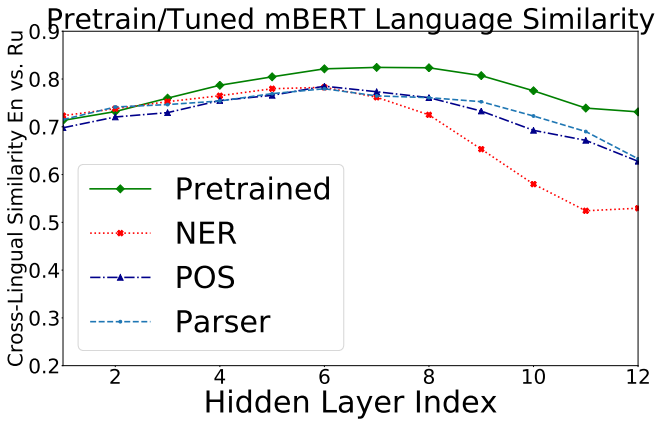


Figure 7.1: Cross-Lingual similarity (CKA) between representations of pretrained and fine-tuned models on POS, NER and Parsing between English and Russian.

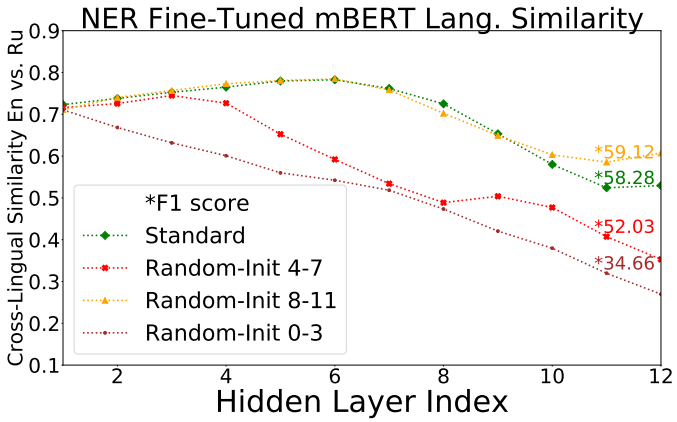


Figure 7.2: Cross-Lingual similarity (CKA) of the representations of a fine-tuned model on NER with and w/o RANDOM-INIT between English (source) and Russian (target). The higher the score the greater the similarity.

(Kornblith et al., 2019) and is therefore better suited to compare representations of deep-learning models.

7.6 Better Alignment Leads to Better Cross-Lingual Transfer

In the previous section we showed that fine-tuned models align the representations between parallel sentences, across languages. Moreover, we demonstrated that the lower part of the model is critical for cross-lingual transfer but hardly impacts the same-language

performance. In this section, we show that the alignment measured plays a critical role in cross-lingual transfer.

As seen in Figure 7.2 in the case of English to Russian (and in Figures 9.2-9.4 in the Appendix for other languages), when we randomly-initialize the lower part of the model, there is no alignment: the similarity between the source and target languages decreases. We observe the same trend for all other languages and tasks and report it in the Appendix in figures 9.2-9.4. This result matches the drop in cross-lingual performance that occurs when we apply RANDOM-INIT to the lower part of the model while impacting moderately same-language performance.

For a more systematic view of the link between the cross-lingual similarities and the cross-lingual transfer, we measure the Spearman correlation between the *cross-lang gap* (i.e the difference between the same-language perfromance and the cross-lingual performance) (Hu et al., 2020) and the cross-lingual similarity averaged over all the layers.

We report in table 7.4 the correlation between the hidden representation of each layer and the *cross-lang gap* between the source and the target averaged across all target languages and all layers. The correlation is strong and significant for all the tasks and for both the fine-tuned and the pretrained models. This shows that multilingual alignment that occurs within the models, learnt during pretraining is strongly related with cross-lingual transfer.

The values of this correlation per layer is reported in Figure 7.3. For the pretrained model, we observe the same distribution for each task with layer 6 being the most correlated to cross-lingual transfer. We observe large variations in the fine-tuned cases, the most notable being NER. This illustrates the task-specific aspect of the relation between cross-lingual similarity and cross-lingual transfer. More precisely, in the case of NER, the sharp increase and decrease in the upper part of the model provide new evidence that for this task, fine-tuning highly impacts the cross-lingual similarity in the upper part of the model which correlates with the cross-lingual transfer.

The cross-lingual similarity is computed on the pretrained and fine-tuned models (without random-initialization) on all the languages. We find that the cross-lingual similarity correlates significantly with the *cross-lang gap* for all three tasks, both on the

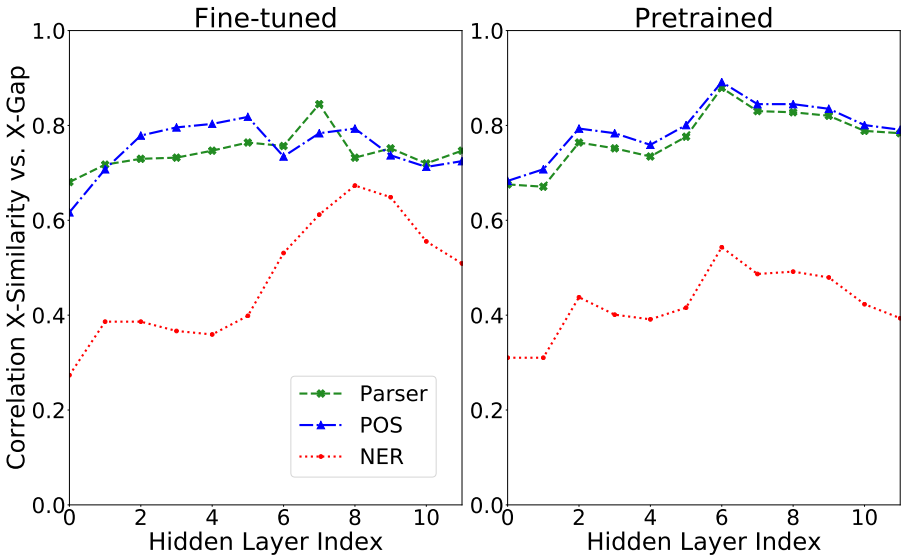


Figure 7.3: Spearman Correlation between Cross-Lingual Similarity (CKA between English and the target representations) and *cross-lang gap* averaged over all 17 target languages for each layer

Task	Cross-Lingual Gap vs. Cross-Lingual Similarity
Parsing	0.76
POS	0.74
NER	0.47

Table 7.4: Spearman-Rank Correlation between the cross-lingual gap and the cross-lingual similarity between the source and the target languages of the fine-tuned models and the pretrained model averaged over all the hidden layers and all the 17 target languages (sample size per task: 17). For NER, the cross-lingual gap is measured on wikiner data and not on the parrallel data itself in contrast with Parsing and POS tagging.

fine-tuned and pretrained models. The spearman correlation for the fine-tuned models are 0.76, 0.75 and 0.47 for parsing, POS and NER, respectively.[11]

7.7 Discussion

Understanding the behavior of pretrained language models is currently a fundamental challenge in NLP (Rogers et al., 2020). A popular approach consists of probing the intermediate representations with external classifiers (Alain and Bengio, 2017; Adi et al.,

[11] Correlations for both the pretrained and the fine-tuned models are reported in the Appendix Table 7.4.

2017; Conneau et al., 2018a) to measure if a specific layer captures a given property. Using this technique, Tenney et al. (2019) showed that BERT encodes linguistic properties in the same order as the "classical NLP pipeline". However, probing techniques only indirectly explain the behavior of a model and do not explain the relationship between the information captured in the representations and its effect on the task (Elazar et al., 2020). Moreover, recent works have questioned the usage of probing as an interpretation tool (Hewitt and Liang, 2019; Ravichander et al., 2020). This motivates our approach to combine a structural analysis based on representation similarity with behavioral analysis. In this regard, our findings extend recent work from Merchant et al. (2020) in the multilingual setting, who show that fine-tuning impacts mainly the upper layers of the model and preserves the linguistic features learned during pretraining. In our case, we show that the lower layers are in charge of aligning representations across languages and that this cross-lingual alignment learned during pretraining is preserved after fine-tuning.

In summary, we combined a *structural* analysis of the similarities between hidden representation across languages with a novel *behavioral* analysis that randomly-initialize the models' parameters to understand it. By combining those experiments on 17 languages and 3 tasks, we showed that mBERT is constructed from: (1) a multilingual encoder in the lower layers, which aligns hidden representations across languages and is critical for cross-lingual transfer, and (2) a task-specific, language-agnostic predictor that has little effect to cross-lingual transfer, in the upper layers. Additionally, we demonstrated that hidden cross-lingual similarity strongly correlates with downstream cross-lingual performance suggesting that this alignment is at the root of these cross-lingual transfer abilities. This shows that mBERT reproduces the standard cross-lingual pipeline described by Ruder et al. (2019) without any explicit supervision signal for it. Practically speaking, our findings provide a concrete tool to measure cross-lingual representation similarity that could be used to design better multilingual pretraining processes.

8 Modeling Unseen Languages with Multilingual Language Models

8.1 Motivation

Language-Model-based Zero-Shot Cross-lingual transfer setting analyzed in the chapter 7 is a challenging and important study case to push the multilingual modeling abilities of NLP systems. However, whenever we can evaluate a system on a test set (as we do in the zero-shot setting), it is usually possible to sample even a small amount of this test set for training purposes. For this reason, in practice, it is better to use as much training data in the language of interest (raw or annotated data) to train the best-performing system. This is the framework we work with in this chapter. This chapter is an adaptation of (Muller et al., 2020c,a, 2021a) papers. (Muller et al., 2021a) was done in collaboration with Antonis Anastasopoulos from George Mason University.

As Joshi et al. (2020b) vividly illustrate, there is a large divergence in the coverage of languages by NLP technologies. The majority of the 7000+ of the world's languages (cf. § 2.1.3) are not studied by the NLP community, since most have few or no annotated datasets, making systems' development challenging.

The development of such models is a matter of high importance for the inclusion of communities, the preservation of endangered languages and more generally to support the rise of tailored NLP ecosystems for such languages (Schmidt and Wiegand, 2017; Stecklow, 2018; Seddah et al., 2020). In that regard, the advent of the Universal Dependencies project (Nivre et al., 2016) and the WikiAnn dataset (Pan et al., 2017) have greatly

increased the number of covered languages by providing annotated datasets for more than 90 languages for dependency parsing and 282 languages for NER.

Regarding modeling approaches, the emergence of multilingual representation models, first with static word embeddings (discussed in § 4.7.2) and then with language model-based contextual representations (Devlin et al., 2019a; Conneau et al., 2020a) enabled transfer from high to low-resource languages, leading to significant improvements in downstream task performance (Rahimi et al., 2019; Kondratyuk and Straka, 2019). Furthermore, in their most recent forms, these multilingual models process tokens at the sub-word level (Kudo and Richardson, 2018b). As such, they work in an open vocabulary setting, only constrained by the pretraining character set. This flexibility enables such models to process any language, even those that are not part of their pretraining data.[1]

However, before this work along with concurrent related papers (Muller et al., 2020c; Pfeiffer et al., 2020), it was not clear how to use efficiently large-scale multilingual language models such as mBERT on languages that are not seen during the pretraining — referred to as *unseen* languages.

In this chapter, we analyze task and language adaptation experiments to get usable language model-based representations for unseen languages. We run experiments on 15 typologically diverse languages on three NLP tasks: part-of-speech (POS) tagging, dependency parsing (DEP) and named-entity recognition (NER).

Our results bring forth a diverse set of behaviors that we classify in three categories reflecting the abilities of pretrained multilingual language models to be used for low-resource languages. We dub those categories Easy, Intermediate and Hard.

Hard languages include both stable and endangered languages, but they predominantly are languages of communities that are mainly under-served by modern NLP. Hence, we direct our attention to these Hard languages.

For those languages, we show that the script they are written in can be a critical element in the transfer abilities of pretrained multilingual language models. Transliterating them leads to large gains in performance outperforming non-contextual strong baselines. In summary:

[1] As long as the script and characters of the target language (e.g. Hindi written in Devanagari characters) is part of the training data.

- We propose a new categorization of the low-resource languages that are unseen by available language models: the Hard, the Intermediate and the Easy languages.
- We show that Hard languages can be better addressed by transliterating them into a better-handled script (typically Latin), providing a promising direction towards making multilingual language models useful for a new set of unseen languages.

What can be done for unseen languages?

Unseen languages strongly vary in the amount of available data, in their script (many languages use non-Latin scripts such as Sorani Kurdish and Mingrelian), and in their morphological or syntactical properties (most largely differ from high-resource Indo-European languages). This makes the design of a single approach to build contextualized models for those languages challenging at best. In this work, by experimenting with 15 typologically diverse unseen languages, (i) we show that there is a diversity of behavior depending on the script, the amount of available data, and the relation to the pretraining languages; (ii) Focusing on the unseen languages that lag in performance compared to their easier-to-handle counterparts, we show that the script plays a critical role in the transfer abilities of multilingual language models. Transliterating such languages to a script which is used by a related language seen during pretraining.

8.2 Experimental Setting

We select a small portion of those languages within a large scope of language families and scripts. Our selection is constrained to 15 typologically diverse languages for which we have evaluation data for at least one of our three downstream tasks. Our selection includes low-resource Indo-European and Uralic languages, as well as members of the Bantu, Semitic, and Turkic families. None of these 15 languages are included in the pretraining corpora of mBERT. Information about their scripts, language families, and amount of available raw data can be found in Table 8.1.

Language (iso)	Script	Family	#sents	source	Category
Faroese (fao)	Latin	North Germanic	297K	(Biemann et al., 2007)	Easy
Mingrelian (xmf)	Georg.	Kartvelian	29K	Wikipedia	Easy
Naija (pcm)	Latin	English Pidgin	237K	(Caron et al., 2019)	Easy
Swiss German (gsw)	Latin	West Germanic	250K	OSCAR	Easy
Bambara (bm)	Latin	Niger-Congo	1K	OSCAR	Intermediate
Wolof (wo)	Latin	Niger-Congo	10K	OSCAR	Intermediate
Narabizi (nrz)	Latin	Semitic*	87K	(Seddah et al., 2020)	Intermediate
Maltese (mlt)	Latin	Semitic	50K	OSCAR	Intermediate
Buryat (bxu)	Cyrillic	Mongolic	7K	Wikipedia	Intermediate
Mari (mhr)	Cyrillic	Uralic	58K	Wikipedia	Intermediate
Erzya (myv)	Cyrillic	Uralic	20K	Wikipedia	Intermediate
Livvi (olo)	Latin	Uralic	9.4K	Wikipedia	Intermediate
Uyghur (ug)	Arabic	Turkic	105K	OSCAR	Hard
Sindhi (sd)	Arabic	Indo-Aryan	375K	OSCAR	Hard
Sorani (ckb)	Arabic	Indo-Iranian	380K	OSCAR	Hard

Table 8.1: Unseen Languages used for our experiments. #sents indicates the number of sentences used for training from scratch Monolingual Language Models as well as for MLM-TUNING mBERT
*code-mixed with French

8.3 The Three Categories of Unseen Languages

For each unseen language and each task, we experiment with our three modeling approaches: (a) **Training a language model from scratch on the available raw data** and then fine-tuning it on any available annotated data in the target language. (b) **Fine-tuning mBERT with TASK-TUNING** directly on the target language. (c) Finally, **adapting mBERT to the unseen language using MLM-TUNING** before fine-tuning it in a supervised way on the target language. We then compare all these experiments to our non-contextual strong baselines. By doing so, we can assess if language models are a practical solution to handle each of these unseen languages.

Interestingly, we find a large diversity of behaviors across languages regarding those language model training techniques. We observe three clear clusters of languages.

The first cluster, which we dub “Easy", corresponds to the languages that do not require extra MLM-TUNING for mBERT to achieve good performance. mBERT has the modeling abilities to process such languages without relying on raw data and can outperform strong non-contextual baselines as such. In the second cluster, the “Intermediate"

languages require MLM-tuning. mBERT is not able to beat strong non-contextual baselines using only Task-tuning, but MLM-tuning enables it to do so. Finally, Hard languages are those on which mBERT fails to deliver any decent performance even after MLM- and Task- fine-tuning. mBERT simply does not have the capacity to learn and process such languages.

We emphasize that our categorization of unseen languages is only based on the relative performance of mBERT after fine-tuning compared to strong non-contextual baseline models. We leave for future work the analysis of the absolute performance of the model on such languages (e.g. analysing the impact of the fine-tuning data set size on mBERT's downstream performance).

In this section, we present our results in detail in each of these language clusters and provide insights into their linguistic properties.

8.3.1 Easy

Languages	mBERT	mBERT+MLM	MLM	Baseline
		UPOS		
Faroese	96.3	**96.5**	91.1	95.4
Naija	89.3	**89.6**	87.1	89.2
Swiss German	76.7	**78.7**	65.4	75.2
		LAS		
Faroese	84.0	**86.4**	67.6	83.1
Naija	**71.5**	69.2	63.0	68.3
Swiss German	41.2	**69.6**	30.0	32.2
		NER		
Faroese	52.1	**58.3**	39.3	44.8
Mingrelian	53.6	**68.4**	42.0	48.2

Table 8.2: **Easy Languages** POS, Parsing and NER scores comparing mBERT, mBERT+MLM and monolingual MLM to strong non-contextual baselines when trained and evaluated on unseen languages. Easy Languages are the ones on which mBERT outperforms strong baselines out-of-the-box. Baselines are LSTM based models from UDPipe-future (Straka, 2018) for parsing and POS tagging and Stanza (Qi et al., 2020b) for NER.

Easy languages are the ones on which mBERT delivers high performance out-of-the-box, compared to strong baselines. We classify Faroese, Swiss German, Naija and Mingrelian as easy languages and report performance in Table 8.2.

We find that those languages match two conditions:

- They are closely related to languages used during MLM pretraining
- These languages use the same script as their closely related languages.

Such languages benefit from multilingual models, as cross-lingual transfer is easy to achieve and hence quite effective.

8.3.2 Intermediate

Language	mBERT	mBERT+MLM	MLM	Baseline
		UPOS		
Maltese	92.0	**96.4**	92.0	96.0
Narabizi	81.6	**84.2**	71.3	**84.2**
Bambara	90.2	**92.6**	78.1	92.3
Wolof	92.8	**95.2**	88.4	94.1
Erzya	89.3	**91.2**	84.4	91.1
Livvi	83.0	**85.5**	81.1	84.1
		LAS		
Maltese	74.4	**82.1**	66.5	79.7
Narabizi	56.5	**57.8**	41.8	52.8
Bambara	71.8	75.4	46.4	**76.2**
Wolof	73.3	**77.9**	62.8	77.0
Erzya	61.2	**66.6**	47.8	65.1
Livvi	36.3	**42.3**	35.2	40.1
		NER		
Maltese	61.2	**66.7**	62.5	63.1
Mari	55.2	**57.6**	44.0	56.1

Table 8.3: **Intermediate Languages** POS, Parsing and NER scores comparing mBERT, mBERT+MLM and monolingual MLM to strong non-contextual baselines when trained and evaluated on unseen languages. Intermediate Languages are the ones for which mBERT requires MLM-tuning to outperform the baselines.

The second type of languages (which we dub "Intermediate") are generally harder to process for pretrained MLMs out-of-the-box. In particular, pretrained multilingual language models are typically outperformed by a non-contextual strong baselines. Still, MLM-TUNING has an important impact and leads to usable state-of-the-art models.

A good example of such an intermediate language is Maltese, a member of the Semitic language but using the Latin script. Maltese has not been seen by mBERT during pretraining. Other Semitic languages though, namely Arabic and Hebrew, have been included in the pretraining languages. As seen in Table 8.3, the non-contextual baseline outperforms mBERT. Additionally, a monolingual MLM trained on only 50K sentences matches mBERT performance for both NER and POS tagging. However, the best results are reached with MLM-TUNING: the proper use of monolingual data and the advantage of similarity to other pretraining languages render Maltese a *tackle-able* language as shown by the performance gain over our strong non-contextual baselines.

Our Maltese dependency parsing results are in line with those of Chau et al. (2020), who also showed that MLM-TUNING leads to significant improvements. They also additionally showed that a small vocabulary transformation allowed fine-tuning to be even more effective and gain 0.8 LAS points more.

We also categorize Bambara, a Niger-Congo Bantu language spoken in Mali and surrounding countries, as Intermediate, relying mostly on the POS tagging results which follow similar patterns as Maltese and Narabizi. We note that the BambaraBERT that we trained achieves notably poor performance compared to the non-contextual baseline, a fact we attribute to the extremely low amount of available data (1000 sentences only). We also note that the non-contextual baseline is the best performing model for dependency parsing, which could also potentially classify Bambara as a "Hard" language instead.

Our results in Wolof follow the same pattern. The non-contextual baseline achieves a 77.0 in LAS outperforming mBERT. However, MLM-TUNING achieves the highest score of 77.9.

8.3.3 HARD

The last category of the hard unseen language is perhaps the most interesting one, as these languages are very hard to process. mBERT is outperformed by non-contextual baselines

Language	mBERT	mBERT+MLM	MLM	Baseline
		UPOS		
Uyghur	77.0	77.3	87.4	**90.0**
		LAS		
Uyghur	45.5	48.9	57.3	**67.9**
		NER		
Uyghur	24.3	34.6	41.4	**53.8**
Sindhi	42.3	47.9	45.2	**51.4**
Sorani Kurdish	70.4	75.6	80.6	**80.5**

Table 8.4: **Hard Languages** POS, Parsing and NER scores comparing mBERT, mBERT+MLM and monolingual MLM to strong non-contextual baselines when trained and evaluated on unseen languages. Hard Languages are the ones for which mBERT fails to reach decent performance even after MLM-tuning.

as well as by monolingual language models trained from scratch on the available raw data. At the same time, MLM-tuning on the available raw data has a minimal impact on performance.

