Enriching large language models with semantic lexicons and analogies

Georgios Zervakis

UNIVERSITÉ DE LORRAINE

Thèse de Doctorat

Georgios Zervakis

École doctorale : IAEM

Unité de recherche : Laboratoire Lorrain de Recherche en Informatique et ses Applications UMR 7503

Thèse N°:

Présentée et soutenue publiquement le 8 mars 2023 pour l'obtention du
Doctorat de l'Université de Lorraine
Mention Informatique

Enriching large language models with semantic lexicons and analogies

Enrichir des modèles de langue de grande taille avec des lexiques sémantiques et des analogies

Composition du jury

Rapporteurs :	**Salvatore Ruggieri**, Professeur, Università di Pisa
	Christian Müller, Principal Researcher, DFKI
Présidente du jury :	**Elisa Fromont**, Professeur, Université Rennes 1
Directeurs de thèse :	**Miguel Couceiro**, Professeur, Université de Lorraine
	Emmanuel Vincent, Directeur de Recherche, Inria Nancy – Grand Est
Co-encadrant :	**Marc Schoenauer**, Directeur de Recherche, Inria Saclay – Île-de-France

Résumé

Les progrès récents de l'apprentissage profond et des réseaux de neurones ont permis d'aborder des tâches complexes de traitement du langage naturel, qui sont appliquées à une pléthore de problèmes réels allant des assistants intelligents dans les appareils mobiles à la prédiction du cancer. Néanmoins, les systèmes modernes basés sur ces approches présentent plusieurs limitations qui peuvent compromettre leurs performances et leur fiabilité, les rendre injustes envers les minorités ou exposer des données personnelles. Nous sommes convaincus que l'intégration de connaissances et de raisonnement symboliques dans le cadre de l'apprentissage profond est une étape nécessaire vers la résolution de ces limitations. Par exemple, les ressources lexicales peuvent enrichir les réseaux de neurones profonds avec des connaissances sémantiques ou syntaxiques, et les règles logiques peuvent fournir des mécanismes d'apprentissage et de raisonnement. Par conséquent, l'objectif de cette thèse est de développer et d'évaluer des moyens d'intégrer différents types de connaissances et de raisonnement symboliques dans un modèle de langage largement utilisé, le Bidirectional Encoder Representations from Transformers (BERT).

Dans un premier temps, nous considérons le *retrofitting*, une technique simple et populaire pour raffiner les plongements lexicaux de mots grâce à des relations provenant d'un lexique sémantique. Nous présentons deux méthodes inspirées par cette technique pour incorporer ces connaissances dans des plongements contextuels de BERT. Nous évaluons ces méthodes sur trois jeux de données biomédicales pour l'extraction de relations et un jeu de données de critiques de films pour l'analyse des sentiments, et montrons qu'elles n'ont pas d'impact substantiel sur les performances pour ces tâches. En outre, nous effectuons une analyse qualitative afin de mieux comprendre ce résultat négatif.

Dans un second temps, nous intégrons le raisonnement analogique à BERT afin d'améliorer ses performances sur la tâche de vérification du sens d'un mot, et de le rendre plus robuste. Pour cela, nous reformulons la vérification du sens d'un mot comme une tâche de détection d'analogie. Nous présentons un modèle hybride qui combine BERT pour encoder les données d'entrée en quadruplets et un classifieur neuronal convolutif pour décider s'ils constituent des analogies valides. Nous testons notre système sur un jeu de données de référence et montrons qu'il peut surpasser les approches existantes. Notre étude empirique montre l'importance de l'encodage d'entrée pour BERT, et comment cette dépendance est atténuée en intégrant les propriétés axiomatiques des analogies lors de l'apprentissage, tout en préservant les performances et en améliorant la robustesse.