Uyghur, a Turkic language with about 10-15 million speakers in central Asia, is a prime example of a hard language for current models. In our experiments, outlined in Table 8.4, the non-contextual baseline outperforms all contextual variants, both monolingual and multilingual, in all the tasks with up to 20 points difference compared to mBERT for parsing. Additionally, the monolingual UyghurBERT trained on only 105K sentences outperforms mBERT even after MLM-tuning.

We attribute this discrepancy to script differences: Uyghur uses the Perso-Arabic script, when the other Turkic languages that were part of mBERT pretraining use either the Latin (e.g. Turkish) or the Cyrillic script (e.g. Kazakh).

Sorani Kurdish (also known as Central Kurdish) is a similarly hard language, mainly spoken in Iraqi Kurdistan by around 8 million speakers, which uses the Sorani alphabet, a variant of the Arabic script. We can solely evaluate on the NER task, where the non-contextual baseline and the monolingual SoraniBERT perform similarly around 80.5 F1-score outperforming significantly mBERT which only reaches 70.4 in F1-score. MLM-tuning on 380K sentences of Sorani texts improves mBERT performance to 75.6 F1-score, but it is still lagging behind the baseline. Our results in Sindhi follow the same pattern. The non-contextual baseline achieves a 51.4 F1-score outperforming with a large

Rabi m3akom et **bon courage** wled bledi	Narabizi
God bless you and good luck children of my country	Translation

Table 8.5: Example of a Narabizi sentence code-mixed with French (French words have been written in bold)

margin our language models (a monolingual SindhiBERT achieves an F1-score of 45.2, and mBERT is worse at 42.3).

8.4 Case Study on Narabizi an Unseen Code-Mixed and non-standard Dialect

We focus here on North-African dialectal Arabic in its Algerian form, understood and spoken by over 40 million people in the Maghreb (Sayahi, 2014). In its written form, it is mostly found online and in Latin script, with a high degree of variability across writers and a high degree of code-switching with French. We refer to this North-African Arabic dialect, which does not belong to mBERT's pre-training corpus, as *North-African Arabizi* or Narabizi following (Seddah et al., 2020).

Narabizi is non-standard so the spelling of words varies a lot based on who is writing as well as on socio-geographical context. As illustrated in table 8.5 (from (Seddah et al., 2020)), a single word in French may corresponds to multiple spellings. As reported in Table 8.3, for both POS tagging and parsing, the multilingual models outperform the monolingual BERT. In addition, MLM-tuning leads to significant improvements over the non-language-tuned mBERT baseline, also outperforming the non-contextual dependency parsing baseline. For this reason, Narabizi is classified as a Intermediate language for mBERT.

Gloss	Observed forms	Lang
why	wa3lach w3alh 3alach 3lache	Narabizi
all	ekl kal kolach koulli kol	Narabizi
many	beaucoup boucoup bcp	French

Table 8.6: Examples of lexical variation in Narabizi (Seddah et al., 2020)

	mBERT		mBERT+MLM	
Source Language	POS	UAS	POS	UAS
Maltese	**35.13**	**40.04**	38.94	42.32
French	33.12	38.54	**47.32**	**43.77**
English	30.67	32.40	44.59	38.83
Arabic (MSA)	16.55	28.23	28.08	34.34
Vietnamese	16.92	13.98	23.21	14.44
	81.60	66.84	82.61	67.12
		Baselines		
	Stanza			
Narabizi	84.20	52.84		
French	27.00	33.74		
	Rule-Based			
Baseline	20.49	18.71		

Table 8.7: Cross-lingual performance on the test set (5 seeds). The baselines is a majority class prediction for POS tagging and the left-tokens head prediction for parsing.

8.4.1 Zero-shot cross-lingual transfer to Narabizi

To analyse in a more refined way how mBERT performs well after adaptation on Narabizi, we experiment with Narabizi in the zero-shot cross-lingual transfer. In this setting, mBERT and mBERT+MLM are fine-tuned in a task-specific way on a source language before being evaluated on Narabizi data.

We study this cross-lingual transfer along three independent directions: how related is the source language to Narabizi, whether it uses the same script, and whether it is present in mBERT's training corpus. We expect transfer to perform better when the source language is closely related to the target language, when it uses the same script, and when it is known to mBERT. In decreasing order of relatedness, we use Modern Standard Arabic (closely related), Maltese (distantly related, see Habash, 2010 and Čéplö et al., 2016), French (used for code-switching), English and Vietnamese. Among those source languages, Modern Standard Arabic is written in a different script (the Arabic script), whereas Maltese is the only language unknown to mBERT.[2]

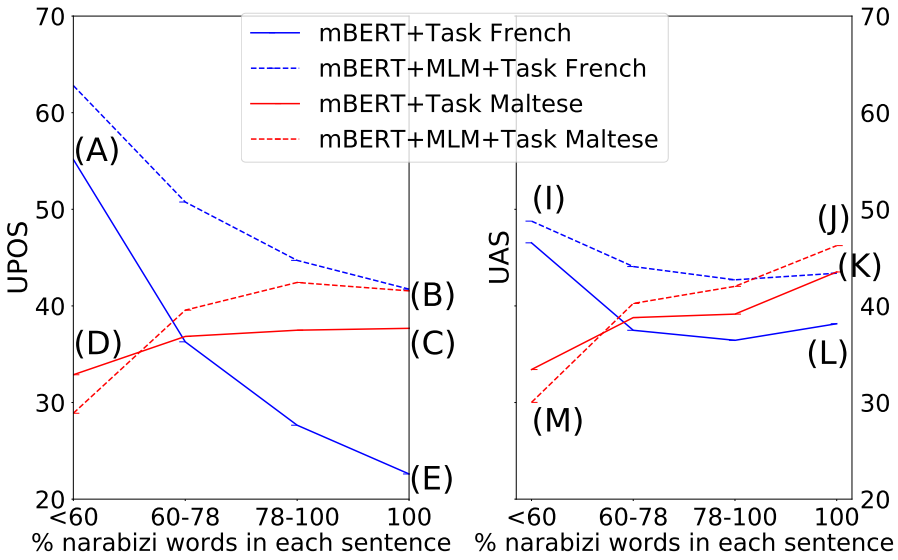


Figure 8.1: Performance as a function of the code-mixing rate, reported on train set to have enough data (5 seeds, no annotated training data seen during fine-tuning). (X) markers commented in sec. 8.4.2.

8.4.2 Impact of code-mixing

Our hypothesis is that the high level of transfer when the source is French is due to the frequency of code-mixing with French in Narabizi. To validate this hypothesis, we compute the performance of the model with respect to the code-mixing ratio (see Figure 8.1). We split the dataset into four buckets of around 25% of the full dataset, according to the ratio of native Narabizi vs. French tokens in each sentence, and compare French and Maltese as source languages. On sentences that have 100% Narabizi tokens, mBERT trained on French performs poorly (in Figure 8.1, cf. mark (E) for POS and (L) for parsing). On sentences that include at least 40% of French tokens, scores reach 54% for POS tagging (cf. (A)) and 47% for parsing (cf. (I)). Moreover, for French, mBERT+MLM leads to an impressive 21.2% error reduction compared to mBERT for POS tagging (33.12 vs. 47.32) and an 8.5% error reduction for parsing (cf. Table 8.7). We observe in Fig. 8.1 (cf. (B) and (K)) that this improvement mostly comes from a better accuracy on Narabizi tokens. Interestingly, unsupervised fine-tuning leads to closing the gap on native Narabizi tokens between models tuned on French and Maltese (+15:

[2]We sample the training datasets to have 1,200 sentences for each source language. We pick the first 1,200 training sentences. More information on the datasets used is given in Appendix 8.4.1.

(B)-(E) vs. +2.4: (B)-(C) for tagging, +5.6: (K)-(L) vs. +2.2: (J)-(K) for parsing). This shows that unsupervised fine-tuning can overcome the lexical divergence between distant languages such as native Narabizi and French.

8.4.3 Transfer between unseen languages

Surprisingly, mBERT tuned on Maltese performs well, with the best performance among mBERT models for both POS tagging and parsing. It outperforms in the zero-shot scenario by 5 points in tagging and 6 points in parsing. As seen in Figure 8.1 (C) and (J), it performs the best on native Narabizi sentences (with no code-mixing). This result is surprising as Maltese is absent from the pre-training corpora. It shows that mBERT is able to capture *structural properties* shared by related languages even if they are absent from the pre-training corpora, which extends observations made by Wang et al. (2019b).

8.4.4 Is the multilingualism of mBERT at play?

Finally, we want to show that the ability of mBERT to achieve cross-lingual transfer is related to the 104 languages it is pre-trained on, rather than because a pre-trained Transformer is an inherently good POS tagger or parser. To do so, we compare mBERT with three other models: Roberta, the optimized English version of BERT (Liu et al., 2019), CamemBERT (introduced in chapter 5), and a randomly initialized mBERT-like Transformer as a baseline (noted Random).

	mBERT		RoBERTA		CamemBERT		Random	
	POS	UAS	POS	UAS	POS	UAS	POS	UAS
French	**33.12**	**38.54**	29.75	27.41	31.77	32.55	30.29	25.30
Maltese	**35.13**	**40.04**	25.45	17.27	31.62	34.65	19.81	19.04

Table 8.8: Zero-shot transfer from French and Maltese to Narabizi. 5 averaged seeds.

Focusing our analysis on French and Maltese, we observe (cf.8.8) that mBERT is the model that leads to the most successful transfer in both cases and for both tasks, by a very

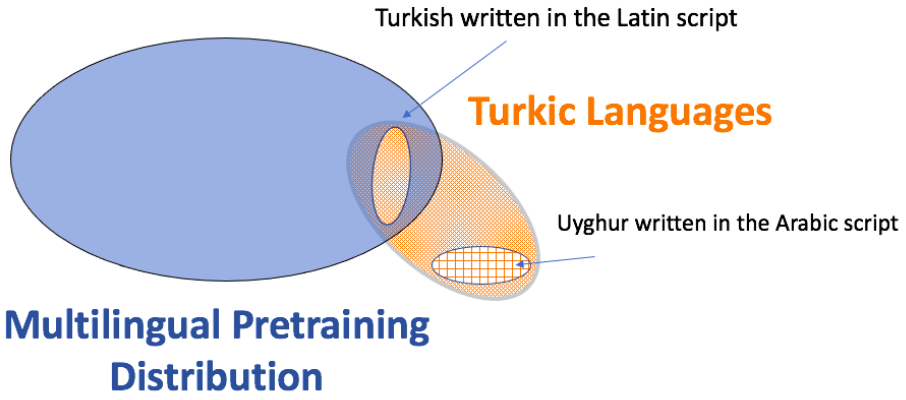


Figure 8.2: An illustration of the pretraining distributions and an unseen language distribution in the case of the Turkic Language Family. Uyghur is unseen but related to Turkish which mBERT has been pretrained on. Uyghur is written in the Arabic script while Turkish is written in the Latin Script making it a tough challenge for mBERT

large margin in the case of Maltese. This shows that pretraining on such a diversity of languages is at the core of the transfer to Narabizi.

8.5 Tackling Hard Languages with Multilingual Language Models

Our intermediate Uralic language results provide initial supporting evidence for our argument on the importance of having pretrained LMs on languages with similar scripts, even for generally high-resource language families. Our hypothesis is that the script is a key element for language models to correctly process unseen languages.

To test this hypothesis, we assess the ability of mBERT to process an unseen language after transliterating it to another script present in the pretraining data. We experiment on six languages belonging to four language families: Erzya, Bruyat and Meadow Mari (Uralic), Sorani Kurdish (Iranian, Indo-European), Uyghur (Turkic) and Mingrelian (Kartvelian). We apply the following transliterations:

- Erzya/Buryat/Mari: Cyrillic → Latin Script
- Uyghur: Arabic Script → Latin Script
- Sorani: Arabic Script → Latin Script

Model	POS	LAS	NER	Model	NER
	Uyghur (Arabic→Latin)			Sorani (Arabic→Latin)	
UyghurBERT	87.4→86.2	57.3→54.6	41.4→41.7	SoraniBERT	80.6→78.9
mBERT	77.0→87.9	45.7→65.0	24.3→35.7	mBERT	70.5→77.8
mBERT+MLM	77.3→**89.8**	48.9→**66.8**	34.7→**55.2**	mBERT+MLM	75.6→**82.7**
	Buryat (Cyrillic→Latin)			Meadow Mari (Cyrillic→Latin)	
BuryatBERT	75.8→75.8	31.4→31.4	–	MariBERT	44.0→45.5
mBERT	83.9→81.6	50.3→45.8	–	mBERT	55.2→58.2
mBERT+MLM	**86.5**→84.6	**52.9**→51.9	–	mBERT+MLM	57.6→**65.9**
	Erzya (Cyrillic→Latin)			Mingrelian (Georgian→Latin)	
ErzyaBERT	84.4→84.5	47.8→47.8	–	MingrelianBERT	42.0→42.2
mBERT	89.3→88.2	61.2→58.3	–	mBERT	53.6→41.8
mBERT+MLM	**91.2**→90.5	**66.6**→65.5	–	mBERT+MLM	**68.4**→62.6

Table 8.9: Transliterating low-resource languages into the Latin script leads to significant improvements in languages like Uyghur, Sorani, and Meadow Mari. For languages like Erzya and Buryat transliteration, does not significantly influence results, while it does not help for Mingrelian. In all cases, mBERT+MLM is the best approach.

- Mingrelian: Georgian Script → Latin Script

8.5.1 Linguistically-motivated transliteration

The strategy we used to transliterate the above-listed language is specific to the purpose of our experiments. Indeed, our goal is for the model to take advantage of the information it has learned during training on a related language written in the Latin script. The goal of our transliteration is therefore to transcribe each character in the source script, which we assume corresponds to a phoneme, into the most frequent (sometimes only) way this phoneme is rendered in the closest related language written in the Latin script, hereafter the target language. This process is not a transliteration strictly speaking, and it needs not be reversible. It is not a phonetization either, but rather a way to render the source language in a way that maximizes the similarity between the transliterated source language and the target language.

We have manually developed transliteration scripts for Uyghur and Sorani Kurdish[3], using respectively Turkish and Kurmanji Kurdish as target languages, only Turkish being one of the languages used to train mBERT. Note however that Turkish and Kurmanji

[3] Transliterations script are available at https://github.com/benjamin-mlr/mbert-unseen-languages

Kurdish share a number of conventions for rendering phonemes in the Latin script (for instance, /ʃ/, rendered in English by "sh", is rendered in both languages by "ş"; as a result, the Arabic letter "ش", used in both languages, is rendered as "ş" by both our transliteration scripts). As for Erzya, Buryat and Mari, we used the readily available transliteration package *transliterate*,[4] which performs a standard transliteration.[5] We used the Russian transliteration module, as it covers the Cyrillic script. Finally, for our control experiments on Mingrelian, we used the Georgian transliteration module from the same package.

8.5.2 Transfer via Transliteration

We train mBERT with MLM-tuning and Task-tuning as well as monolingual BERT model trained from scratch on the transliterated data. We also run controlled experiments on high-resource languages written in the Latin script on which mBERT was pretrained on, namely Arabic, Japanese and Russian (reported in Table 8.10).

Our results with and without transliteration are listed in Table 8.9. Transliteration for Sorani and Uyghur has a noticeable positive impact. For instance, transliterating Uyghur to Latin leads to an improvement of 16 points in parsing and 20 points in NER. For one of the low-resource Uralic languages, Meadow Mari, we observe an 8 F1-score points improvement on NER, while for other Uralic languages like Erzya the effect of transliteration is very minor. The only case where transliteration to the Latin script leads to a drop in performance for mBERT and mBERT+MLM is Mingrelian.

We interpret our results as follows. When running MLM-tuning and Task-tuning, mBERT associates the target unseen language to a set of similar languages seen during pretraining based on the script. In consequence, mBERT is not able to associate a language to its related language if they are not written in the same script. For instance, transliterating Uyghur enables mBERT to match it to Turkish, a language which accounts for a sizable portion of mBERT pretraining. In the case of Mingrelian, transliteration has the opposite effect: transliterating Mingrelian in the Latin script is harming the performance

[4] https://pypi.org/project/transliterate/

[5] In future work, we intend to develop dedicated transliteration scripts using the strategy described above, and to compare the results obtained with it with those described here.

	Original Script → Latin Script		
Model	POS	LAS	NER
Arabic	96.4 → 94.9	82.9 → 78.8	87.8 → 80.9
Russian	98.1 → 96.0	88.4 → 84.5	88.1 → 86.0
Japanese	97.4 → 95.7	88.5 → 86.9	61.5 → 55.6

Table 8.10: mBERT Task-Tuned on high resource languages for POS tagging, parsing and NER. We compare fine-tuning done on data written the original language script with fine-tuning done on Latin transliteration. In all cases, transliteration degrades downstream performance.

as mBERT is not able to associate it to Georgian which is seen during pretraining and uses the Georgian script.

This is further supported by our experiments on high resource languages (cf. table 8.10). When transliterating pretrained languages such as Arabic, Russian or Japanese, mBERT is not able to compete with the performance reached when using the script seen during pretraining. Transliterating the Arabic script and the Cyrillic script to Latin does not automatically improve mBERT performance as it does for Sorani, Uyghur and Meadow Mari. For instance, transliterating Arabic to the Latin script leads to a drop in performance of 1.5, 4.1 and 6.9 points for POS tagging, parsing and NER respectively.

Our findings are generally in line with previous work. Transliteration to English specifically (Lin et al., 2016; Durrani et al., 2014b) and named entity transliteration (Kundu et al., 2018; Grundkiewicz and Heafield, 2018) has been proven useful for cross-lingual transfer in tasks like NER, entity linking (Rijhwani et al., 2019), morphological inflection (Murikinati et al., 2020), and Machine Translation (Amrhein and Sennrich, 2020).

The transliteration approach provides a viable path for rendering large pretrained models like mBERT useful for all languages of the world. Indeed, as reported in Table 8.9, transliterating both Uyghur and Sorani leads to matching or outperforming the performance of non-contextual strong baselines and deliver usable models (e.g. +12.5 POS accuracy in Uyghur).

8.6 Discussion

Pretraining ever larger language models is a research direction that has been receiving a lot of attention and resources from the NLP research community in the past three years (Devlin et al., 2019a; Raffel et al., 2019; Brown et al., 2020). Still, a large majority of human languages are under-resourced making the development of monolingual language models very challenging in those settings. Another path is to build large scale multilingual language models.[6]

However, such an approach faces the inherent zipfian structure of human languages and the "curse of multilinguality"[7] making the training of a single model to cover all languages an unfeasible solution (Conneau et al., 2020b). Reusing large scale pretrained language models for new unseen languages seems to be a more promising and reasonable solution from a cost-efficiency and environmental perspective (Strubell et al., 2019).

Pfeiffer et al. (2020) proposed to use adapter layers (Houlsby et al., 2019) to build parameter-efficient multilingual language models for unseen languages. However, this solution brings no significant improvement in the supervised setting, compared to a more simple Masked-Language Model finetuning. Furthermore, developing a language agnostic adaptation method is an unreasonable wish with regard to the large typological diversity of human languages.

On the other hand, the promising vocabulary adaptation technique of Chau et al. (2020) which leads to good dependency parsing results on unseen languages when combined with task-tuning has so far been tested only on Latin script languages (Singlish and Maltese). We expect that it will be orthogonal to our transliteration approach. Pfeiffer et al. (2021) was able to extend efficiently multilingual language models to scripts that are not supported during the pretraining.

In this context, we bring empirical evidence to assess the efficiency of language models pretraining and adaptation methods on 15 low-resource and typologically diverse unseen languages. Our results show that the "Hard" languages are currently out-of-the-scope

[6] Even though we explore a different research direction, recent advances in small scale and domain specific language models suggest such models could also have an important impact for those languages (Micheli et al., 2020).

[7] That states that scaling the number of languages require to scale the number of parameters to keep the same level of performance.

of any currently available language models and are therefore left outside of the current NLP progress. By focusing on those, we find that this challenge is mostly due to the script. Transliterating them to a script that is used by a related higher resource language on which the language model has been pretrained on leads to large improvements in downstream performance. Our results shed some new light on the importance of the script in multilingual pretrained models. While previous work suggests that multilingual language models could transfer efficiently across scripts in zero-shot settings (Pires et al., 2019b; Karthikeyan et al., 2019), our results show that this cross-script transfer is possible only if the model has seen related languages in the same script during pretraining.

9 Conclusion

To conclude, we summarize the main contributions presented in this thesis before drawing future research directions.

9.1 Summary

9.1.1 Building Transformers Masked Language Models

We presented our work (cf. Chapter 5) on building and analyzing the performance of a transformer-based Masked-Language Model for French. This work was done in collaboration with Louis Martin and Pedro Ortiz under the guidance of our supervisors. As our main contribution, we showed that building a state-of-the-art model was possible with only a few gigabytes of raw data and that heterogeneous web data is superior to homogeneous domains like Wikipedia for pretraining. These observations were concurrently made for sequence to sequence models in English by Raffel et al. (2019) and extended with models trained on an even smaller corpora (Micheli et al., 2020). Additionally, we found that the number of pretraining steps required to reach optimal performance varies across tasks. Less complex tasks such as NER and dependency parsing need much fewer pretraining steps than natural language inference. Finally, based on our model, we improved the state-of-the-art performance on four downstream tasks in French.

9.1.2 Domain Gap Reduction: Lexical Normalization for Social Media Data

We then addressed lexical normalization for User Generated Content (UGC) in English. UGC is inherently different from edited text on which most NLP models are usually

trained (e.g., Wikipedia text, News text, etc.). In Chapter 6, we enhanced BERT's architecture to perform lexical normalization. For this purpose, we reframed word-level lexical normalization as a subword classification task. Given word-level normalization data (Baldwin et al., 2015), and despite the small amount of data available (only about two thousand sentences), we enhanced BERT's architecture and fine-tuned it to perform subword-level normalization after aligning the non-standard tokens and the standard ones. We showed that our system was competitive with state-of-the-art feature-rich systems like MoNoise (van der Goot, 2019). However, based on recent results described in (van der Goot et al., 2021), it is established that accurate lexical normalization leads to only moderate improvement in downstream dependency parsing on noisy social media data. On the one hand, this suggests that language models based on subword level tokenization are robust to lexical variability (Riabi et al., 2021; Itzhak and Levy, 2022). On the other hand, this shows the limit of lexical normalization to cope with the variability of UGC data and the need for direct adaptation techniques.

9.1.3 Explaining the Zero-shot Cross-Lingual Transfer Abilities of mBERT

Without any adaptation, large-scale multilingual language models can transfer across languages in the zero-shot setting — i.e., without using any supervised signal for the target language for the task of interest. To explain this behavior, we combined a behavioral analysis of mBERT with a structural analysis. This work was done in collaboration with Yanai Elazar. We introduced an analysis technique called Random-init, which consists of selectively randomly-initializing specific layers of mBERT before fine-tuning it. This technique allowed us to disentangle what is learned during pretraining from what is learned during fine-tuning. Based on Random-init, we showed that the lower layers of mBERT are critical for zero-shot cross-lingual transfer while there are not for same-language generalization. Looking at the hidden states, we showed that the model aligns representations across different languages in those specific lower layers. Overall, we showed that mBERT is schematically composed of two parts: the lower part is critical for zero-shot cross-lingual transfer and aligns representations across different languages,

while the upper part is language-agnostic and task-specific and can be randomly initialized before fine-tuning.

We note that Random-init is not limited to understanding the cross-lingual transfer of BERT-like models. It can be used to disentangle pretraining from fine-tuning and locate what layers contribute to downstream performance for any deep-learning models such that BERT-like models (Devlin et al., 2018a), fine-tuned sequence to sequence models (Lewis et al., 2020; Raffel et al., 2019), and in-context instruction training (Sanh et al., 2021a; Wang et al., 2022).

9.1.4 Handling Unseen Languages with mBERT

Finally, to achieve the practical goal of building the best models we could for low-resource languages, we studied how to make the best language models for languages that are not seen during the pretraining of available large-scale models (monolingual and multilingual). We refer to those languages as *unseen*. We showed that for most unseen languages, even a small amount of raw and annotated data is enough to fine-tune mBERT and outperforms strong non-contextual baselines. However, for a small number of languages like Uyghur or Sorani Kurdish, we find that this simple approach does not work. In those cases, we find that the script is usually the reason for this failure and that doing linguistically-motivated transliteration boosts the performance very significantly.

9.2 Future Directions

We will start with general observations about important research directions for the future given the current state of NLP (9.2.1 to 9.2.3). I will then describe research directions that have emerged from the work done during this thesis and that I aspire to work on in the future.

9.2.1 Scaling Transformers

As discussed in the introduction, one of the leading driving forces of empirical progress in NLP has been scaling the size of the models, pretraining data, and computing power (Kaplan et al., 2020). Very recently, the centi-billion parameters models like GPT-3

(Brown et al., 2020), OPT (Zhang et al., 2022), Megatron-LM (Smith et al., 2022), PaLM (Chowdhery et al., 2022) and BLOOM[1] have shown increasingly good performance in the zero-shot and few-shot setting for a vast number of tasks (Srivastava et al., 2022; Wei et al., 2022). Without a doubt, this scaling trend will keep delivering empirical progress on most benchmarks. It will also drive performance improvement in the cross-lingual setting, specifically for low-resource languages.

Making Large-scale models useable Beyond their substantial pretraining cost, those models are also very costly and slow at inference time. As of today, this is an essential limiting factor in their practical use. Therefore, one crucial research direction is to make them faster at inference time. Beyond progress in hardware (Jouppi et al., 2021), knowledge distillation (Buciluundefined et al., 2006; Hinton et al., 2015; Sanh et al., 2019) and pruning techniques, which aim at making models smaller (LeCun et al., 1989; Lagunas et al., 2021), are two promising paths that can make those models smaller and faster without harming too much their performance.

9.2.2 Controllability

Large-scale generative pretrained models (Brown et al., 2020; Zhang et al., 2022; Chowdhery et al., 2022) can potentially generate any textual data regardless of how harmful, factually incorrect, or degenerated the predicted sequence may be (Weidinger et al., 2021a; Bender et al., 2021). In the recent literature, this has been addressed by modeling prior information into the model (Martin et al., 2020; Dathathri et al., 2020) or by integrating humans in the learning process (referred to as the human-in-the-loop approach) (Ouyang et al., 2022). In the future, controlling the generated sequence will become more critical. We, again, note that most of the research done on controllability has been done in English. Generalizing those findings to other languages will also be an important endeavor.

9.2.3 Evaluation of Generative Models

Large-scale generative pretrained models (Brown et al., 2020; Zhang et al., 2022; Chowdhery et al., 2022) can potentially be used for any NLP task that can be framed in a

[1] https://huggingface.co/blog/bloom-megatron-deepspeed

"text-to-text" manner (Raffel et al., 2019). However, evaluating generative systems is inherently more challenging than classification or structured prediction systems (the output space being infinite). In work done during an internship, we experimented with generative Question Answering in a multilingual setting. We showed that BLEU or ROUGE scores were not correlated with human judgment (Muller et al., 2021c) in open domain question answering.

Evaluating these powerful generative models in a more refined way has been recently addressed with very large benchmarks such as the BIG-Bench benchmark (Srivastava et al., 2022), Cross-FIT (Ye et al., 2021) and FLEX (Bragg et al., 2021). Those benchmarks are designed to probe various natural language abilities and cross-task transfer. Even when tackling model classification tasks, defining the output space in a way compatible with the model is not trivial. This step is referred to as *verbalization* (Tam et al., 2021). In summary, evaluating all the dimensions of the prediction of a generative model is an important research direction. Again, designing a multilingual evaluation benchmark will also be important to ensure that the largest number of languages is supported.

9.2.4 Collecting Data for the Largest Number of Languages

Annotated Data An essential piece to evaluate NLP systems is to have annotated data. While there are a plethora of benchmarks in English, most languages are still left-out (Joshi et al., 2020b). As shown with our work on Narabizi (Seddah et al., 2020), annotating data for low-resource languages is costly and challenging. Indeed, crowd-sourcing platforms only have limited number of workers speaking those languages. Consequently, designing more efficient data annotation and curation techniques for low-resource languages is of first importance.

Raw Data Another essential piece for scaling the size of language models is how large the pretraining dataset needs to be (Hoffmann et al., 2022), as predicted by the scaling laws of neural language models (Kaplan et al., 2020).

We note that this statement contrasts with our findings on building the CamemBERT model (chapter 5). Indeed, we showed that pretraining a model on as little as 4GB of

Web Crawled data (from the OSCAR corpus (Ortiz Suárez et al., 2019)) was competitive with a model trained on 138GB of data. In that work, we were experimenting with fixed model sizes so our findings are consistent with the scaling laws.

Scaling pretraining datasets has been done successfully for English (Brown et al., 2020; Chowdhery et al., 2022) and several other languages thanks to the release of large-scale corpora in multiple languages such as the OSCAR corpus (Ortiz Suárez et al., 2019) and the mC4 corpus (Xue et al., 2021). However, replicating it for many other languages is much more tricky. Indeed, first, 90% of languages have no online presence (Kornai, 2013) making the collection of large corpora of raw text much more costly. Second, many of these languages have very little open-source textual data online.[2] Third, language identification is still a challenging problem for many languages (Siddhant et al., 2022). Finally, only about 60% of languages are used in a written form (according to Eberhard and Fennig (2021)). For this reason, the only way to collect data for such languages would be to work on speech utterances. In our work (Seddah et al., 2020), we experienced these challenges in collecting data based on Common Crawl for a North-African Arabic dialect. Indeed, based on a language identifier for this dialect, we could only extract 100k sentences from the entire Common Crawl dump. Another approach to collecting raw data for low-resource languages is to generate synthetic data. Given the increasing quality of generated data, large-scale multilingual generative language models, if prompted accordingly, could be used for this purpose.

9.2.5 From Cross-Lingual Adaptation to Cross-Cultural Adaptation

As explored for this thesis, large-scale multilingual language models exhibit surprising and efficient zero-shot cross-lingual transfer abilities. This behavior emerges from pretraining on a large quantity of raw data in multiple languages without explicit alignment across languages. However, cross-lingual transfer usually implicitly makes two hypotheses: (i) On the one hand, an *intra-language cultural homogeneity* hypothesis is made. On the other hand, (ii) we assume *inter-language cultural compatibility*. In more detail:

[2] Many languages are used online on social media, which is not accessible by open-source API (e.g., Facebook).

(i) **Intra-language cultural homogeneity**: When we experiment with cross-lingual transfer, we usually disregard language variations inside a given language, specifically from a cultural standpoint. This is particularly important for models that may directly impact what content users see online, like Hate-speech detection models or Question Answering models. For instance, models built for English are usually created using data collected with a US or UK-centric approach (Faisal et al., 2022). It is likely that those systems used in Nigeria or India in the English language would not generalize well or potentially carry cultural biases (Laaksonen et al., 2020; Hovy and Yang, 2021).
(ii) **Inter-language cultural compatibility**: Cross-lingual transfer usually assumes cultural compatibility across languages. Again, this is particularly critical for user-facing tasks like hate-speech detection or QA. For instance, when we train a system on hate-speech detection in English and use it for a target language like Bangla, we implicitly assume that what is considered hate speech in English would also be considered hate speech in Bangla. This is a very strong assumption that is often incorrect (Massey, 1992; Davidson et al., 2019) and if such an approach is used in practice, it could have harmful consequences. Similarly, some multilingual open-domain QA systems can now answer a question using evidence in multiple languages (Asai et al., 2021; Muller et al., 2021c). Again, this poses very concretely the question of cultural differences between the language of the question and the language from which the answer is extracted. For instance, many historical facts are perceived very differently whether you are in one country or another. By finding an answer in a language different from the user language, a multilingual QA system could therefore deliver an answer that is distant from the user's cultural context and potentially not be acceptable for the user.

Overall, with the emergence of accurate cross-lingual transfer techniques based on large-scale multilingual language models, we advocate for a move from cross-lingual transfer to cross-cultural transfer by integrating cultural dimensions into cross-lingual transfer. The first challenge for it is to define in an actionable way what we mean by culture. Hershcovich et al. (2022) recently provided a simple framework to think about culture in NLP by seeing culture along four main dimensions, namely linguistic style, *aboutness*, common ground, and values. Based on the different dimensions of cultural transfer, the second step would be to model cultural signals in NLP systems.

9.2.6 Toward Multi-view Multilingual language models

Finally, large-scale multilingual language models have reached remarkable cross-lingual transfer abilities. Those models rely on a pretraining algorithm that only requires raw textual data in multiple languages segmented at the sub-word-level (cf. §4.2.5). However, zero-shot cross-lingual transfer is still far from delivering usable models in practice (cf. chapter 7), specifically for distant languages (e.g. English to Arabic). To do better, one attempted research direction was to use parallel data to force the model's internal representations to be aligned across languages (Hu et al., 2021). However, this requires a lot of parallel data, which is not available for many language pairs. A second direction would be to use other modalities and let the pretraining process learn an internal mapping between distant languages. These modalities could be used as a *grounding* to align distant languages together. This grounding could be phonetic, speech (Bapna et al., 2022), images (Ramesh et al., 2021), video (Zellers et al., 2021), reward (Chaabouni et al., 2020). Given the modeling power of large-scale transformer models and the emerging cross-lingual abilities of those models when pretrained on large amounts of data, it is clear that feeding them richer signals could potentially lead to better multilingual models.

In this thesis, we made several contributions to answering: *How can we make language models better at handling the diversity and variability of natural languages?* . To answer it, we analyzed the behavior of transformers-based language models in a large variety of training and evaluation settings. We found concrete solutions based on cross-domain and cross-lingual transfer to build use-able models for low-resource environments. We hope our work will pave the way for further progress in building NLP systems for the most significant number of linguistic communities.

Appendix

9.3 CamemBERT complete scores

We report detailed performance of the CamemBERT model (§5).

Data	Size	GSD		Sequoia		Spoken		ParTUT		**Average**		NER	NLI
		POS	LAS	POS	LAS	POS	LAS	POS	LAS	POS	LAS	F1	Acc.
Fine-tuning													
Wiki	4GB	98.28	93.04	98.74	92.71	96.61	79.61	96.20	89.67	97.45	88.75	89.86	78.32
CCNet	4GB	98.34	93.43	98.95	93.67	96.92	**82.09**	96.50	**90.98**	97.67	**90.04**	90.46	**82.06**
OSCAR	4GB	98.35	93.55	98.97	93.70	96.94	81.97	96.58	90.28	97.71	89.87	90.65	81.88
OSCAR	138GB	**98.39**	**93.80**	**98.99**	**94.00**	**97.17**	81.18	**96.63**	90.56	**97.79**	89.88	**91.55**	81.55
Embeddings (with UDPipe Future (tagging, parsing) or LSTM+CRF (NER))													
Wiki	4GB	98.09	92.31	98.74	93.55	96.24	78.91	95.78	89.79	97.21	88.64	91.23	-
CCNet	4GB	**98.22**	**92.93**	99.12	94.65	97.17	**82.61**	**96.74**	89.95	97.81	90.04	**92.30**	-
OSCAR	4GB	98.21	92.77	99.12	**94.92**	97.20	82.47	**96.74**	**90.05**	**97.82**	**90.05**	91.90	-
OSCAR	138GB	98.18	92.77	**99.14**	94.24	**97.26**	82.44	96.52	89.89	97.77	89.84	91.83	-

Table 9.1: Results on the four tasks using language models pre-trained on data sets of varying homogeneity and size, reported on validation sets (average of 4 runs for POS tagging, parsing and NER, average of 10 runs for NLI).

9.4 Cross-Lingual Transfer Performance and Similarity

Detailed Scores and Analysis per language

We report here detailed results illustrating trends discussed in the chapter 7.

Eval	Ref	All	*Random-init of layers* 1	2	1-2	3-4	1-3	4-6	7-9	10-12	1-4	5-8	9-12
						Parsing							
English Dev	88.52	74.66	87.77	88.03	87.28	86.81	83.77	85.86	87.53	88.78	84.30	85.41	88.35
English Test	88.59	74.58	87.77	88.09	87.25	86.79	83.37	85.54	87.36	88.62	83.10	85.37	88.69
French	68.94	3.70	65.73	65.21	55.31	61.31	43.81	61.77	67.03	69.36	37.29	61.82	69.26
German	67.43	4.73	64.97	65.20	57.08	60.62	47.85	58.93	64.12	66.67	36.05	59.37	67.21
Turkish	28.40	2.76	21.65	23.77	16.78	21.21	10.69	20.23	25.39	30.43	9.70	20.94	29.33
Indonesian	45.13	4.99	43.33	43.48	39.83	39.09	33.06	40.65	44.42	46.96	30.35	40.85	47.53
Russian	59.70	2.95	57.81	57.53	54.10	53.51	47.01	52.37	56.45	61.41	38.58	52.41	60.72
Arabic	23.37	3.19	23.66	23.49	21.01	19.55	16.17	18.84	20.70	24.54	13.26	18.27	23.93
						POS							
English Dev	96.45	87.47	96.04	96.06	95.92	95.81	95.38	95.43	96.25	96.58	94.01	95.35	96.39
English Test	96.53	87.71	96.08	96.24	95.94	95.72	95.40	95.59	96.34	96.74	94.05	95.45	96.51
French	88.25	28.96	86.70	87.66	79.84	87.14	69.43	86.42	86.94	88.30	62.28	86.37	88.26
German	90.63	28.93	88.26	89.53	82.26	88.39	71.63	88.30	90.26	90.83	59.16	89.12	90.64
Turkish	72.65	32.23	62.17	66.17	54.50	63.22	47.77	66.37	70.91	72.92	44.16	69.30	73.08
Indonesian	84.06	36.98	82.15	82.89	80.13	81.40	75.94	81.99	83.78	84.42	72.42	82.59	84.09
Russian	82.97	32.63	83.14	83.63	81.95	82.26	77.93	81.69	82.98	81.76	70.33	82.56	83.19
Arabic	56.66	19.61	58.10	58.06	57.89	55.62	57.93	54.69	56.04	55.97	52.28	53.60	58.84
						NER							
English Dev	83.29	56.99	82.04	82.26	79.52	80.36	76.22	79.53	82.18	82.53	69.31	80.05	82.47
English Test	83.06	56.56	81.46	82.00	79.63	79.25	76.68	78.93	81.64	82.39	69.08	79.91	82.27
French	76.76	35.35	75.46	77.57	69.94	72.83	65.14	70.34	75.42	75.90	55.79	73.12	75.77
German	76.68	18.95	73.73	75.39	66.18	70.12	56.50	69.53	75.38	77.11	42.37	71.14	75.50
Turkish	67.64	20.76	62.54	64.84	52.20	57.11	53.03	60.59	65.66	64.87	39.38	61.43	66.62
Indonesian	53.47	21.20	49.19	49.27	46.50	46.87	43.75	47.83	54.39	48.71	36.11	46.06	48.23
Russian	58.23	7.43	55.63	58.08	50.67	52.89	42.83	46.13	53.38	58.09	34.66	52.03	59.12
Arabic	41.81	5.49	35.79	34.80	32.37	32.31	26.21	38.88	38.55	40.83	21.85	38.67	41.23

Table 9.2: Zero-shot cross-lingual performance when applying Random-init to specific set of consecutive layers compared to the Ref model. Source language is English. Baseline model ALL (for all layers randomly initialized) corresponds to a model trained from scratch on the task. For reproducibility purposes, we report performance on the Validation set English Dev. For all target languages, we report the scores on the test split of each dataset. Each score is the average of 5 runs with different random seeds. For more insights into the variability of our results, we report the min., median and max. value of the standard deviations (std) across runs with different random seeds for each task: Parsing:0.02/0.34/1.48, POS:0.01/0.5/2.38, NER:0.0/0.47/2.62 (std min/median/max).
$\geq$ Ref | $<$ Ref | $\leq$ 5 points | $\leq$ 10 points

		Random-init of layers					
Source - Target	Ref	Δ 0-1	Δ 2-3	Δ 4-5	Δ 6-7	Δ 8-9	Δ 10-11
				POS			
En - English	96.51	-0.30	-0.25	-0.40	-0.00	0.05	0.02
En - Arabic	70.20	-3.63	-1.88	-2.40	-1.26	-1.89	-2.74
En - French	89.16	-9.68	-2.09	-1.49	-1.03	0.29	0.59
En - German	89.32	-7.81	-2.12	-1.27	-0.99	-0.46	-0.68
En - Turkish	71.67	-11.62	-4.43	-1.48	-0.95	0.04	-0.95
En - Indo	71.44	-6.39	-2.80	-1.74	-0.59	-0.41	-1.10
En - Russian	86.26	-2.66	-0.94	-0.27	0.13	0.37	0.62
En - Porthughese	86.51	-10.84	-1.83	-1.44	-0.81	-0.01	-0.14
En - Spanish	87.26	-8.09	-1.30	-1.36	-1.13	0.20	0.17
En - Finish	84.85	-20.00	-8.09	-2.77	-0.97	-0.06	-0.86
En - Italian	91.35	-13.97	-3.35	-2.66	-1.34	-0.01	0.27
En - Slovenian	89.64	-16.46	-2.41	-1.09	-0.18	0.34	0.19
En - Czech	83.39	-19.62	-3.93	-0.73	-0.56	0.21	0.29
En - Polish	81.45	-13.33	-3.52	-1.19	-1.22	-0.50	-0.16
En - Hindi	65.43	-10.04	-2.70	-2.89	-3.25	3.00	0.28
En - Chinese	67.89	-3.04	-2.82	-3.59	-0.29	0.66	0.29
En - Japanese	48.86	-2.19	1.52	-1.51	-1.13	1.42	1.79
En - X (Mean)	79.37	-8.94	-2.49	-1.66	-0.88	0.20	-0.14
Ru - Russian	96.90	-0.52	-0.55	-0.40	-0.07	0.02	-0.03
Ru - English	82.55	-20.72	-7.06	-5.01	-3.93	0.74	-1.57
Ru - Arabic	79.30	-4.04	-1.48	-2.06	0.64	0.01	0.47
Ru - French	86.02	-18.66	-4.64	-4.10	-9.00	-0.13	-1.84
Ru - German	84.90	-12.50	-4.80	-2.79	-3.90	0.47	-1.82
Ru - Turkish	69.92	-15.20	-2.06	-0.55	-1.41	-0.11	0.68
Ru - Indo	71.16	-8.33	-3.44	-1.03	-0.56	-0.73	0.15
Ru - Porthughese	84.24	-19.56	-7.15	-3.00	-7.78	-0.15	-2.08
Ru - SPanish	84.84	-13.64	-4.09	-2.66	-7.67	-0.35	-2.48
Ru - Finish	81.08	-18.55	-5.42	-1.37	-1.00	-0.16	0.02
Ru - Italian	85.56	-21.04	-5.11	-3.41	-8.21	-0.20	-3.36
Ru - Slovenian	85.37	-14.65	-3.53	-1.72	-2.00	-0.15	-0.15
Ru - Czech	87.37	-8.43	-1.99	-0.71	-1.16	-0.50	-0.28
Ru - Polish	86.42	-4.41	-1.89	-0.64	-0.44	-0.21	0.09
Ru - Hindi	65.49	-1.16	0.41	-1.49	-2.17	1.13	3.20
Ru - Chinese	65.85	-5.12	-1.43	-0.32	-0.74	-0.13	-0.47
Ru - Japanese	46.91	-0.72	2.16	0.00	-1.30	1.15	1.12
Ru - X (Mean)	79.25	-10.08	-2.83	-1.65	-2.74	0.01	-0.45
Ar - Arabic	79.28	-0.35	-0.49	-0.36	-0.19	-0.05	-0.00
Ar - English	63.26	-3.32	-1.09	-1.72	-1.68	-1.03	-1.78
Ar - French	63.33	-4.41	-1.53	-1.14	-1.30	-0.44	-0.92
Ar - German	63.23	-4.95	-2.97	-1.04	-1.58	-0.53	-2.09
Ar - Turkish	60.99	-13.76	-8.74	-2.86	-4.49	-1.08	-1.88
Ar - Indo	64.24	-5.11	-3.43	-1.87	-0.58	-0.28	-0.63
Ar - Russian	74.52	-4.01	-2.37	-2.40	-1.84	-1.69	-2.03
Ar - Porthughese	67.28	-6.51	-2.84	-1.30	-1.23	0.04	-0.96
Ar - SPanish	64.84	-3.08	-0.51	-0.74	-0.48	0.02	-0.14
Ar - Finish	64.28	-19.72	-8.32	-3.72	-2.56	-1.64	-3.03
Ar - Italian	63.55	-4.25	-1.60	-0.94	-1.15	0.14	-0.64
Ar - Slovenian	68.06	-12.21	-4.31	-2.17	-1.85	0.68	-1.81
Ar - Czech	72.65	-13.57	-3.14	-1.88	-1.77	-1.35	-1.57
Ar - Polish	75.00	-8.87	-2.94	-1.46	-0.62	-1.00	-1.37
Ar - Hindi	62.29	-7.31	-6.07	-2.42	-1.26	0.19	-1.72
Ar - Chinese	56.51	-5.02	-4.94	-2.10	-1.35	-1.02	-1.77
Ar - Japanese	47.06	-3.34	-3.34	-0.65	-0.89	-1.54	-0.35
Ar - X (Mean)	64.81	-6.73	-3.50	-1.63	-1.56	-0.73	−1.29