Abstract

Recent advances in deep learning and neural networks have made it possible to address complex natural language processing tasks, which find application in a plethora of real-world problems ranging from smart assistants in mobile devices to the prediction of cancer. Nonetheless, modern systems based on these frameworks exhibit various limitations that may compromise their performance and trustworthiness, render them unfair towards minorities, or subject them to privacy leakage. It is our belief that integrating symbolic knowledge and reasoning into the deep learning framework is a necessary step towards addressing the aforementioned limitations. For example, lexical resources can enrich deep neural networks with semantic or syntactic knowledge, and logical rules can provide learning and reasoning mechanisms. Therefore, the scope of this thesis is to develop and evaluate ways of integrating different types of symbolic knowledge and reasoning into a widely used language model, Bidirectional Encoder Representations from Transformers (BERT).

In a first stage, we consider retrofitting, a simple and popular technique for refining distributional word embeddings based on relations coming from a semantic lexicon. Inspired by this technique, we present two methods for incorporating this knowledge into BERT contextualized embeddings. We evaluate these methods on three biomedical datasets for relation extraction and one movie review dataset for sentiment analysis, and show that they do not substantially impact the performance for these tasks. Furthermore, we conduct a qualitative analysis to provide further insights on this negative result.

In a second stage, we integrate analogical reasoning with BERT as a means to improve its performance on the target sense verification task, and make it more robust. To do so, we reformulate target sense verification as an analogy detection task. We present a hybrid model that combines BERT to encode the input data into quadruples and a convolutional neural classifier to decide whether they constitute valid analogies. We test our system on a benchmark dataset, and show that it can outperform existing approaches. Our empirical study shows the importance of the input encoding for BERT, and how this dependence gets alleviated by integrating the axiomatic properties of analogies during training, while preserving performance and improving robustness.

Acknowledgements

This work is dedicated to my family for giving me the opportunity to fulfill my goals and the courage to overcome any difficulties in the process. To my father for making sure I had everything I needed, to my brother for giving me the most valuable advice, and to my dearest mother for teaching me to never give up.

In the following, I would like to thank all individuals without whom I would not have reached this outcome in my life. My deepest appreciation to my math teacher Konstantinos Perakis, whose passion and dedication made me love the subject and pursue a degree in the field. Many thanks to Konstantinos Fergadakis, whose pedagogical approach shaped my way of learning Math during the university. I am also grateful to Stylianos Kiagias, for his unconditional support and valuable advice which boosted my motivation and helped me grow as a person. Thanks should also be given to Alexandros Ferles, whose assistance greatly accelerated my transition from theoretical to applied sciences. Words cannot express my gratitude to Guilherme Alves Da Silva, for the fruitful discussions and feedback as a research colleague, but also his willingness and dedication as a person that greatly smoothed my integration to France, especially during the hard times of the pandemic. I am truly grateful to Hee-Soo Choi, whose advice and support greatly boosted my motivation and productivity, particularly towards the end of the thesis. I am deeply indebted to my doctoral supervisors, Miguel Couceiro, Emmanuel Vincent and Marc Schoenauer, whose diverse backgrounds, expertise and guidance made this outcome possible. It has been a great honour being under their supervision, and I am sincerely grateful for all the valuable skills and experiences that I have acquired from our collaboration. I would like to extend my sincere thanks to Elisa Fromont for being the head of the jury, as well as Salvatore Ruggieri and Christian Müller for reviewing my thesis and providing valuable feedback.

I would equally like to give thanks to my professors, colleagues and friends, namely, Amedeo Napoli, Bobby Lee Townsend Sturm JR, Pawel Herman, George Costakis, Mihalis Kolountzakis, Takis Benos, Giouli Dolapsaki, Laura Zanella, Alexandre Bazin, Nacira Abbas, Laurine Huber, Athénaïs Vaginay, Noémie Gonnier, Esteban Marquer, Diego Amaya-Ramirez, Tatiana Makhalova, Claire Theobald, Hans-Jörg Schurr, Priyansh Trivedi, Gabriel Sauger, Vincent Tourneur, Siyana Pavlova, Chuyuan Li, Valentin Richard, Prerak Srivastava, Vinícius Ribeiro, Nicolas Furnon, Paul Magron, Efstahia Vlassopoulou, Iliana Koutani, Sotirios Lekkas, Panteleimon Myriokefalitakis, Christos Matsoukas, Antonios Katsarakis, Nikolaos Xenakis, Harra Ailamaki, Iosif Charkiolakis, Antonios Tsifetakis, and the rest, for their great advice, collaboration and support throughout these years.