Table 9.3: POS tagging Relative Zero shot Cross-Lingual performance of mBERT with Random-init (section 7.2.1) on pairs of consecutive layers compared to mBERT without any random-initialization (Ref). In Src - Trg, Src indicates the source language on which we fine-tune mBERT, and Trg the target language on which we evaluate it. Src-X is the average across all 17 target language with $X \neq$ Src. $\geq$ Ref $<$ Ref $\leq$ -2 points $\leq$ -5 points

		Random-init of layers					
Source - Target	Ref	Δ 0-1	Δ 2-3	Δ 4-5	Δ 6-7	Δ 8-9	Δ 10-11
		Parsing					
EN - English	88.98	-0.96	-0.66	-0.93	-0.55	0.04	-0.09
EN - Arabic	35.88	-4.05	-2.38	-3.16	-0.78	1.74	1.68
EN - French	74.04	-21.30	-6.84	-2.93	-0.69	0.03	0.76
EN - German	70.34	-15.06	-9.26	-4.75	-1.54	-0.29	1.82
EN - Turkish	34.03	-16.37	-10.10	-5.11	-3.71	0.43	1.43
EN - Indo	44.11	-10.57	-5.87	-2.66	-0.96	-0.74	0.73
EN - Russian	62.52	-7.31	-5.37	-2.84	-1.09	0.44	0.71
EN - Porthughese	68.59	-25.83	-6.22	-2.97	-0.77	0.15	0.82
EN - SPanish	69.96	-18.05	-5.74	-2.78	-0.96	0.13	0.72
EN - Finish	48.42	-24.25	-9.48	-4.39	-2.51	-0.28	0.22
EN - Italian	74.54	-30.54	-9.63	-4.18	-1.32	-0.12	0.90
EN - Slovenian	73.04	-29.89	-6.52	-3.00	-1.68	-0.05	0.18
EN - Czech	60.44	-31.84	-10.69	-4.61	-1.82	0.18	1.17
EN - Polish	55.23	-23.57	-9.11	-3.34	-1.83	0.28	0.89
EN - Hindi	28.86	-9.13	-7.58	-5.84	-2.50	1.35	1.49
EN - Chinese	27.48	-7.31	-4.47	-1.65	-0.62	0.65	1.32
EN - Japanese	11.99	-4.36	-2.76	-1.91	-1.19	0.47	1.12
EN - X (mean)	53.23	-15.77	-6.51	-3.39	-1.47	0.29	1.00
Ru - Russian	85.15	-0.82	-1.38	-1.51	-0.86	-0.29	0.18
Ru - English	61.40	-8.37	-3.55	-3.90	-0.72	1.77	1.14
Ru - Arabic	59.41	-5.65	-5.26	-5.15	-1.47	0.24	0.16
Ru - French	65.84	-8.87	-2.93	-1.81	-1.05	3.81	1.24
Ru - German	65.90	-7.02	-4.19	-1.97	-1.45	2.58	2.05
Ru - Turkish	32.20	-13.13	-7.18	-6.82	-3.77	-0.85	1.21
Ru - Indo	47.59	-4.74	-2.99	-2.30	-1.81	0.04	1.02
Ru - Porthughese	66.41	-11.17	-1.61	-1.09	-1.25	4.16	1.94
Ru - SPanish	66.74	-4.52	-1.38	-0.69	-0.97	2.95	1.37
Ru - Finish	52.92	-15.43	-6.59	-4.09	-1.35	0.12	0.77
Ru - Italian	65.28	-12.97	-3.56	-2.34	-1.46	3.16	1.55
Ru - Slovenian	62.91	-16.67	-2.71	-3.18	-1.03	0.31	1.08
Ru - Czech	72.77	-11.95	-4.17	-3.13	-1.57	-0.33	0.30
Ru - Polish	66.07	-5.70	-3.22	-2.57	-1.54	-0.12	0.54
Ru - Hindi	28.67	-6.02	-5.77	-5.27	-3.75	-0.06	0.99
Ru - Chinese	28.77	-4.66	-4.38	-3.22	-1.80	0.15	1.12
Ru - Japanese	15.10	-4.89	-3.56	-3.95	-3.11	0.68	0.73
Ru - X (Mean)	55.41	-7.69	-3.71	-3.13	-1.70	0.92	0.94
Ar - Arabic	59.54	-0.78	-2.14	-1.20	-0.67	-0.27	0.08
Ar - English	25.46	-2.09	-2.92	-0.90	-1.40	-0.97	-0.61
Ar - French	28.92	-4.85	-1.45	-0.25	-2.72	-1.60	-0.88
Ar - German	27.14	-6.38	-4.51	-0.98	-2.24	0.13	0.09
Ar - Turkish	9.58	-3.90	-3.14	-2.76	-2.33	0.31	0.15
Ar - Indo	36.16	-5.85	-4.86	-1.71	-0.68	-0.17	0.58
Ar - Russian	42.25	-3.52	-5.28	-2.46	-1.66	-0.67	-0.27
Ar - Porthughese	34.71	-4.80	-1.22	0.10	-2.98	-0.33	-0.24
Ar - SPanish	31.95	-4.02	-0.15	-0.44	-1.46	-0.77	0.38
Ar - Finish	28.18	-9.89	-7.03	-3.17	-1.81	-0.58	-0.42
Ar - Italian	28.85	-3.01	0.60	1.45	-2.26	-1.47	-0.70
Ar - Slovenian	35.78	-9.73	-4.97	-2.21	-1.43	-0.41	-0.56
Ar - Czech	40.04	-13.61	-6.82	-3.20	-2.38	-1.12	-0.21
Ar - Polish	41.16	-8.46	-5.52	-2.48	-1.48	-0.47	-0.55
Ar - Hindi	10.24	-2.46	-2.86	-2.57	-1.55	1.00	0.14
Ar - Chinese	11.46	-2.42	-2.43	-1.26	-0.82	0.23	-0.05
Ar - Japanese	6.66	-1.28	-0.79	-1.20	-1.04	0.74	0.30
Ar - X (Mean)	27.97	-4.91	-3.17	-1.48	-1.68	-0.36	-0.14

Table 9.4: Parsing (LAS score) Relative Zero shot Cross-Lingual performance of mBERT with Random-init (section 7.2.1) on pairs of consecutive layers compared to mBERT without any random-initialization (Ref). In Src - Trg, Src indicates the source language on which we fine-tune mBERT, and Trg the target language on which we evaluate it. Src-X is the average across all 17 target language with X $\neq$ Src $\geq$ Ref $<$ Ref $\leq$ -2 points $\leq$ -5 points

		Random-init of layers					
Source - Target	Ref	Δ 0-1	Δ 2-3	Δ 4-5	Δ 6-7	Δ 8-9	Δ 10-11
				NER			
EN - English	83.27	-2.64	-2.12	-1.41	-0.61	-0.21	-0.14
EN - French	76.20	-4.41	-2.72	-2.09	-0.30	0.51	0.08
EN - German	75.58	-8.25	-4.65	-2.50	-0.40	0.06	0.26
EN - Turkish	66.23	-8.71	-6.57	-2.16	-1.01	0.51	0.51
EN - Indo	50.24	-2.94	-1.43	-2.54	2.49	-0.70	0.82
EN - Porthughese	76.09	-4.66	-0.88	-1.16	-0.57	0.62	-0.70
EN - SPanish	67.00	-0.99	4.37	2.03	-1.69	1.57	-1.38
EN - Finish	75.61	-11.89	-4.47	-2.29	0.63	0.54	-0.37
EN - Italian	78.48	-6.65	-3.64	-3.08	-1.32	-0.30	-0.28
EN - Slovenian	72.80	-10.37	-2.96	-3.11	-0.36	0.10	-0.72
EN - Czech	76.90	-8.02	-6.81	-3.17	0.09	1.00	0.39
EN - Russian	60.20	-5.87	-6.65	-5.71	-2.82	-0.82	-0.37
EN - Arabic	39.15	-8.98	-5.31	-1.97	1.56	0.31	-0.98
EN - Polish	77.20	-8.32	-5.53	-3.05	-0.06	0.67	0.09
EN - Hindi	60.61	-12.08	-13.88	-9.23	-0.91	-1.25	2.08
EN - Chinese	37.74	-13.68	-6.49	-4.59	-2.41	-5.23	-1.00
EN - Japanese	25.19	-11.40	-7.54	-4.67	-2.53	-3.45	-0.23
EN - X (mean)	64.17	-8.28	-5.09	-3.07	-0.79	-0.47	-0.13
Ru - Russian	88.20	-2.08	-2.13	-1.52	-0.64	-0.33	-0.13
Ru - English	56.62	-13.83	-8.52	-4.70	-1.50	-0.76	1.38
Ru - French	67.35	-18.45	-9.70	-4.32	-1.76	-1.77	2.29
Ru - German	69.23	-13.94	-9.01	-5.80	-2.98	-1.65	0.40
Ru - Turkish	63.64	-18.52	-10.06	-6.01	-4.16	-0.67	-0.27
Ru - Indo	41.92	-10.29	-7.20	-5.19	-1.20	-1.91	0.50
Ru - Porthughese	67.33	-21.23	-8.27	-8.84	-2.83	-1.83	1.51
Ru - SPanish	69.15	-16.74	-10.00	-8.16	-5.80	-1.66	0.26
Ru - Finish	73.03	-17.17	-8.70	-5.88	-2.12	0.86	1.48
Ru - Italian	70.05	-19.47	-9.54	-6.90	-3.06	0.73	1.04
Ru - Slovenian	71.18	-12.02	-9.48	-3.61	-0.70	1.16	2.14
Ru - Czech	74.87	-17.93	-10.59	-6.34	-4.02	0.17	-0.23
Ru - Arabic	38.63	-8.67	-6.81	-0.13	-0.65	-1.34	-0.29
Ru - Polish	75.16	-15.38	-7.97	-6.33	-3.07	-0.63	1.34
Ru - Hindi	58.01	-19.60	-12.36	-6.18	0.93	-1.64	1.17
Ru - Chinese	43.86	-23.73	-11.68	-6.80	-4.27	-4.13	-6.01
Ru - Japanese	30.79	-16.80	-11.29	-5.26	-2.77	-3.99	-6.91
Ru - X (Mean)	62.13	-15.85	-9.36	-5.50	-2.44	-1.16	-0.06
Ar - Arabic	87.97	-2.37	-2.11	-0.96	-0.39	-0.15	0.21
Ar - French	75.21	-18.71	-8.31	-3.76	-0.19	0.82	1.07
Ar - German	74.24	-15.25	-7.19	-3.72	-1.38	-0.04	0.27
Ar - Turkish	68.45	-14.89	-8.65	-2.78	-0.30	0.98	1.90
Ar - Indo	54.65	-13.86	-10.95	-8.53	-4.66	-2.82	0.09
Ar - Porthughese	74.67	-20.42	-10.54	-3.17	-1.59	0.10	1.28
Ar - SPanish	74.88	-18.16	-12.18	-3.06	-1.95	0.52	0.63
Ar - Finish	78.01	-18.79	-8.84	-4.30	-2.03	-0.30	0.19
Ar - Italian	75.76	-16.37	-7.73	-3.98	-1.49	-0.06	0.74
Ar - Slovenian	63.08	-11.13	-5.49	4.79	0.88	2.17	0.79
Ar - Czech	74.70	-21.93	-10.95	-5.84	-2.42	-1.36	0.09
Ar - Russian	45.51	-7.59	-5.81	-2.63	0.15	-0.22	0.47
Ar - English	57.94	-12.79	-6.03	-4.57	-0.32	0.29	1.65
Ar - Polish	77.29	-20.61	-9.47	-5.93	-2.64	-1.09	-0.19
Ar - Hindi	65.31	-14.95	-9.12	-3.84	-1.48	0.72	0.98
Ar - Chinese	45.88	-25.72	-10.67	-3.99	-1.41	-2.72	0.57
Ar - Japanese	24.75	-14.66	-5.19	-3.82	-0.99	-1.17	1.50
Ar - X (Mean)	65.59	-16.10	-8.42	-3.73	-1.40	-0.25	0.67

Table 9.5: NER (F1 score) Relative Zero shot Cross-Lingual performance of mBERT with Random-init (section 7.2.1) on pairs of consecutive layers compared to mBERT without any random-initialization (Ref). In Src - Trg, Src indicates the source language on which we fine-tune mBERT, and Trg the target language on which we evaluate it. Src-X is the average across all 17 target language with X ≠ Src ≥ Ref < Ref ≤ -2 points ≤ -5 points

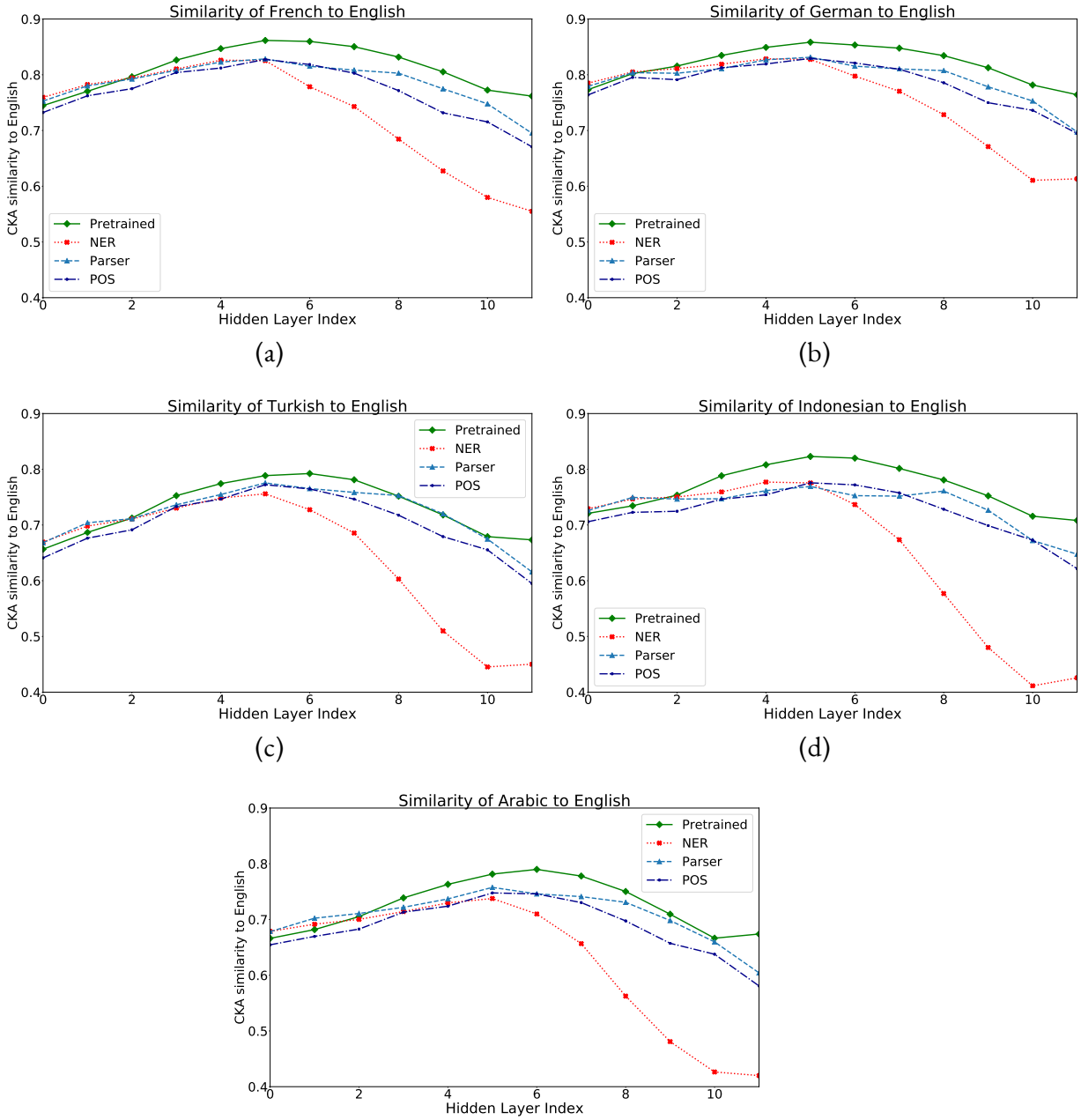


Figure 9.1: Cross-Lingual similarity (CKA) similarity (§7.5) of hidden representations of a source language (English) sentences with a target language sentences on fine-tuned and pretrained mBERT. The higher the CKA value the greater the similarity.

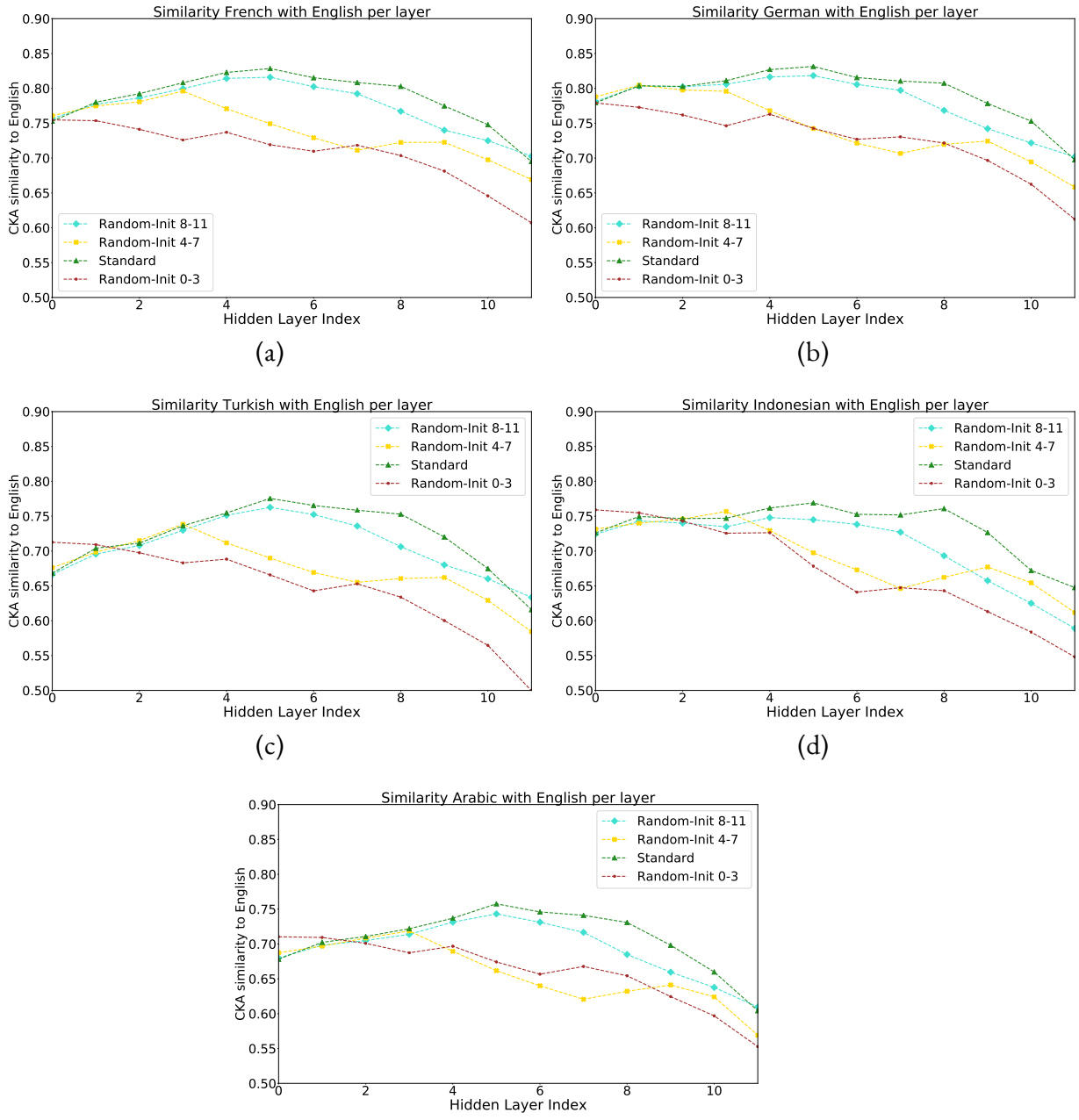


Figure 9.2: Cross-Lingual similarity (CKA) (§7.5) of hidden representations of a source language (English) sentences with target languages sentences on fine-tuned **Parsing** models with and without RANDOM-INIT. The higher the CKA value the greater the similarity.

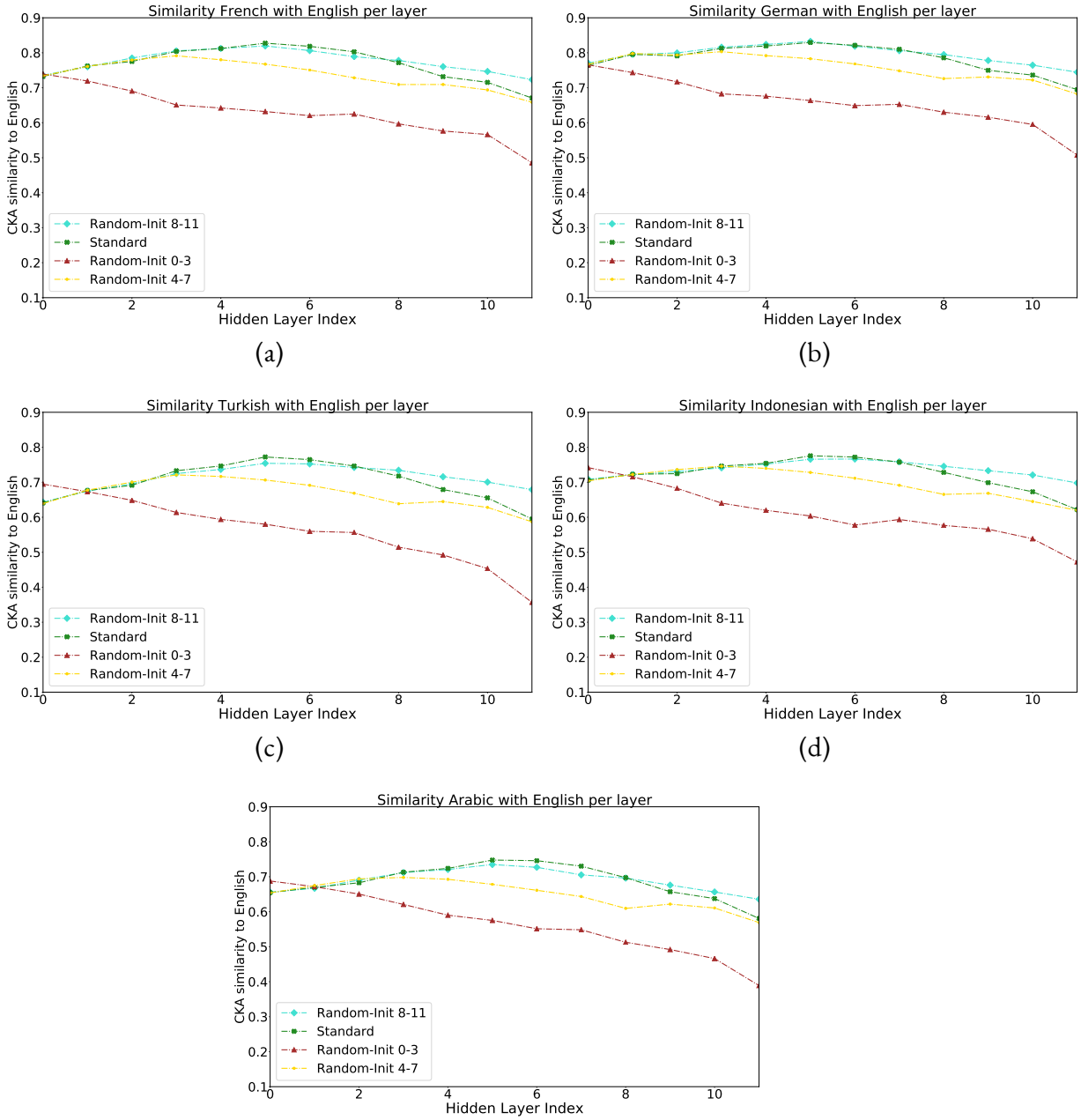


Figure 9.3: Cross-Lingual similarity (CKA) (§7.5) of hidden representations of a source language (English) sentences with target languages sentences on fine-tuned **POS** models with and w/o RANDOM-INIT. The higher the CKA value the greater the similarity.

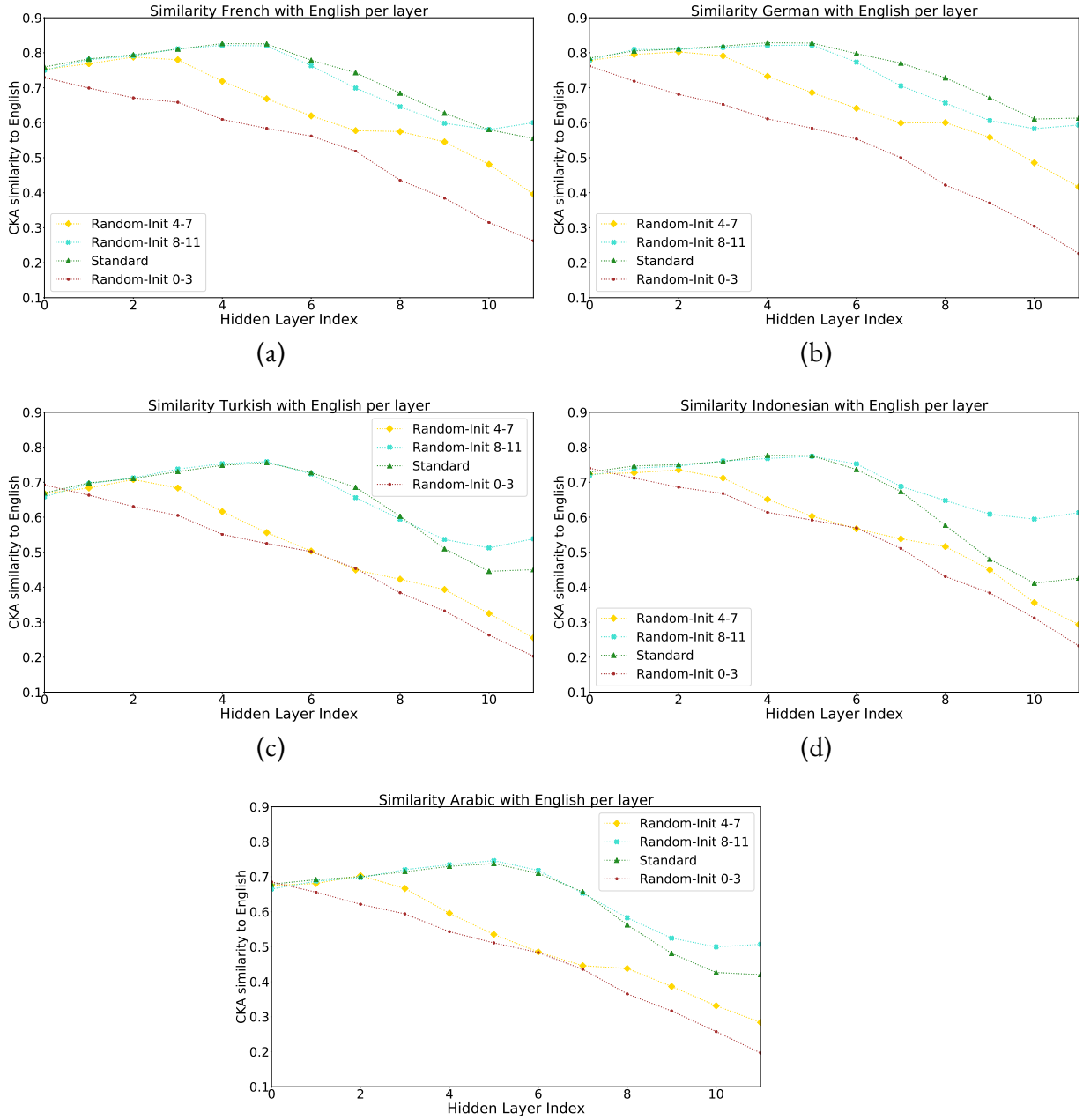


Figure 9.4: Cross-Lingual similarity (CKA) (§7.5) of hidden representations of a source language (English) sentences with target languages sentences on fine-tuned **NER** models with and w/o RANDOM-INIT. The higher the CKA value the greater the similarity.

Bibliography

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, page 265–283, USA. USENIX Association.

A. Abeillé, Lionel Clément, and Alexandra Kinyon. 2000. Building a treebank for french. In *LREC*.

Steven Abney. 1997. The scol manual, version 0.1 b.

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Zeljko Agic, Anders Johannsen, Barbara Plank, Héctor Martínez Alonso, Natalie Schluter, and Anders Søgaard. 2016. Multilingual projection for parsing truly low-resource languages. *Transactions of the Association for Computational Linguistics*, 4:301–312.

Roee Aharoni and Yoav Goldberg. 2020. Unsupervised domain clusters in pretrained language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7747–7763, Online. Association for Computational Linguistics.

Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. 1986. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley series in computer science / World student series edition. Addison-Wesley.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 1638–1649. Association for Computational Linguistics.

Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. 2019. Character-level language modeling with deeper self-attention. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3159–3166.

Rami Al-Rfou', Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria. Association for Computational Linguistics.

Guillaume Alain and Yoshua Bengio. 2017. Understanding intermediate layers using linear classifier probes. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net.

Chantal Amrhein and Rico Sennrich. 2020. On Romanization for model transfer between scripts in neural machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2461–2469, Online. Association for Computational Linguistics.

Alex Andonian, Quentin Anthony, Stella Biderman, Sid Black, Preetham Gali, Leo Gao, Eric Hallahan, Josh Levy-Kramer, Connor Leahy, Lucas Nestler, Kip Parker, Michael Pieler, Shivanshu Purohit, Tri Songz, Phil Wang, and Samuel Weinbach. 2021. GPT-NeoX: Large scale autoregressive language modeling in pytorch.

Wissam Antoun, Fady Baly, and Hazem M. Hajj. 2020. Arabert: Transformer-based model for arabic language understanding. *ArXiv*, abs/2003.00104.

Douglas E. Appelt, Jerry R. Hobbs, John Bear, David J. Israel, and Mabry Tyson. 1993. Fastus: A finite-state processor for information extraction from real-world text. In *IJCAI*.

Bolanle Arokoyo. 2006. A comparative study of reduplication in hausa and standard yoruba. 16.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations. In *AAAI*.

Akari Asai, Xinyan Yu, Jungo Kasai, and Hanna Hajishirzi. 2021. One question answering model for many languages with cross-lingual dense passage retrieval. In *Advances in Neural Information Processing Systems*, volume 34, pages 7547–7560. Curran Associates, Inc.

Lucas F.E. Ashby, Travis M. Bartley, Simon Clematide, Luca Del Signore, Cameron Gibson, Kyle Gorman, Yeonju Lee-Sikka, Peter Makarov, Aidan Malanoski, Sean Miller, Omar Ortiz, Reuben Raff, Arundhati Sengupta, Bora Seo, Yulia Spektor, and Winnie Yan. 2021. Results of the second SIGMORPHON shared task on multilingual grapheme-to-phoneme conversion. In *Proceedings of the 18th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 115–125, Online. Association for Computational Linguistics.

Fatemeh Torabi Asr and Maite Taboada. 2019. Big data and quality data for fake news and misinformation detection. *Big Data & Society*, 6.

Giuseppe Attardi. 2015. Wikiextractor. `https://github.com/attardi/wikiextractor`.

Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *ArXiv*, abs/1607.06450.

Stephen H. Bach, Victor Sanh, Zheng Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M. Saiful Bari, Thibault Févry, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-David, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Alan Fries, Maged Saeed AlShaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru Tang, Mike Tian-Jian Jiang, and Alexander M. Rush. 2022. Promptsource: An integrated development environment and repository for natural language prompts. *CoRR*, abs/2202.01279.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

L. R. Bahl and R. L. Mercer. 1976. Part of speech assignment by a statistical decision algorithm. In *Proceedings IEEE International Symposium on Information Theory*, pages 88–89.

Timothy Baldwin, Paul Cook, Marco Lui, Andrew MacKinlay, and Li Wang. 2013. How noisy social media text, how diffrnt social media sources? In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 356–364.

Timothy Baldwin, Marie-Catherine de Marneffe, Bo Han, Young-Bum Kim, Alan Ritter, and Wei Xu. 2015. Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 126–135.

Ankur Bapna, Colin Cherry, Yu Zhang, Ye Jia, Melvin Johnson, Yong Cheng, Simran Khanuja, Jason Riesa, and Alexis Conneau. 2022. mslam: Massively multilingual joint pre-training for speech and text. *ArXiv*, abs/2202.01374.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247, Baltimore, Maryland. Association for Computational Linguistics.

Leonard E. Baum and Ted Petrie. 1966. Statistical inference for probabilistic functions of finite state markov chains. *Annals of Mathematical Statistics*, 37:1554–1563.

T. Bayes. 1763. An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions*, 53:370–418.

Russell Beckley. 2015. Bekli: A simple approach to twitter text normalization. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 82–86.

Yonatan Belinkov, Sebastian Gehrmann, and Ellie Pavlick. 2020. Interpretability and analysis in neural nlp. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 1–5.

Emily M. Bender. 2013. Linguistic fundamentals for natural language processing.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? . In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA. Association for Computing Machinery.

Oliver Bender, Franz Josef Och, and Hermann Ney. 2003. Maximum entropy models for named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, page 148–151, USA. Association for Computational Linguistics.

Julian Benello, Andrew Mackie, and J.A. Anderson. 1989. Syntactic category disambiguation with neural networks. *Computer Speech & Language*, 3:203–217.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2001. A neural probabilistic language model. In *Advances in Neural Information Processing Systems*, volume 13. MIT Press.

Gábor Berend and Ervin Tasnádi. 2015. Uszeged: correction type-sensitive normalization of english tweets using efficiently indexed n-gram statistics. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 120–125.

Adam L. Berger, Stephen Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Comput. Linguistics*, 22:39–71.

Douglas Biber, Susan Conrad, and Randi Reppen. 1998. *Corpus Linguistics: Investigating Language Structure and Use*. Cambridge Approaches to Linguistics. Cambridge University Press.

Chris Biemann, Gerhard Heyer, Uwe Quasthoff, and Matthias Richter. 2007. The Leipzig Corpora collection-monolingual corpora of standard size. *Proceedings of Corpus Linguistic*, 2007.

Alexandra Birch, Nadir Durrani, and Philipp Koehn. 2013. English SLT and MT system description for the IWSLT 2013 evaluation. In *Proceedings of the 10th International Workshop on Spoken Language Translation: Evaluation Campaign*, Heidelberg, Germany.

Alan W. Black, Kevin A. Lenzo, and Vincent Pagel. 1998. Issues in building general letter to sound rules. In *SSW*.

Damian Blasi, Antonios Anastasopoulos, and Graham Neubig. 2022. Systematic inequalities in language technology performance across the world's languages. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5486–5505, Dublin, Ireland. Association for Computational Linguistics.

Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. 2020. Language (technology) is power: A critical survey of "bias" in NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5454–5476, Online. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*, 5:135–146.

Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. 1992. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, page 144–152, New York, NY, USA. Association for Computing Machinery.

Alexandre Bouchard, Percy Liang, Thomas Griffiths, and Dan Klein. 2007. A probabilistic approach to diachronic phonology. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 887–896, Prague, Czech Republic. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015a. A large annotated corpus for learning natural language inference. In *Proceed-*

ings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015b. A large annotated corpus for learning natural language inference. In *EMNLP*.

Jonathan Bragg, Arman Cohan, Kyle Lo, and Iz Beltagy. 2021. Flex: Unifying evaluation for few-shot nlp. In *NeurIPS*.

L. Breiman, Jerome H. Friedman, Richard A. Olshen, and C. J. Stone. 1983. Classification and regression trees.

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

Eric Brill. 1992. A simple rule-based part of speech tagger. In *HLT*.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.

Eric Brill and Robert C. Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 286–293, Hong Kong. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Marc Brysbaert, Michaël Stevens, Paweł Mandera, , and Emmanuel Keuleers. 2016. How many words do we know? practical estimates of vocabulary size dependent on word definition, the degree of language input and the participant's age. *Frontiers in Psychology*.

Mary Bucholtz and Kira Hall. 2004. *Language and Identity*, pages 369 – 394.

Cristian Buciluundefined, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, page 535–541, New York, NY, USA. Association for Computing Machinery.

A.W. Burks. 1947. Electronic computing circuits of the eniac. *Proceedings of the IRE*, 35(8):756–767.

Lyle Campbell. 2013. *Historical linguistics*. Edinburgh University Press.

Bernard Caron, Marine Courtin, Kim Gerdes, and Sylvain Kahane. 2019. A surface-syntactic UD treebank for Naija. In *Proceedings of the 18th International Workshop on Treebanks and Linguistic Theories (TLT, SyntaxFest 2019)*, pages 13–24.

Augustin Cauchy et al. 1847. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538.

José Cañete, Gabriel Chaperon, Rodrigo Fuentes, Jou-Hui Ho, Hojin Kang, and Jorge Pérez. 2020. Spanish pre-trained bert model and evaluation data. In *PML4DC at ICLR 2020*.

Slavomír Čéplö, Ján Bátora, Adam Benkato, Jiří Milička, Christophe Pereira, and Petr Zemánek. 2016. Mutual intelligibility of spoken maltese, libyan arabic, and tunisian arabic functionally tested: A pilot study. *Folia Linguistica*, 50(2):583–628.

Rahma Chaabouni, Eugene Kharitonov, Diane Bouchacourt, Emmanuel Dupoux, and Marco Baroni. 2020. Compositionality and generalization in emergent languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4427–4442, Online. Association for Computational Linguistics.

Wallace Chafe and Deborah Tannen. 1987. The relation between written and spoken language. *Annual Review of Anthropology*, 16:383–407.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *AAAI/IAAI*.

Eugene Charniak, Curtis Hendrickson, Neil Jacobson, and Mike Perkowitz. 1993. Equations for part-of-speech tagging. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, AAAI'93, page 784–789. AAAI Press.

Ethan C. Chau, Lucy H. Lin, and Noah A. Smith. 2020. Parsing with multilingual BERT, a small corpus, and a small treebank. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1324–1334, Online. Association for Computational Linguistics.

Yves Chauvin and David E. Rumelhart. 1995. Backpropagation: theory, architectures, and applications.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. *ArXiv*, abs/1606.02858.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *34th Annual Meeting of the Association for Computational Linguistics*, pages 310–318, Santa Cruz, California, USA. Association for Computational Linguistics.

Ethan A. Chi, John Hewitt, and Christopher D. Manning. 2020. Finding universal grammatical relations in multilingual BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5564–5577, Online. Association for Computational Linguistics.

Gennaro Chierchia and Sally Mcconnell-Ginet. 1990. Meaning and grammar: An introduction to semantics.

N. Chinchor and P. Robinson. 1998. Appendix E: MUC-7 named entity task definition (version 3.5). In *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29 - May 1, 1998*.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Franccois Chollet. 2019. On the measure of intelligence. *ArXiv*, abs/1911.01547.

Francois Chollet. 2021. *Deep learning with Python*. Simon and Schuster.

Monojit Choudhury, Rahul Saraf, Vijit Jain, Animesh Mukherjee, Sudeshna Sarkar, and Anupam Basu. 2007. Investigation and modeling of the structure of texting language. *International Journal of Document Analysis and Recognition (IJDAR)*, 10:157–174.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta,

Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways.

Christos Christodoulopoulos and Mark Steedman. 2015. A massively parallel corpus: the bible in 100 languages. *Language Resources and Evaluation*, 49:375 – 395.

Yoeng-Jin Chu and T-H Liu. 1965. On the shortest arborescence of a directed graph. *Scientia Sinica*, 14:1396–1400.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Comput. Linguist.*, 16(1):22–29.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 845–855, Melbourne, Australia. Association for Computational Linguistics.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020a. TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470.

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2021. Canine: Pre-training an efficient tokenization-free encoder for language representation.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020b. Pre-training transformers as energy-based cloze models. In *EMNLP*.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, ACL '98/EACL '98, page 16–23, USA. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2007. Fast semantic extraction using a novel neural network architecture. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 560–567, Prague, Czech Republic. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 160–167, New York, NY, USA. Association for Computing Machinery.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12(null):2493–2537.

Bernard Comrie. 2013. Writing systems. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020a. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020b. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018a. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, pages 7057–7067.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018b. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium. Association for Computational Linguistics.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1107–1116, Valencia, Spain. Association for Computational Linguistics.