Experiments presented in this research were partially supported by the European Union's Horizon 2020 Research and Innovation Program under Grant Agreement No. 952215 TAILOR and by the Inria Project Lab HyAIAI. Experiments presented in this thesis were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

Contents

Abbreviations

AB4TSV	*Analogy and BERT for target sense verification*
AI	artificial intelligence
ANN	artificial neural network
BERT	Bidirectional Encoder Representations from Transformers
BiLSTM	Bidirectional Long Short-Term Memory
CNN	convolutional neural network
DNN	deep neural network
FOL	first-order logic
GPT	Generative Pre-trained Transformer
KG	knowledge graph
LSTM	Long Short-Term Memory
NLP	natural language processing
LLM	large language model
TSV	target sense verification

1. Introduction

Automated processes that rely on artificial intelligence (AI) have gained an increased popularity over the past years, due to their ability to solve complex tasks and advance the state of the art in various fields. The most prominent ones among these methods rely on deep learning and neural networks, which greatly benefit from today's access to large amounts of labeled data and computational resources.

The main challenge in natural language processing (NLP) is to create models that are able to process and accurately represent natural language, whereas natural understanding or natural language interpretation focus on the understanding of the text, e.g., syntax, semantics, sentiment, intent, etc. Recent advancements on the field made it possible to facilitate real-world problems by automating intermediate processes, which previously required domain expertise, human labor and vast amount of time. Consider the example of hiring and recruiting personnel for a company with thousands of applicants for a single position. Manually assessing each and every candidate can be very time-consuming. Instead, an AI can make a first pass and filter those that are relevant for the position, or even recommend others for a different role. This is done by performing *information extraction* and *named-entity recognition*, which allow to automatically extract useful information from the application form such as names, locations, skills, education, etc. Another example is that of language translators. Earlier approaches used to rely on dictionaries or rule-bases systems to translate a piece of text from one language to another. Nowadays, a neural network can be specifically trained to perform *machine translation*, using pairs of texts corresponding to the source and the target language as input. Furthermore, a plethora of applications such as smart assistants, dialogue systems and chatbots, consist of a complex of units that perform several NLP tasks in a sequence. For example, *automatic speech recognition* can be used to recognize and convert the user's request into text. Then, a subsequent component is responsible for parsing the generated text, e.g., to perform *part-of-speech tagging* in order to identify the function of each particular word or phrase in the text (verb, noun, etc.). After the necessary preprocessing steps, a *natural language generation* model is used to output an answer depending on the input request, which is usually followed by a *text-to-speech* engine for translating it back to an utterance. Last but not least, being able to answer a question posed by a user is a really common and challenging application of NLP. This type of problem is frequently encountered in search engines, customer service automation, phone conversational interfaces and more. In such cases, a common approach is to train a dedicated neural network for *question-answering* using data from the domain of interest, e.g., phone call recordings between clients and automated banking service bots. All of the aforementioned tasks benefit from the upgrowth of deep learning but how exactly are they addressed by modern NLP systems?

Deep learning makes it easier to address NLP tasks compared to traditional approaches that require a lot of feature engineering. In principle, the training schema is more or less the same between various approaches, and it can be briefly described as follows. First of all, the initial and most important step is the data acquisition. Good quality and large amount of data are highly responsible for the actual performance of the end system. Next, a training algorithm (model) is selected depending on the task at hand and the nature of the data. The training process consists of an optimization problem in which the goal is to optimize an objective function. During training, the parameters of the network are adjusted as the output of the model gets compared with a ground truth value. Once training is completed, the model's parameters can be directly applied on new data. In the previous example of hiring and recruiting process, we can imagine that the training data consist of thousands of application forms from various job positions, along with entity-level annotations of important features such as location, organization, etc. An end-to-end neural network for named-entity recognition is then trained to learn vectorial representations of the entities of interest, also referred to as *embeddings*, by trying to predict the correct label out of a set of entity types. Similarly for machine translation, the data consist of pairs of sentences corresponding to the source and the target language. Typically, the models starts by encoding the source sentence and produces embeddings for each token in that sentence. Then, the model tries to generate the target sentence word by word, based on the generated tokens so far and the embeddings from the source sentence. In the same manner, natural language generation systems are usually trained to predict the next word based on previously seen words. To this end, the training data are split into pairs of input-target words, e.g., "my name is" becomes "my name"-"name is", where the goal is to generate the target based on the input. At generation time, the model outputs a probability distribution over a vocabulary of words, and selects the one with the highest probability based on the processed words so far. Finally, for question-answering the data consist of paragraphs of texts, different questions related to these texts, along with the respective answers which always lie inside the paragraphs. During training, the model identifies the passage corresponding to the answer of a question, by trying to predict the start and end tokens of that answer.