Alexis Conneau, Shijie Wu, Haoran Li, Luke Zettlemoyer, and Veselin Stoyanov. 2020c. Emerging cross-lingual structure in pretrained language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6022–6034, Online. Association for Computational Linguistics.

Kate Crawford. 2017. The trouble with bias. URL: https://nips.cc/Conferences/2017/Schedule?showEvent=8742.

D Alan Cruse, David Alan Cruse, D A Cruse, and D A Cruse. 1986. *Lexical semantics*. Cambridge university press.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2020. Revisiting pre-trained models for Chinese natural language processing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 657–668, Online. Association for Computational Linguistics.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. 2021. Pre-training with whole word masking for chinese bert. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3504–3514.

Walter Daelemans and Miles Osborne, editors. 2003. *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*. ACL.

Walter Daelemans and Antal van den Bosch. 1996. Language-independent data-oriented grapheme-to-phoneme conversion.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005a. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005b. The pascal recognising textual entailment challenge. In *MLCW*.

Ido Dagan, Shaul Marcus, and Shaul Markovitch. 1993. Contextual word similarity and estimation from sparse data. In *31st Annual Meeting of the Association for Computational Linguistics*, pages 164–171, Columbus, Ohio, USA. Association for Computational Linguistics.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

P. Daniels and W. Bright. 2010. *The World's Writing Systems*. Oxford University Press, Incorporated.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. *ArXiv*, abs/1912.02164.

Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z. Wang. 2008. Image retrieval: Ideas, influences, and trends of the new age. *ACM Comput. Surv.*, 40(2).

Yann N. Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. 2014. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2933–2941, Cambridge, MA, USA. MIT Press.

Thomas Davidson, Debasmita Bhattacharya, and Ingmar Weber. 2019. Racial bias in hate speech and abusive language detection datasets. In *Proceedings of the Third Workshop on Abusive Language Online*, pages 25–35, Florence, Italy. Association for Computational Linguistics.

Éric de La Clergerie, Benoît Sagot, and Djamé Seddah. 2017. The ParisNLP entry at the ConLL UD shared task 2017: A tale of a #ParsingTragedy. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 243–252, Vancouver, Canada. Association for Computational Linguistics.

Marie-Catherine de Marneffe, Christopher D. Manning, Joakim Nivre, and Daniel Zeman. 2021. Universal Dependencies. *Computational Linguistics*, 47(2):255–308.

Wietse de Vries, Andreas van Cranenburgh, Arianna Bisazza, Tommaso Caselli, Gertjan van Noord, and Malvina Nissim. 2019. Bertje: A dutch bert model. *ArXiv*, abs/1912.09582.

Pieter Delobelle, Thomas Winters, and Bettina Berendt. 2020. RobBERT: a Dutch RoBERTa-based Language Model. ArXiv preprint 2001.06286.

Pascal Denis and Benoît Sagot. 2009. Coupling an annotated corpus and a morphosyntactic lexicon for state-of-the-art POS tagging with less human effort. In *Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation, Volume 1*, pages 110–119, Hong Kong. City University of Hong Kong.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018a. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018b. Multilingual bert. https://github.com/google-research/bert/blob/master/multilingual.md.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019a. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019b. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the CoNLL 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Vancouver, Canada, August 3-4, 2017*, pages 20–30. Association for Computational Linguistics.

Matthew S. Dryer. 2013a. Order of subject, object and verb. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Matthew S. Dryer. 2013b. Prefixing vs. suffixing in inflectional morphology. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Yoann Dupont. 2017. Exploration de traits pour la reconnaissance d'entit'es nomm'ees du français par apprentissage automatique. In *24e Conf'erence sur le Traitement Automatique des Langues Naturelles (TALN)*, page 42.

Nadir Durrani, Hassan Sajjad, Hieu Hoang, and Philipp Koehn. 2014a. Integrating an unsupervised transliteration model into statistical machine translation. In *EACL*.

Nadir Durrani, Hassan Sajjad, Hieu Hoang, and Philipp Koehn. 2014b. Integrating an unsupervised transliteration model into statistical machine translation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 148–153, Gothenburg, Sweden. Association for Computational Linguistics.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics.

Gary F. Simons Eberhard, David M. and Charles D. Fennig. 2021. *Ethnologue: Languages of the World*, 24th edition. SIL International, Dallas, TX, USA.

Abdessamad Echihabi and Daniel Marcu. 2003. A noisy-channel approach to question answering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 16–23, Sapporo, Japan. Association for Computational Linguistics.

Penelope Eckert. 2017. Age as a sociolinguistic variable. *The handbook of sociolinguistics*, pages 151–167.

Jack Edmonds et al. 1967. Optimum branchings. *Journal of Research of the national Bureau of Standards B*, 71(4):233–240.

Jacob Eisenstein. 2013a. What to do about bad language on the internet. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 359–369, Atlanta, Georgia. Association for Computational Linguistics.

Jacob Eisenstein. 2013b. What to do about bad language on the internet. In *HLT-NAACL*, Atlanta, USA.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Y. Goldberg. 2020. Amnesic probing: Behavioral explanation with amnesic counterfactuals. *arXiv: Computation and Language*.

Joseph Eska and Oswald Szemerenyi. 2000. Introduction to indo-european linguistics. *Language*, 76:456.

Fahim Faisal, Yinkai Wang, and Antonios Anastasopoulos. 2022. Dataset geography: Mapping language data to language users. In *ACL*.

Heidi M. Feldman, Christine A. Dollaghan, Thomas F. Campbell, Marcia Kurs-Lasky, Janine E. Janosky, and Jack L. Paradise. 2000. Measurement properties of the macarthur communicative development inventories at ages one and two years. *Child development*, 71 2:310–22.

Larry Fenson, Philip S. Dale, Jeffrey S. Reznick, Elizabeth A. Bates, Donna J. Thal, and Stephen Pethick. 1994. Variability in early communicative development. *Monographs of the Society for Research in Child Development*, 59 5:1–173; discussion 174–85.

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *ACL*.

J. R. Firth. 1935. The technique of semantics. *Transactions of the Philological Society*, 34:36–73.

J. R. Firth. 1957. A synopsis of linguistic theory, 1930-1955.

Matthew Fisher, Mariel K Goddu, and Frank C Keil. 2015. Searching for explanations: How the internet inflates estimates of internal knowledge. *Journal of experimental psychology: General*, 144(3):674.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 168–171.

Jennifer Foster. 2010a. "cba to check the spelling": Investigating parser performance on discussion forum posts. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 381–384, Los Angeles, California. Association for Computational Linguistics.

Jennifer Foster. 2010b. "cba to check the spelling": Investigating parser performance on discussion forum posts. In *NAACL*, Los Angeles, California.

Carmen Fought. 2011. *Language and ethnicity*, Cambridge Handbooks in Language and Linguistics, page 238–258. Cambridge University Press.

W Nelson Francis and Henry Kucera. 1979. Brown corpus manual. *Letters to the Editor*, 5(2):7.

Roy Frostig, Matthew Johnson, and Chris Leary. 2018. Compiling machine learning programs via high-level tracing.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal archive*, 12:23–38.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3356–3369, Online. Association for Computational Linguistics.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Comput. Linguist.*, 28(3):245–288.

Howard Giles and Nikolas Coupland. 1991. *Language: Contexts and consequences.* Thomson Brooks/Cole Publishing Co.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual language processing from bytes. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1296–1306, San Diego, California. Association for Computational Linguistics.

Lila R. Gleitman and Anna Papafragou. 2005. Language and thought.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.

Hila Gonen, Shauli Ravfogel, Yanai Elazar, and Yoav Goldberg. 2020. It's not greek to mbert: Inducing word-level translations from multilingual bert. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 45–56.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Charlotte Gooskens, Sebastian Kürschner, and Renée van Bezooijen. 2011. Intelligibility of standard german and low german to speakers of dutch. *Dialectologia*, pages 35–63.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*. European Language Resources Association (ELRA).

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2017a. Improving neural language models with a continuous cache. *ArXiv*, abs/1612.04426.

Edouard Grave, Tomas Mikolov, Armand Joulin, and Piotr Bojanowski. 2017b. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pages 427–431. Association for Computational Linguistics.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks : the official journal of the International Neural Network Society*, 18 5-6:602–10.

Bert F. Green, Alice K. Wolf, Carol L. Chomsky, and Kenneth Laughery. 1961. Baseball: an automatic question-answerer. In *IRE-AIEE-ACM '61 (Western)*.

Roman Grundkiewicz and Kenneth Heafield. 2018. Neural machine translation techniques for named entity transliteration. In *Proceedings of the Seventh Named Entities Workshop*, pages 89–94, Melbourne, Australia. Association for Computational Linguistics.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1234–1244, Beijing, China. Association for Computational Linguistics.

Gregory R. Guy. 2011. *Language, social class, and status*, Cambridge Handbooks in Language and Linguistics, page 159–185. Cambridge University Press.

Nizar Habash. 2010. *Introduction to Arabic Natural Language Processing*. Morgan and Claypool.

Nizar Habash and Owen Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 573–580, Ann Arbor, Michigan. Association for Computational Linguistics.

Mariam Haghegh. 2021. Arabizi across three different generations of arab users living abroad: A case study. *Arab World English Journal*, 5:156–173.

Jan Hajic, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez i Villodre, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Stepánek, Pavel Stranák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *CoNLL Shared Task*.

Jan Hajič, Otakar Smrž, Petr Zemánek, Petr Pajas, Jan Šnaidauf, Emanuel Beška, Jakub Kracmar, and Kamila Hassanová. 2009. Prague arabic dependency treebank 1.0.

Michael Alexander Kirkwood Halliday, Christian MIM Matthiessen, Michael Halliday, and Christian Matthiessen. 2014. *An introduction to functional grammar*. Routledge.

Kristy A Hamilton and Mike Z Yao. 2018. Blurring boundaries: Effects of device features on metacognitive evaluations. *Computers in Human Behavior*, 89:213–220.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a# twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 368–378. Association for Computational Linguistics.

Lifeng Han, Derek F. Wong, Lidia S. Chao, Liangye He, Ling Zhu, and S. Li. 2013. A study of chinese word segmentation based on the characteristics of chinese. In *GSCL*.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Trevor J. Hastie, Robert Tibshirani, and Jerome H. Friedman. 2001. The elements of statistical learning.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. {DEBERTA}: {DECODING}-{enhanced} {bert} {with} {disentangled} {attention}. In *International Conference on Learning Representations*.

David Heckerman, David Maxwell Chickering, Christopher Meek, Robert Rounthwaite, and Carl Kadie. 2001. Dependency networks for inference, collaborative filtering, and data visualization. *J. Mach. Learn. Res.*, 1:49–75.

Monica Heller, Lindsay Bell, Michelle Daveluy, Mireille Mclaughlin, and Hubert Noël. 2016. *Sustaining the Nation: The Making and Moving of Language and Nation*.

John L Hennessy and David A Patterson. 2019. A new golden age for computer architecture. *Communications of the ACM*, 62(2):48–60.

Henry and Pramoolsook. 2014. Arabizi: An analysis of the romanization of the arabic script from a sociolinguistic perspective.

Kucera Henry. 1986. Automated word substitution using numerical rankings of structural disparity between misspelled words candidate substitution words. US Patent US4783758.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 1693–1701, Cambridge, MA, USA. MIT Press.

Ulf Hermjakob, Jonathan May, and Kevin Knight. 2018. Out-of-the-box universal Romanization tool uroman. In *Proceedings of ACL 2018, System Demonstrations*, pages 13–18, Melbourne, Australia. Association for Computational Linguistics.

Daniel Hershcovich, Stella Frank, Heather Lent, Miryam de Lhoneux, Mostafa Abdou, Stephanie Brandl, Emanuele Bugliarello, Laura Cabello Piqueras, Ilias Chalkidis, Ruixiang Cui, Constanza Fierro, Katerina Margatina, Phillip Rust, and Anders Søgaard. 2022. Challenges and strategies in cross-cultural NLP. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6997–7013, Dublin, Ireland. Association for Computational Linguistics.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743.

Francis Heylighen, Jean-Marc Dewaele, and Léo Apostel. 1999. Formality of language: definition, measurement and behavioral determinants.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1367–1377, San Diego, California. Association for Computational Linguistics.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531.

Lynette Hirschman, Marc Light, Eric Breck, and John D. Burger. 1999. Deep read: A reading comprehension system. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 325–332, College Park, Maryland, USA. Association for Computational Linguistics.

Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 6(2):107–116.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and L. Sifre. 2022. Training compute-optimal large language models. *ArXiv*, abs/2203.15556.

John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. 2006. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., USA.

Harold Hotelling. 1992. Relations between two sets of variates. In *Breakthroughs in statistics*, pages 162–190. Springer.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. *arXiv preprint arXiv:1902.00751*.

Dirk Hovy and Anders Søgaard. 2015. Tagging performance correlates with author age. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 483–488, Beijing, China. Association for Computational Linguistics.

Dirk Hovy and Diyi Yang. 2021. The importance of modeling social factors of language: Theory and practice. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 588–602, Online. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*.

Annie Hu, Cindy Wang, and Brandon Yang. 2017. Question answering using match-lstm and answer pointer.

Junjie Hu, Melvin Johnson, Orhan Firat, Aditya Siddhant, and Graham Neubig. 2021. Explicit alignment objectives for multilingual bidirectional encoders. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3633–3643, Online. Association for Computational Linguistics.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4411–4421. PMLR.

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam M. Shazeer, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. 2019. Music transformer: Generating music with long-term structure. In *ICLR*.

Matthew Hutson. 2021. Robo-writers: the rise and risks of language-generating AI. *Nature*, 591(7848):22–25.

IEA. 2022. *Global Energy Review: CO2 Emissions in 2021*.

Itay Itzhak and Omer Levy. 2022. Models in a spelling bee: Language models implicitly learn the character composition of tokens. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5061–5068, Seattle, United States. Association for Computational Linguistics.

Shoichi Iwasaki and Preeya Ingkaphirom. 2005. A reference grammar of thai.

Román Jakobson and Morris Halle. 1956. Fundamentals of language.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *Tree-Based Methods*, pages 303–335.

Ganesh Jawahar, Benjamin Muller, Amal Fethi, Louis Martin, Éric Villemonte de la Clergerie, Benoît Sagot, and Djamé Seddah. 2018. ELMoLex: Connecting ELMo and lexicon features for dependency parsing. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 223–237, Brussels, Belgium. Association for Computational Linguistics.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In (Korhonen et al., 2019), pages 3651–3657.

Edwin T. Jaynes. 1957. Information theory and statistical mechanics. *Physical Review*, 106:620–630.

Frederick Jelinek. 1976. Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64:532–556.

Frederick Jelinek. 1980. Interpolated estimation of markov source parameters from sparse data.

Kåre Jean Jensen and Søren Kamaric Riis. 2000. Self-organizing letter code-book for text-to-phoneme neural network model. In *INTERSPEECH*.

Jespersen. 2013. *The Articulations of Speech Sounds Represented by Means of Analphabetic Symbols*.

Zhengbao Jiang, Frank F. Xu, J. Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Mark Johnson. 2001. Joint and conditional estimation of tagging and parsing models. In *ACL*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2019. Spanbert: Improving pre-training by representing and predicting spans. *CoRR*, abs/1907.10529.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020a. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020b. The state and fate of linguistic diversity and inclusion in the NLP world. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6282–6293, Online. Association for Computational Linguistics.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hervé Jégou, and Tomas Mikolov. 2016. Fasttext.zip: Compressing text classification models. ArXiv preprint 1612.03651.

Norman P. Jouppi, Doe Hyun Yoon, Matthew Ashcraft, Mark Gottscho, Thomas B. Jablin, George Kurian, James Laudon, Sheng Li, Peter Ma, Xiaoyu Ma, Thomas Norrie,

Nishant Patil, Sushma Prasad, Cliff Young, Zongwei Zhou, and David Patterson. 2021. *Ten Lessons from Three Generations Shaped Google's TPUv4i*, page 1–14. IEEE Press.

Norman P. Jouppi, Cliff Young, Nishant Patil, David A. Patterson, Gaurav Agrawal, Raminder Singh Bajwa, Sarah Bates, Suresh, Bhatia, Nanette J. Boden, Al Borchers, Rick Boyle, Pierre luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert B. Hagmann, C., Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Daniel Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, Diemthu Le, Chris Leary, Zhuyuan, Liu, Kyle A. Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi, Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter, Wang, and Eric Wilcox. In-datacenter performance analysis of a tensor processing unittm.

Jumanto Jumanto. 2014. Phatic communication: How english native speakers create ties of union. *Linguistics*, 3:9–16.

Dan Jurafsky. 2018. *Speech & language processing*, 3rd edition. Currently in draft.

Daniel Jurafsky and James H. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 1st edition. Prentice Hall PTR, USA.

Eivind Kahrs. 1990. Sumitra mangesh katre (ed. and tr.): The aááādhyāyī of pāáini in romanised transliteration. (texas linguistic series.) xlvi, 1330 pp. austin: University of texas press, 1987. usd100. *Bulletin of the School of Oriental and African Studies*, 53(3):531–533.

Jared Kaplan, Sam McCandlish, T. J. Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeff Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *ArXiv*, abs/2001.08361.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Yu Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense passage retrieval for open-domain question answering. *ArXiv*, abs/2004.04906.

K Karthikeyan, Zihan Wang, Stephen Mayhew, and Dan Roth. 2019. Cross-lingual ability of multilingual BERT: An empirical study. In *International Conference on Learning Representations*.

Tadao Kasami. 1966. An efficient recognition and syntax-analysis algorithm for context-free languages. *Coordinated Science Laboratory Report no. R-257*.

Tom Kenter, Llion Jones, and Daniel Hewlett, editors. 2018. *Byte-level Machine Reading across Morphologically Varied Languages*.

Brian W Kernighan and Dennis M Ritchie. 1988. *The C programming language*. Pearson Educación.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI conference on artificial intelligence*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

E. Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Sheldon Klein and Robert F. Simmons. 1963. A computational approach to grammatical coding of english words. *J. ACM*, 10(3):334–347.

A. Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *COLING*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of Machine Translation Summit X: Papers*, pages 79–86, Phuket, Thailand.

Daphne Koller and Nir Friedman. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

Dan Kondratyuk and Milan Straka. 2019. 75 languages, 1 model: Parsing Universal Dependencies universally. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2779–2795, Hong Kong, China. Association for Computational Linguistics.

Daniel Kondratyuk. 2019. 75 languages, 1 model: Parsing universal dependencies universally. *CoRR*, abs/1904.02099.

Anna Korhonen, David R. Traum, and Lluís Màrquez, editors. 2019. *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Association for Computational Linguistics.

András Kornai. 2013. Digital language death. *PLoS ONE*, 8.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pages 3519–3529.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 66–75. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018a. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations, Brussels, Belgium, October 31 - November 4, 2018*, pages 66–71. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018b. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Thomas Kuhn, Heinrich Niemann, and Ernst Günter Schukat-Talamazzini. 1994. Ergodic hidden markov models and polygrams for language modeling. *Proceedings of ICASSP '94. IEEE International Conference on Acoustics, Speech and Signal Processing*, i:I/357–I/360 vol.1.

Soumyadeep Kundu, Sayantan Paul, and Santanu Pal. 2018. A deep learning based approach to transliteration. In *Proceedings of the Seventh Named Entities Workshop*, pages 79–83, Melbourne, Australia. Association for Computational Linguistics.

Julian Kupiec. 1993. Murax: A robust linguistic approach for question answering using an on-line encyclopedia. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '93, page 181–190, New York, NY, USA. Association for Computing Machinery.

Henry Kučera and W. Nelson Francis. 1967. *Computational Analysis of Present-Day American English*. Brown University Pres.

Salla-Maaria Laaksonen, Jesse Haapoja, Teemu Kinnunen, Matti Nelimarkka, and Reeta Pöyhtäri. 2020. The datafication of hate: expectations and challenges in automated hate speech monitoring. *Frontiers in big Data*, 3:3.

François Lagunas, Ella Charlaix, Victor Sanh, and Alexander Rush. 2021. Block pruning for faster transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10619–10629, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 260–270. The Association for Computational Linguistics.

Guillaume Lample and François Charton. 2020. Deep learning for symbolic mathematics. In *International Conference on Learning Representations*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *CoRR*, abs/1901.07291.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. *ArXiv*, abs/1909.11942.

Raymond Lau, Ronald Rosenfeld, and Salim Roukos. 1993. Adaptive language modeling using the maximum entropy principle. In *Proceedings of the Workshop on Human Language Technology*, HLT '93, page 108–113, USA. Association for Computational Linguistics.

Hai Son Le, Alexandre Allauzen, and François Yvon. 2012. Measuring the influence of long range dependencies with neural network language models. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 1–10, Montréal, Canada. Association for Computational Linguistics.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. In *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann.

Yann LeCun, Ido Kanter, and Sara Solla. 1990. Second order properties of error surfaces: Learning time and generalization. In *Advances in Neural Information Processing Systems*, volume 3. Morgan-Kaufmann.

CL Lee. 2003. Motivations of code-switching in multi-lingual singapore. *Journal of Chinese Linguistics*, 31:145–176.

David Yong Wey Lee. 2001. Genres, registers, text types, domains and styles: Clarifying the concepts and navigating a path through the bnc jungle. *Language Learning & Technology*, 5:37–72.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378.

Gurpreet Singh Lehal and Tejinder Singh Saini. 2012. Development of a complete Urdu-Hindi transliteration system. In *Proceedings of COLING 2012: Posters*, pages 643–652, Mumbai, India. The COLING 2012 Organizing Committee.

Wendy G. Lehnert. 1977. A conceptual theory of question answering. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'77, page 158–164, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Chen Li and Yang Liu. 2014. Improving text normalization via unsupervised model and discriminative reranking. In *Proceedings of the ACL 2014 Student Research Workshop*, pages 86–93, Baltimore, Maryland, USA. Association for Computational Linguistics.

Chen Li and Yang Liu. 2015. Joint pos tagging and text normalization for informal text. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 1263–1269. AAAI Press.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Jindřich Libovickỳ, Rudolf Rosa, and Alexander Fraser. 2019. How language-neutral is multilingual bert? *arXiv preprint arXiv:1911.03310*.

Rochelle Lieber. 2009. *Inflection*, Cambridge Introductions to Language and Linguistics, page 87–116. Cambridge University Press.

Ying Lin, Xiaoman Pan, Aliya Deri, Heng Ji, and Kevin Knight. 2016. Leveraging entity linking and related language projection to improve name transliteration. In *Proceedings of the Sixth Named Entity Workshop*, pages 1–10, Berlin, Germany. Association for Computational Linguistics.

John M Lipski. 2014. Spanish-english code-switching among low-fluency bilinguals: Towards an expanded typology. *Sociolinguistic Studies*, 8(1):23.

Pierre Lison and Jörg Tiedemann. 2016. OpenSubtitles2016: Extracting large parallel corpora from movie and TV subtitles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 923–929, Portorož, Slovenia. European Language Resources Association (ELRA).

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ArXiv*, abs/2107.13586.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Eugene E. Loos, Jr. Paul C. Jordan Susan Anderson, Dwight H. Day, and J. Douglas Wingate. 2003. *GLOSSARY OF LINGUISTIC TERMS*. SIL International, Dallas, TX, USA.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *ICLR*.

Jiaming Luo, Yuan Cao, and Regina Barzilay. 2019. Neural decipherment via minimum-cost flow: from ugaritic to linear b. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, page 3146–3155.

Bill MacCartney and Christopher D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 521–528, Manchester, UK. Coling 2008 Organizing Committee.

Ian Maddieson. 2013. Consonant inventories. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Mandy J. Maguire, Julie M. Schneider, Anna E Middleton, Yvonne K Ralph, Michael Lopez, Robert A. Ackerman, and Alyson D. Abel. 2018. Vocabulary knowledge mediates the link between socioeconomic status and word learning in grade school. *Journal of experimental child psychology*, 166:679–695.

Bill Manaris. 1998a. Natural language processing: A human-computer interaction perspective. *Advances in Computers*, 47:1–66.

Bill Manaris. 1998b. Natural language processing: A human-computer interaction perspective. *Advances in Computers*, 47:1–66.

Christopher D. Manning and Hinrich Schütze. 2002. Foundations of statistical natural language processing. In *SGMD*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Louis Martin. 2021. *Automatic sentence simplification using controllable and unsupervised methods*. Ph.D. thesis, Sorbonne Université.

Louis Martin, Éric de la Clergerie, Benoît Sagot, and Antoine Bordes. 2020. Controllable sentence simplification. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4689–4698, Marseille, France. European Language Resources Association.

Louis Martin*, Benjamin Muller*, Pedro Javier Ortiz Suárez*, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. 2020. CamemBERT: a tasty French language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7203–7219, Online. Association for Computational Linguistics.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. 2020. CamemBERT: a tasty French language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7203–7219, Online. Association for Computational Linguistics.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric Villemonte de la Clergerie, Djamé Seddah, and Benoît Sagot. 2019. CamemBERT: a Tasty French Language Model. *arXiv e-prints*. ArXiv preprint : 1911.03894.

Calvin R. Massey. 1992. Hate speech, cultural diversity, and the foundational paradigms of free expression. *UCLA Law Review*, 40:103.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *ArXiv*, abs/1806.08730.

John McCarthy, Marvin Minsky, Nathaniel Rochester, and Claude E. Shannon. A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI Mag.*

Warren S McCulloch and Walter Pitts. 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013a. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97, Sofia, Bulgaria. Association for Computational Linguistics.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013b. Universal Dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97, Sofia, Bulgaria. Association for Computational Linguistics.

Mary McGroarty. 2012. *Language Uses in Professional Contexts*. Oxford University Press.

M. Douglas McIlroy. 1982. Development of a spelling list. *IEEE Trans. Commun.*, 30:91–99.

Jacques Mehler and Emmanuel Dupoux. 2002. *Naître humain*. Odile Jacob.

Amil Merchant, Elahe Rahimtoroghi, Ellie Pavlick, and Ian Tenney. 2020. What happens to BERT embeddings during fine-tuning? In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 33–44, Online. Association for Computational Linguistics.

Bernard Merialdo. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–171.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing lstm language models. *ArXiv*, abs/1708.02182.

Jacob L. Mey. 1993. Pragmatics: An introduction: Jacob l. mey, oxford: Blackwell, 1993. 357 pp. us $ 19.95 (pb.). *Journal of Pragmatics*.

Paul Michel and Graham Neubig. 2018. MTNT: A testbed for machine translation of noisy text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 543–553, Brussels, Belgium. Association for Computational Linguistics.

Vincent Micheli, Martin d'Hoffschmidt, and François Fleuret. 2020. On the importance of pre-training data volume for compact language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7853–7858, Online. Association for Computational Linguistics.

Sabrina J. Mielke, Zaid Alyafeai, Elizabeth Salesky, Colin Raffel, Manan Dey, Matthias Gallé, Arun Raja, Chenglei Si, Wilson Y. Lee, Benoît Sagot, and Samson Tan. 2021. Between words and characters: A brief history of open-vocabulary modeling and tokenization in NLP. *CoRR*, abs/2112.10508.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018.*

Tomas Mikolov, Stefan Kombrink, Lukas Burget, J.H. Cernocky, and Sanjeev Khudanpur. 2011a. Extensions of recurrent neural network language model. pages 5528 – 5531.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119.

Tomas Mikolov, Ilya Sutskever, Anoop Deoras, Hai Son Le, Stefan Kombrink, and Jan Honza Cernocký. 2011b. Subword language modeling with neural networks.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Marvin Minsky and Papert Seymour. 1969. Perceptrons, an introduction to computational geometry.

David R. Mortensen, Siddharth Dalmia, and Patrick Littell. 2018. Epitran: Precision G2P for many languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France. European Language Resources Association (ELRA).

Benjamin Muller, Antonios Anastasopoulos, Benoît Sagot, and Djamé Seddah. 2021a. When being unseen from mBERT is just the beginning: Handling new languages with multilingual language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 448–462, Online. Association for Computational Linguistics.

Benjamin Muller, Antonis Anastasopoulos, Benoît Sagot, and Djamé Seddah. 2020a. When being unseen from mbert is just the beginning: Handling new languages with multilingual language models. *CoRR*, abs/2010.12858.

Benjamin Muller, Yanai Elazar, Benoît Sagot, and Djamé Seddah. 2021b. First align, then predict: Understanding the cross-lingual ability of multilingual BERT. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2214–2231, Online. Association for Computational Linguistics.

Benjamin Muller, Benoit Sagot, and Djamé Seddah. 2019. Enhancing BERT for lexical normalization. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 297–306, Hong Kong, China. Association for Computational Linguistics.

Benjamin Muller, Benoit Sagot, and Djamé Seddah. 2020b. Can multilingual language models transfer to an unseen dialect? a case study on north african arabizi. *arXiv preprint arXiv:2005.00318*.

Benjamin Muller, Benoît Sagot, and Djamé Seddah. 2020c. Can multilingual language models transfer to an unseen dialect? A case study on north african arabizi. *CoRR*, abs/2005.00318.

Benjamin Muller, Luca Soldaini, Rik Koncel-Kedziorski, Eric Lind, and Alessandro Moschitti. 2021c. Cross-lingual genqa: A language-agnostic generative question answering approach for open-domain question answering. *arXiv preprint arXiv:2110.07150*.

Nikitha Murikinati, Antonios Anastasopoulos, and Graham Neubig. 2020. Transliteration for cross-lingual morphological inflection. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 189–197, Online. Association for Computational Linguistics.

Masami Nakamura and Kiyohiro Shikano. 1988. A study of english word category prediction based on neural networks. *Journal of the Acoustical Society of America*, 84.

Preslav Nakov and Jörg Tiedemann. 2012. Combining word-level and character-level models for machine translation between closely-related languages. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 301–305, Jeju Island, Korea. Association for Computational Linguistics.

Ani Nenkova, Sameer Maskey, and Yang Liu. 2011. Automatic summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, page 3, Portland, Oregon. Association for Computational Linguistics.

Hwee Tou Ng, Leong Hwee Teo, and Jennifer Lai Pheng Kwan. 2000. A machine learning approach to answering questions for reading comprehension tests. In *2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 124–132, Hong Kong, China. Association for Computational Linguistics.

John Nickolls and William J. Dally. 2010. The gpu computing era. *IEEE Micro*, 30(2):56–69.

Sergei Nirenburg and Victor Raskin. 2000. Ontological semantics. In *AMTA*.

Joakim Nivre, Mitchell Abrams, Željko Agić, Lars Ahrenberg, Lene Antonsen, Maria Jesus Aranzabe, Gashaw Arutie, Masayuki Asahara, Luma Ateyah, Mohammed Attia,

Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, John Bauer, Sandra Bellato, Kepa Bengoetxea, Riyaz Ahmad Bhat, Erica Biagetti, Eckhard Bick, Rogier Blokland, Victoria Bobicev, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Adriane Boyd, Aljoscha Burchardt, Marie Candito, Bernard Caron, Gauthier Caron, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Savas Cetin, Fabricio Chalub, Jinho Choi, Yongseok Cho, Jayeol Chun, Silvie Cinková, Aurélie Collomb, Çağrı Çöltekin, Miriam Connor, Marine Courtin, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Carly Dickerson, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Ali Elkahky, Binyam Ephrem, Tomaž Erjavec, Aline Etienne, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Céline Guillot-Barbance, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà Mỹ, Na-Rae Han, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Jena Hwang, Radu Ion, Elena Irimia, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Sylvain Kahane, Hiroshi Kanayama, Jenna Kanerva, Tolga Kayadelen, Václava Kettnerová, Jesse Kirchner, Natalia Kotsyba, Simon Krek, Sookyoung Kwak, Veronika Laippala, Lorenzo Lambertino, Tatiana Lando, Septina Dian Larasati, Alexei Lavrentiev, John Lee, Phng Lê Hồng, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, KyungTae Lim, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Ruli Manurung, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Niko Miekka, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Shinsuke Mori, Bjartur Mortensen, Bohdan Moskalevskyi, Kadri Muischnek, Yugo Murawaki, Kaili Müürisep, Pinkey Nainwani, Juan Ignacio Navarro Horñiacek, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lng Nguyễn Thị, Huyền Nguyễn Thị Minh, Vitaly Nikolaev, Rattima Nitisaroj, Hanna Nurmi, Stina Ojala, Adédayọ̀ Olúòkun, Mai Omura, Petya Osenova, Robert Östling, Lilja Øvrelid, Niko Partanen, Elena Pascual, Marco Passarotti, Agnieszka Patejuk, Siyao Peng, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Thierry Poibeau, Martin Popel, Lauma Pretkalniņa, Sophie Prévost, Prokopis Prokopidis, Adam Przepiórkowski, Tiina Puolakainen, Sampo Pyysalo, Andriela Rääbis, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Carlos Ramisch, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Michael Rießler, Larissa Rinaldi, Laura Rituma, Luisa Rocha, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Valentin Roșca, Olga Rudina, Shoval Sadde, Shadi Saleh, Tanja Samardžić, Stephanie

Samson, Manuela Sanguinetti, Baiba Saulīte, Yanin Sawanakunanon, Nathan Schneider, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Muh Shohibussirri, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Isabela Soares-Bastos, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Yuta Takahashi, Takaaki Tanaka, Isabelle Tellier, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Veronika Vincze, Lars Wallin, Jonathan North Washington, Seyi Williams, Mats Wirén, Tsegay Woldemariam, Tak-sum Wong, Chunxiao Yan, Marat M. Yavrumyan, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, Manying Zhang, and Hanzhi Zhu. 2018. Universal dependencies 2.2. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Joakim Nivre, Željko Agić, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Cristina Bosco, Sam Bowman, Giuseppe G. A. Celano, Miriam Connor, Marie-Catherine de Marneffe, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Daniel Galbraith, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Berta Gonzales, Bruno Guillaume, Jan Hajič, Dag Haug, Radu Ion, Elena Irimia, Anders Johannsen, Hiroshi Kanayama, Jenna Kanerva, Simon Krek, Veronika Laippala, Alessandro Lenci, Nikola Ljubešić, Teresa Lynn, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Shunsuke Mori, Hanna Nurmi, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Prokopis Prokopidis, Sampo Pyysalo, Loganathan Ramasamy, Rudolf Rosa, Shadi Saleh, Sebastian Schuster, Wolfgang Seeker, Mojgan Seraji, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Kiril Simov, Aaron Smith, Jan Štěpánek, Alane Suhr, Zsolt Szántó, Takaaki Tanaka, Reut Tsarfaty, Sumire Uematsu, Larraitz Uria, Viktor Varga, Veronika Vincze, Zdeněk Žabokrtský, Daniel Zeman, and Hanzhi Zhu. 2015. Universal dependencies 1.2. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666.

J. Norman, S.R. Anderson, J. Bresnan, B. Comrie, W. Dressler, C. Ewen, and R. Lass. 1988. *Chinese*. ACLS Humanities E-Book. Cambridge University Press.

Pedro Ortiz Suarez. 2022. *A Data-driven Approach to Natural Language Processing for Contemporary and Historical French*. Theses, Sorbone Université.

Pedro Javier Ortiz Suárez, Benoît Sagot, and Laurent Romary. 2019. Asynchronous pipeline for processing huge corpora on medium to low resource infrastructures. *Challenges in the Management of Large Corpora (CMLC-7) 2019*, page 9.

Pedro Javier Ortiz Suárez, Benoît Sagot, and Laurent Romary. 2019. Asynchronous pipelines for processing huge corpora on medium to low resource infrastructures. Proceedings of the Workshop on Challenges in the Management of Large Corpora (CMLC-7) 2019. Cardiff, 22nd July 2019, pages 9 – 16, Mannheim. Leibniz-Institut für Deutsche Sprache.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Demonstrations*, pages 48–53. Association for Computational Linguistics.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. 2022. Training language models to follow instructions with human feedback. *ArXiv*, abs/2203.02155.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958, Vancouver, Canada. Association for Computational Linguistics.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, page III–1310–III–1318. JMLR.org.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*.

David A. Patterson, Joseph Gonzalez, Urs Holzle, Quoc Le, Chen Liang, Lluís-Miquel Munguía, Daniel Rothchild, David R. So, Maud Texier, and Jeff Dean. 2022. The carbon footprint of machine learning training will plateau, then shrink. *ArXiv*, abs/2204.05149.

Ellie Pavlick and Joel Tetreault. 2016. An empirical analysis of formality in online communication. *Transactions of the Association for Computational Linguistics*, 4:61–74.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018b. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018c. Deep contextualized word representations. In *Proc. of NAACL*.

Slav Petrov, Dipanjan Das, and Ryan T. McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources*

and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012, pages 2089–2096. European Language Resources Association (ELRA).

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2021. UNKs everywhere: Adapting multilingual language models to new scripts. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10186–10203, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Isabelle Pierozak. 2003. Le "français tchaté" : un objet à géométrie variable ? *Langage et société*, 104.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019a. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019b. How multilingual is multilingual bert? *arXiv preprint arXiv:1906.01502*.

Barbara Plank. 2016. What to do about non-standard (or non-canonical) language in nlp. *ArXiv*, abs/1608.07836.

Barbara Plank, Héctor Martínez Alonso, Željko Agić, Danijela Merkler, and Anders Søgaard. 2015. Do dependency parsing metrics correlate with human judgments? In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 315–320, Beijing, China. Association for Computational Linguistics.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418, Berlin, Germany. Association for Computational Linguistics.

David M. W. Powers. 1998. Applications and explanations of Zipf's law. In *New Methods in Language Processing and Computational Natural Language Learning*.

Ofir Press, Noah Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. BPE-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892, Online. Association for Computational Linguistics.

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D. Manning. 2018. Universal Dependency parsing from scratch. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 160–170, Brussels, Belgium. Association for Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020a. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020b. Stanza: A python natural language processing toolkit for many human languages. In *ACL*.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5203–5212, Online. Association for Computational Linguistics.

J.R. Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann series in machine learning. Elsevier Science.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *OpenAI Blog*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John F. J. Mellor, Irina Higgins, Antonia Creswell, Nathan McAleese, Amy Wu, Erich Elsen, Siddhant M. Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, L. Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, N. K. Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Tobias Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew G. Johnson, Blake A. Hechtman, Laura Weidinger, Iason Gabriel, William S. Isaac, Edward Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem W. Ayoub, Jeff Stanway, L. L. Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. 2021. Scaling language models: Methods, analysis & insights from training gopher. *ArXiv*, abs/2112.11446.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. ArXiv preprint 1910.10683.

Afshin Rahimi, Yuan Li, and Trevor Cohn. 2019. Massively multilingual transfer for NER. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 151–164, Florence, Italy. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. Zero-shot text-to-image generation. *ArXiv*, abs/2102.12092.

Kanishka Rao, Fuchun Peng, Hasim Sak, and Françoise Beaufays. 2015. Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4225–4229.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Conference on Empirical Methods in Natural Language Processing*.

Abhilasha Ravichander, Yonatan Belinkov, and Eduard Hovy. 2020. Probing the probing paradigm: Does probing accuracy entail task relevance? *arXiv preprint arXiv:2005.00719*.

Anjani Kumar Ray, Shamim Fatma, and Vijay Kumar kaul. 2022. Devanagari to urdu transliteration system with and without airaab. In *Sustainable Advanced Computing*, pages 415–425, Singapore. Springer Singapore.

Luz Rello, Ricardo Baeza-Yates, Stefan Bott, and Horacio Saggion. 2013. Simplify or help? text simplification strategies for people with dyslexia.

Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*.

Arij Riabi, Benoît Sagot, and Djamé Seddah. 2021. Can character-based language models improve downstream task performances in low-resource and noisy language scenarios? In *Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021)*, pages 423–436, Online. Association for Computational Linguistics.

Søren Riis and Anders Krogh. 1996. Improving prediction of protein secondary structure using structured neural networks and multiple sequence alignments. *Journal of computational biology : a journal of computational molecular cell biology*, 3:163–83.

Shruti Rijhwani, Jiateng Xie, Graham Neubig, and Jaime Carbonell. 2019. Zero-shot neural transfer for cross-lingual entity linking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6924–6931.

Dennis M Ritchie, Stephen C Johnson, ME Lesk, BW Kernighan, et al. 1978. The c programming language. *Bell Sys. Tech. J*, 57(6):1991–2019.

Brian Roark, Lawrence Wolf-Sonkin, Christo Kirov, Sabrina J. Mielke, Cibu Johny, Isin Demirsahin, and Keith Hall. 2020. Processing South Asian languages written in the Latin script: the Dakshina dataset. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 2413–2423, Marseille, France. European Language Resources Association.

Herbert Robbins and Sutton Monro. 1951. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Frank Rosenblatt. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Carl Rubino, Bernard Comrie, Matthew Dryer, and Martin Haspelmath. 2002. Reduplication: Form, function and distribution.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2019. A survey of cross-lingual word embedding models. *Journal of Artificial Intelligence Research*, 65:569–631.

Pablo Ruiz, Montse Cuadros, and Thierry Etchegoyhen. 2014. Lexical normalization of spanish tweets with rule-based components and language models. *Procesamiento del Lenguaje Natural*, page 8.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1985. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

Benoît Sagot. 2010. The lefff, a freely available and large-coverage morphological and syntactic lexicon for French. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Benoît Sagot and Héctor Martínez Alonso. 2017. Improving neural tagging with lexical information. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 25–31, Pisa, Italy. Association for Computational Linguistics.

Benoît Sagot, Marion Richard, and Rosa Stern. 2012. Annotation référentielle du corpus arboré de Paris 7 en entités nommées (referential named entity annotation of the paris 7 french treebank) [in french]. In *Proceedings of the Joint Conference JEP-TALN-RECITAL 2012, volume 2: TALN, Grenoble, France, June 4-8, 2012*, pages 535–542. ATALA/AFCP.

David Samuel and Milan Straka. 2021. ÚFAL at MultiLexNorm 2021: Improving multilingual lexical normalization by fine-tuning ByT5. In *Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021)*, pages 483–492, Online. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M. Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan Chang,

Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Févry, Jason Alan Fries, Ryan Teehan, Stella Biderman, Leo Gao, Tali Bers, Thomas Wolf, and Alexander M. Rush. 2021a. Multitask prompted training enables zero-shot task generalization. *CoRR*, abs/2110.08207.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021b. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Edward Sapir. 1968. *Language an Introduction to the Study of Speech.* –. Harcourt, Brace & World.

Ferdinand de Saussure. 1916. *Course in general linguistics*. Columbia University Press.

Beatrice Savoldi, Marco Gaido, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2021. Gender Bias in Machine Translation. *Transactions of the Association for Computational Linguistics*, 9:845–874.

Sayahi. 2014. *The languages of the Maghreb. In Diglossia and Language Contact: Language Variation and Change in North Africa*. Cambridge: Cambridge University Press.

Lotfi Sayahi. 2011. Code-switching and language change in tunisia. *International Journal of the Sociology of Language*, 2011.

Jürgen Schmidhuber and Stefan Heil. 1996. Sequential neural text compression. *IEEE transactions on neural networks*, 7 1:142–6.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10, Valencia, Spain. Association for Computational Linguistics.

Helmut Schmidt. 1994. Probabilistic part-of-speech tagging using decision trees.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152. IEEE.