However, despite their success these approaches face some serious limitations. First of all, they operate as a black-box meaning that it is unclear how the system arrives at a certain conclusion for the task at hand, and why it follows a specific pattern of decision-making. According to Ras et al. (2022), any piece of information that assists in the understanding of these concepts and is conveyable to others is considered an explanation. There exist several methods that enhance the *explainability* of deep models by providing evidence that justify or support their behaviour (Simonyan et al., 2014; Ribeiro et al., 2016; Lundberg and Lee, 2017; Shrikumar et al., 2017). This is different from *interpretability* that points to the intrinsic properties of the model measuring the extent to which the prediction is easily understood by humans (Li et al., 2022). These qualities are crucial for establishing trust between humans and AI, since in many applications the risk is too high, e.g., medical domain, law enforcement and autonomous cars. Yet another issue is related to the *fairness* of deep neural networks. There have been

many reports where automated systems are biased against specific social groups, e.g., they discriminate women during the hiring process for a working position, or associate black people with higher future criminal rates compared to white. The main cause of this behaviour is hidden in the collection of data used to train a particular algorithm. Often the data are imperfect in the sense that they might contain erroneous measurements, missing and unbalanced samples that are not representative of the intended target population, or even sensitive features that should not be taken into account at inference time, e.g., ethnicity, race, age, etc (Pessach and Shmueli, 2022). Moreover, a lot of attention is centered on how deep learning models deal with *security* and *privacy* issues. Many organizations utilize sensitive data from users, e.g., images, voice recordings, conversational texts, in order to improve their AI systems. However, is it shown that these models can be breached through various attacks that aim to steal the parameters of the network, infer private data from users, compromise the performance or manipulate the model's decisions (Liu et al., 2020b). For instance, adversarial examples is a typical situation where small perturbations in the data can lead to wrong predictions with high confidence. This phenomenon is also related to the model's *robustness* which is measured as the expected performance on unseen data which are typically generated by synthetically perturbing the input, or drawn from a different distribution (Wang et al., 2022). In addition, the evolution of deep learning requires more and more computational resources, making it impractical for practitioners to develop their own models. Finally, their performance is poor in the absence of sufficient training data and often overspecialized to particular target groups present in the data. This is an issue considering that the collection of good quality data is both costly and time-consuming, and comes in contrast to how human beings reason and learn — by treating concepts in relation with each other instead of separate units — which allows to grasp new knowledge with just a few examples, and little supervision.

1.1. Context

A step towards addressing the aforementioned limitations, is to exploit existing knowledge by integrating it in the deep learning framework. Following the categorization by Deng et al. (2020), knowledge is divided in two broad classes, *general knowledge* and *domain knowledge*. The former includes the fields of computer science, statistics, neuroscience, physics and others that founded or motivated the development of many known concepts and algorithms in deep learning such as neural networks, backpropagation and dropout. The latter is bounded to specific fields, e.g., biology, sociology, linguistics, is usually gathered by experts in those fields, and is often accompanied with domain-specific applications. Moreover, domain knowledge comes in various forms depending on the level of structure it possesses: from plain language to specialized formats. For the remainder of the thesis we focus on domain-symbolic knowledge, as it is more applicable for integration with deep learning models compared to general knowledge, due to the formal representation (symbols) of its contents. For example using the formalism from propositional logic we can model binary relationships between objects in the form of logic rules.