Djamé Seddah, Farah Essaidi, Amal Fethi, Matthieu Futeral, Benjamin Muller, Pedro Javier Ortiz Suárez, Benoît Sagot, and Abhishek Srivastava. 2020. Building a user-generated content North-African Arabizi treebank: Tackling hell. In *Proceedings*

of the 58th Annual Meeting of the Association for Computational Linguistics, pages 1139–1150, Online. Association for Computational Linguistics.

Djamé Seddah, Benoît Sagot, Marie Candito, Virginie Mouilleron, and Vanessa Combet. 2012. The French Social Media Bank: a Treebank of Noisy User Generated Content. In *CoLing*, Mumbai, India.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

Ali Shazal, Aiza Usman, and Nizar Habash. 2020. A unified model for Arabizi detection and transliteration using sequence-to-sequence models. In *Proceedings of the Fifth Arabic Natural Language Processing Workshop*, pages 167–177, Barcelona, Spain (Online). Association for Computational Linguistics.

Aditya Siddhant, Ankur Bapna, Orhan Firat, Yuan Cao, Mia Xu Chen, Isaac Caswell, and Xavier Garcia. 2022. Towards the next 1000 languages in multilingual machine translation: Exploring the synergy between supervised and self-supervised learning. *arXiv preprint arXiv:2201.03110*.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Christopher D. Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

Jasdeep Singh, Bryan McCann, Richard Socher, and Caiming Xiong. 2019. Bert is not an interlingua and the bias of tokenization. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 47–55.

Amit Singhal and I. Google. 2001. Modern information retrieval: A brief overview. *IEEE Data Engineering Bulletin*, 24.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Shaden Smith, Mostofa Ali Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Anand Korthikanti, Elton Zhang, Rewon Child, Reza Yazdani Aminabadi, Julie Bernauer, Xia Song, Mohammad Shoeybi, Yuxiong He, Michael Houston, Saurabh Tiwary, and Bryan Catanzaro. 2022. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *ArXiv*, abs/2201.11990.

Anders Søgaard. 2011. Data point selection for cross-language adaptation of dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 682–686. Association for Computational Linguistics.

Wee Meng Soon, Hwee Tou Ng, and Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27:521–544.

Aarohi Srivastava, Abhinav Rastogi, Abhishek B Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, Alice Xiang, Alicia Parrish, Allen Nie, Aman Hussain, Amanda Askell, Amanda Dsouza, Ameet Annasaheb Rahane, Anantharaman S. Iyer, Anders Johan Andreassen, Andrea Santilli, Andreas Stuhlmuller, Andrew M. Dai, Andrew D. La, Andrew Kyle Lampinen, Andy Zou, Angela Jiang, Angelica Chen, Anh Vuong, Animesh Gupta, Anna Gottardi, Antonio Norelli, Anu Venkatesh, Arash Gholamidavoodi, Arfa Tabassum, Arul Menezes, Arun Kirubarajan, Asher Mullokandov, Ashish Sabharwal, Austin Herrick, Avia Efrat, Aykut Erdem, Ayla Karakacs, Bridget R. Roberts, Bao Sheng Loe, Barret Zoph, Bartlomiej Bojanowski, Batuhan Ozyurt, Behnam Hedayatnia, Behnam Neyshabur, Benjamin Inden, Benno Stein, Berk Ekmekci, Bill Yuchen Lin, Blake Stephen Howald, Cameron Diao, Cameron Dour, Catherine Stinson, Cedrick Argueta, C'esar Ferri Ram'irez, Chandan Singh, Charles Rathkopf, Chenlin Meng, Chitta Baral, Chiyu

Wu, Chris Callison-Burch, Chris Waites, Christian Voigt, Christopher D. Manning, Christopher Potts, Cindy Tatiana Ramirez, Clara Rivera, Clemencia Siro, Colin Raffel, Courtney Ashcraft, Cristina Garbacea, Damien Sileo, Daniel H Garrette, Dan Hendrycks, Dan Kilman, Dan Roth, Daniel Freeman, Daniel Khashabi, Daniel Levy, Daniel Gonz'alez, Danny Hernandez, Danqi Chen, Daphne Ippolito, Dar Gilboa, David Dohan, D. Drakard, David Jurgens, Debajyoti Datta, Deep Ganguli, Denis Emelin, Denis Kleyko, Deniz Yuret, Derek Chen, Derek Tam, Dieuwke Hupkes, Diganta Misra, Dilyar Buzan, Dimitri Coelho Mollo, Diyi Yang, Dong-Ho Lee, Ekaterina Shutova, Ekin Dogus Cubuk, Elad Segal, Eleanor Hagerman, Elizabeth Barnes, Elizabeth P. Donoway, Ellie Pavlick, Emanuele Rodolà, Emma FC Lam, Eric Chu, Eric Tang, Erkut Erdem, Ernie Chang, Ethan A. Chi, Ethan Dyer, Ethan Jerzak, Ethan Kim, Eunice Engefu Manyasi, Evgenii Zheltonozhskii, Fan Xia, Fatemeh Siar, Fernando Mart'inez-Plumed, Francesca Happ'e, François Chollet, Frieda Rong, Gaurav Mishra, Genta Indra Winata, Gerard de Melo, Germán Kruszewski, Giambattista Parascandolo, Giorgio Mariani, Gloria Wang, Gonzalo Jaimovitch-L'opez, Gregor Betz, Guy Gur-Ari, Hana Galijasevic, Han Sol Kim, Hannah Rashkin, Hanna Hajishirzi, Harsh Mehta, Hayden Bogar, Henry Shevlin, Hinrich Schütze, Hiromu Yakura, Hongming Zhang, Hubert Wong, Ian Aik-Soon Ng, Isaac Noble, Jaap Jumelet, Jack Geissinger, John Kernion, Jacob Hilton, Jaehoon Lee, Jaime Fernández Fisac, J. Brooker Simon, James Koppel, James Zheng, James Zou, Jan Koco'n, Jana Thompson, Jared Kaplan, Jarema Radom, Jascha Narain Sohl-Dickstein, Jason Phang, Jason Wei, Jason Yosinski, Jekaterina Novikova, Jelle Bosscher, Jenni Marsh, Jeremy Kim, Jeroen Taal, Jesse Engel, Jesujoba Oluwadara Alabi, Jiacheng Xu, Jiaming Song, Jillian Tang, Jane W Waweru, John Burden, John Miller, John U. Balis, Jonathan Berant, Jorg Frohberg, Jos Rozen, José Hernández-Orallo, Joseph Boudeman, Joseph Jones, Joshua B. Tenenbaum, Joshua S. Rule, Joyce Chua, Kamil Kanclerz, Karen Livescu, Karl Krauth, Karthik Gopalakrishnan, Katerina Ignatyeva, Katja Markert, Kaustubh D. Dhole, Kevin Gimpel, Kevin Ochieng' Omondi, Kory Wallace Mathewson, Kristen Chiafullo, Ksenia Shkaruta, Kumar Shridhar, Kyle McDonell, Kyle Richardson, Laria Reynolds, Leo Gao, Li Zhang, Liam Dugan, Lianhui Qin, Lidia Contreras-Ochando, Louis-Philippe Morency, Luca Moschella, Luca Lam, Lucy Noble, Ludwig Schmidt, Luheng He, Luis Oliveros Col'on, Luke Metz, Lutfi Kerem cSenel, Maarten Bosma, Maarten Sap, Maartje ter Hoeve, Madotto Andrea, Maheen Saleem Farooqi, Manaal Faruqui, Mantas Mazeika, Marco Baturan, Marco Marelli, Marco Maru, M Quintana, Marie Tolkiehn, Mario Giulianelli, Martha Lewis, Martin Potthast, Matthew Leavitt, Matthias Hagen, M'aty'as Schubert, Medina Baitemirova, Melissa Arnaud, Melvin Andrew McElrath, Michael A. Yee, Michael Cohen, Mi Gu, Michael I. Ivanitskiy, Michael Starritt, Michael Strube, Michal Swkedrowski, Michele Bevilacqua, Michihiro Yasunaga, Mihir Kale, Mike Cain, Mimee Xu, Mirac Suzgun, Monica Tiwari, Mohit Bansal, Moin Aminnaseri, Mor Geva, Mozhdeh Gheini, T MukundVarma, Nanyun Peng, Nathan Chi, Nayeon Lee, Neta Gur-Ari Krakover, Nicholas Cameron,

Nicholas S. Roberts, Nicholas Doiron, Nikita Nangia, Niklas Deckers, Niklas Muennighoff, Nitish Shirish Keskar, Niveditha Iyer, Noah Constant, Noah Fiedel, Nuan Wen, Oliver Zhang, Omar Agha, Omar Elbaghdadi, Omer Levy, Owain Evans, Pablo Antonio Moreno Casares, Parth Doshi, Pascale Fung, Paul Pu Liang, Paul Vicol, Pegah Alipoormolabashi, Peiyuan Liao, Percy Liang, Peter W. Chang, Peter Eckersley, Phu Mon Htut, Pi-Bei Hwang, P. Milkowski, Piyush S. Patil, Pouya Pezeshkpour, Priti Oli, Qiaozhu Mei, QING LYU, Qinlang Chen, Rabin Banjade, Rachel Etta Rudolph, Raefer Gabriel, Rahel Habacker, Ram'on Risco Delgado, Raphaël Millière, Rhythm Garg, Richard Barnes, Rif A. Saurous, Riku Arakawa, Robbe Raymaekers, Robert Frank, Rohan Sikand, Roman Novak, Roman Sitelew, Ronan Le Bras, Rosanne Liu, Rowan Jacobs, Rui Zhang, Ruslan Salakhutdinov, Ryan Chi, Ryan Lee, Ryan Stovall, Ryan Teehan, Rylan Yang, Sahib J. Singh, Saif M. Mohammad, Sajant Anand, Sam Dillavou, Sam Shleifer, Sam Wiseman, Samuel Gruetter, Sam Bowman, Samuel S. Schoenholz, Sanghyun Han, Sanjeev Kwatra, Sarah A. Rous, Sarik Ghazarian, Sayan Ghosh, Sean Casey, Sebastian Bischoff, Sebastian Gehrmann, Sebastian Schuster, Sepideh Sadeghi, Shadi Sameh Hamdan, Sharon Zhou, Shashank Srivastava, Sherry Shi, Shikhar Singh, Shima Asaadi, Shixiang Shane Gu, Shubh Pachchigar, Shubham Toshniwal, Shyam Upadhyay, Shyamolima Debnath, Siamak Shakeri, Simon Thormeyer, Simone Melzi, Siva Reddy, Sneha Priscilla Makini, Soo hwan Lee, Spencer Bradley Torene, Sriharsha Hatwar, Stanislas Dehaene, Stefan Divic, Stefano Ermon, Stella Rose Biderman, Stephanie C. Lin, Stephen Prasad, Steven T. Piantadosi, Stuart M. Shieber, Summer Misherghi, Svetlana Kiritchenko, Swaroop Mishra, Tal Linzen, Tal Schuster, Tao Li, Tao Yu, Tariq A. Ali, Tatsuo Hashimoto, Te-Lin Wu, Theo Desbordes, Theodore Rothschild, Thomas Phan, Tianle Wang, Tiberius Nkinyili, Timo Schick, T. N. Kornev, Timothy Telleen-Lawton, Titus Tunduny, Tobias Gerstenberg, Trenton Chang, Trishala Neeraj, Tushar Khot, Tyler O. Shultz, Uri Shaham, Vedant Misra, Vera Demberg, Victoria Nyamai, Vikas Raunak, Vinay V. Ramasesh, Vinay Uday Prabhu, Vishakh Padmakumar, Vivek Srikumar, William Fedus, William Saunders, William Zhang, W Vossen, Xiang Ren, Xiaoyu F Tong, Xinyi Wu, Xudong Shen, Yadollah Yaghoobzadeh, Yair Lakretz, Yang Song, Yasaman Bahri, Ye Ji Choi, Yichi Yang, Yiding Hao, Yifu Chen, Yonatan Belinkov, Yu Hou, Yu Hou, Yushi Bai, Zachary Seid, Zhao Xinran, Zhuoye Zhao, Zi Fu Wang, Zijie J. Wang, Zirui Wang, Ziyi Wu, Sahib Singh, and Uri Shaham. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *ArXiv*, abs/2206.04615.

Nitish Srivastava. 2013. Improving neural networks with dropout.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Steve Stecklow. 2018. Why Facebook is losing the war on hate speech in Myanmar, Reuters. https://www.reuters.com/investigates/special-report/myanmar-facebook-hate/.

Gustaf Stern. 1975. Meaning and change of meaning: with special reference to the english language.

G. W. Stewart. 1993. On the early history of the singular value decomposition. *SIAM Rev.*, 35:551–566.

Walter S. Stolz, Percy H. Tannenbaum, and Frederick V. Carstensen. 1965. Stochastic approach to the grammatical coding of english. *Commun. ACM*, 8(6):399–405.

Milan Straka. 2018. Udpipe 2.0 prototype at conll 2018 ud shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207.

Milan Straka, Jana Straková, and Jan Hajic. 2019. Evaluating contextualized embeddings on 54 languages in POS tagging, lemmatization and dependency parsing. ArXiv preprint 1908.07448.

Jana Straková, Milan Straka, and Jan Hajic. 2019. Neural architectures for nested NER through linearization. In (Korhonen et al., 2019), pages 5326–5331.

Bjarne Stroustrup. 1986. The c++ programming language, first edition.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. 2021. Roformer: Enhanced transformer with rotary position embedding. *CoRR*, abs/2104.09864.

Pedro Ortiz Suarez, Yoann Dupont, Benjamin Muller, Laurent Romary, and Benoît Sagot. 2020. Establishing a new state-of-the-art for french named entity recognition. In *LREC*.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *INTERSPEECH*.

Dmitry Supranovich and Viachaslau Patsepnia. 2015. Ihs_rd: Lexical normalization for english tweets. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 78–81.

Harini Suresh and John V. Guttag. 2019. A framework for understanding unintended consequences of machine learning. *ArXiv*, abs/1901.10002.

Ilya Sutskever, James Martens, and Geoffrey Hinton. 2011. Generating text with recurrent neural networks. pages 1017–1024.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Corso Svizzera. 2014. Converting the parallel treebank partut in universal stanford dependencies.

H. Sweet. 1899. *The Practical Study of Languages: A Guide for Teachers and Learners*. Creative Media Partners, LLC.

Derek Tam, Rakesh R. Menon, Mohit Bansal, Shashank Srivastava, and Colin Raffel. 2021. Improving and simplifying pattern exploiting training. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4980–4991, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In (Daelemans and Osborne, 2003), pages 142–147.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, page 173–180, USA. Association for Computational Linguistics.

Kristina Toutanova and Robert Moore. 2002. Pronunciation modeling for improved spelling correction. In *Proceedings of the 40th Annual Meeting of the Association for*

Computational Linguistics, pages 144–151, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Enrica Troiano, Aswathy Velutharambath, and Roman Klinger. 2021. From theories on styles to their transfer in text: Bridging the gap with a hierarchical survey. *CoRR*, abs/2110.15871.

Peter Trudgill. 2000. *Sociolinguistics: An introduction to language and society*. Penguin UK.

Huihsin Tseng, Daniel Jurafsky, and Christopher Manning. 2005. Morphological features help POS tagging of unknown words across language varieties. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*.

Rob van der Goot. 2019. MoNoise: A multi-lingual and easy-to-use lexical normalization tool. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 201–206, Florence, Italy. Association for Computational Linguistics.

Rob van der Goot, Alan Ramponi, Arkaitz Zubiaga, Barbara Plank, Benjamin Muller, Iñaki San Vicente Roncal, Nikola Ljubešić, Özlem Çetinoğlu, Rahmad Mahendra, Talha Çolakoğlu, Timothy Baldwin, Tommaso Caselli, and Wladimir Sidorenko. 2021. MultiLexNorm: A shared task on multilingual lexical normalization. In *Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021)*, pages 493–509, Online. Association for Computational Linguistics.

Rob van der Goot and Gertjan van Noord. 2017. Monoise: modeling noise using a modular normalization system. *arXiv preprint arXiv:1710.03476*.

Rob van der Goot and Gertjan van Noord. 2018. Modeling input uncertainty in neural network dependency parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4984–4991.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Kees Versteegh. 1997. The arabic language.

Marc Vilain and David Day. 1996. Finite-state phrase parsing by rule sequences. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

Andrew J. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theory*, 13:260–269.

Ellen M. Voorhees and Dawn M. Tice. 2000. The TREC-8 question answering track. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC'00)*, Athens, Greece. European Language Resources Association (ELRA).

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems. In *NeurIPS*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Changhan Wang, Kyunghyun Cho, and Jiatao Gu. 2020. Neural machine translation with byte-level subwords. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:9154–9160.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 22–32, Prague, Czech Republic. Association for Computational Linguistics.

Peilu Wang, Yao Qian, Frank K. Soong, Lei He, and Zhao Hai. 2015. Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. *ArXiv*, abs/1510.06168.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, A. Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Gary Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Maitreya Patel, Kuntal Kumar Pal, M. Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Shailaja Keyur Sampat, Savan Doshi, Siddharth Deepak Mishra, Sujan C. Reddy, Sumanta Patro, Tanay Dixit, Xudong Shen, Chitta Baral, Yejin Choi, Hannaneh Hajishirzi, Noah A. Smith, and Daniel Khashabi. 2022. Benchmarking generalization via in-context instructions on 1, 600+ language tasks. *ArXiv*, abs/2204.07705.

Zihan Wang, Stephen Mayhew, Dan Roth, et al. 2019b. Cross-lingual ability of multilingual bert: An empirical study. *arXiv preprint arXiv:1912.07840*.

Ronald Wardhaugh and Janet M Fuller. 2014. *An introduction to sociolinguistics*. John Wiley & Sons.

Dominica Watt. 2009. The identification of the individual through speech.

Warren Weaver. 1952. Translation. In *Proceedings of the Conference on Mechanical Translation*, Massachusetts Institute of Technology.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, et al. 2021a. Ethical and social risks of harm from language models. *arXiv preprint arXiv:2112.04359*.

Laura Weidinger, John F. J. Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, Zachary Kenton, Sande Minnich Brown, William T. Hawkins, Tom Stepleton, Courtney Biles, Abeba Birhane, Julia Haas, Laura Rimell, Lisa Anne Hendricks, William S. Isaac, Sean Legassick, Geoffrey Irving, and Iason Gabriel. 2021b. Ethical and social risks of harm from language models. *ArXiv*, abs/2112.04359.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. 2019. CCNet: Extracting High Quality Monolingual Datasets from Web Crawl Data. ArXiv preprint 1911.00359.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. 2020. CCNet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4003–4012, Marseille, France. European Language Resources Association.

Ruth Wodak and Gertraud Benke. 2017. Gender as a sociolinguistic variable: New perspectives on variation studies. *The handbook of sociolinguistics*, pages 127–150.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. ArXiv preprint 1910.03771.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu,

Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

William A Woods and WOODS WA. 1977. Lunar rocks in natural english: Explorations in natural language question answering.

Ellen Woolford. 1983. Bilingual code-switching and syntactic theory. *Linguistic inquiry*, 14(3):520–536.

Songtao Wu, Shenghua Zhong, and Yan Liu. 2018. Deep residual learning for image steganalysis. *Multimedia tools and applications*, 77(9):10437–10453.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. ByT5: Towards a Token-Free Future with Pre-trained Byte-to-Byte Models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

M. Yaguello. 1981. *Alice au pays du langage: pour comprendre la linguistique*. Seuil.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.

Taha Yasseri, András Kornai, and János Kertész. 2012. A practical approach to language complexity: a wikipedia case study. *PloS one*, 7(11):e48386.

Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. 2021. CrossFit: A few-shot learning challenge for cross-task generalization in NLP. In *Proceedings of the 2021 Conference*

on Empirical Methods in Natural Language Processing, pages 7163–7189, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jihene Younes, Emna Souissi, Hadhemi Achour, and Ahmed Ferchichi. 2018. A sequence-to-sequence based approach for the double transliteration of tunisian dialect. In *ACLING*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *ArXiv*, abs/1409.2329.

Rowan Zellers, Ximing Lu, Jack Hessel, Youngjae Yu, Jae Sung Park, Jize Cao, Ali Farhadi, and Yejin Choi. 2021. MERLOT: Multimodal neural script knowledge models. In *Advances in Neural Information Processing Systems*.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajič, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinková, Jan Hajič jr., Jaroslava Hlaváčová, Václava Kettnerová, Zdeňka Urešová, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria de Paiva, Kira Droganova, Héctor Martínez Alonso, Çağrı Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonça, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. CoNLL 2017 shared task: Multilingual parsing from raw text to Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19, Vancouver, Canada. Association for Computational Linguistics.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models. *ArXiv*, abs/2205.01068.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. Ernie: Enhanced language representation with informative entities. In *ACL*.

Ziqi Zhang and Le Luo. 2019. Hate speech detection: A solved problem? the challenging case of long tail on twitter. *Semantic Web*, 10:925–945.

GuoDong Zhou and Jian Su. 2002. Named entity recognition using an hmm-based chunk tagger. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 473–480, USA. Association for Computational Linguistics.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*.

Álvaro Figueira and Luciana Oliveira. 2017. The current state of fake news: challenges and opportunities. *Procedia Computer Science*, 121:817–825. CENTERIS 2017 - International Conference on ENTERprise Information Systems / ProjMAN 2017 - International Conference on Project MANagement / HCist 2017 - International Conference on Health and Social Care Information Systems and Technologies, CENTERIS/ProjMAN/HCist 2017.

Slavomír Čéplö, Ján Bátora, Adam Benkato, Jiří Milička, Christophe Pereira, and Zemanek Petr. 2016. Mutual intelligibility of spoken maltese, libyan arabic, and tunisian arabic functionally tested: A pilot study. *Folia Linguistica*, 50.