For instance, if we know that the relation between objects A and B is either equivalence, implication, conjunction, or disjunction we can denote it as $A \rightarrow B$, $A \longleftrightarrow B$, $A \vee B$ or $A \wedge B$, respectively. More complex relations can be constructed from simpler ones using the parenthesis association, or even further extended by adding quantified variables, such as $\forall x \exists y(\textbf{house}(x) \rightarrow \textbf{owns}(y, x))$ which translates to "*every house has an owner*". The latter is considered a first-order logic (FOL) formula decomposition where x, y are variables and **house**(x), **owns**(x,y) are predicates denoting the relations "*x has a house*" and "*y owns house x*", respectively. In a similar way, knowledge can be represented in the form of mathematical expressions such as equations, inequalities and so on. Newton's second law of motion ($F = m \cdot a$) is an example of knowledge that is constant. Typically, this kind of equations can be incorporated in the training objective of a deep learning model, as some sort of constraint, e.g., a regularization term that enforces the model to exhibit or avoid a particular behaviour. Another category is that of probabilistic knowledge. This is closely related to Bayesian theory of probabilities, where Bayes' rule models our beliefs regarding certain events based on priors and new knowledge (also referred to as evidence). Knowledge here takes the form of probabilities from observed data over a population, which can better help to estimate target distributions, infer probabilities for events or even impose constraints. Finally, the most popular structure for domain knowledge in deep learning is represented as a graph whose nodes encode concepts of interest, and its edges denote their attributes and relations. In its most general form, a knowledge graph (KG) constitutes a set of triples that follow a "subject-relation-object" pattern. This is different from ontologies that aim to describe what we know of a certain domain, by providing definitions, rules, relationships for concepts without getting too specific, e.g., generally define the concept of "dog" without focusing on the different breeds. In that regard, knowledge graphs can be viewed as instantiated ontologies.

1.2. Research questions & contributions

We believe that the integration of domain knowledge and reasoning with deep learning models is a good step towards addressing the various limitations that modern AI systems exhibit. For example, in the context of image recognition, the fact that "cats have whiskers and fur" and "sea lions have whiskers but no fur" should help the system recognize whiskers even if the training images are only labeled in terms of "cat" and "sea lion". Then, training a "whisker" classifier on images of various animals is expected to increase the recognition performance for these two, and also enable the discovery of other animals that are neither a cat nor a sea lion. To this end, this thesis project aims to develop and evaluate ways of integrating different types of symbolic knowledge and reasoning into BERT based large language models (LLMs) (Devlin et al., 2019), in an attempt to address the following questions:

- How does one reformat domain knowledge to be compatible with BERT based LLMs?

- How to effectively incorporate domain knowledge and reasoning into BERT based

LLMs?

- Is the use of domain knowledge and reasoning beneficial in terms of performance for a particular task?

- Could domain knowledge and reasoning help increase the robustness of BERT based LLMs?

A summary of the contributions is listed bellow.

1.2.1. Refining BERT embeddings using semantic lexicons

Retrofitting is a simple technique for modifying distributional word embeddings based on relations coming from a semantic lexicon. Building upon the work of Faruqui et al. (2015), we extend retrofitting to operate with contextualized-based systems such as BERT. We propose two methods for refining BERT embeddings that incorporate knowledge coming from both general and domain specific semantic lexicons. We evaluate these methods on three biomedical datasets for relation extraction, and one movie review dataset for sentiment analysis. Our qualitative analysis shows that although such source of knowledge contains too much noise, restricting the lexicons to relevant neighbouring words could help boost performance.

1.2.2. Integrating analogical reasoning into a BERT based architecture

In this work, we focus on the problem of target sense verification (TSV) (Breit et al., 2021). Given a target word in a context along with a definition and a set of hypernyms for that word, we wish to determine whether their senses match or not. In order to solve this task, we translate TSV into analogy detection, and propose a hybrid architecture based on BERT and a specific convolutional neural network (CNN) previously used for solving semantic and morphological analogies. After optimizing the input encodings of BERT, and promoting analogical reasoning by using the axiomatic properties of analogical proportions explicitly during training, we achieve state-of-the-art results on the WiC-TSV evaluation benchmark Breit et al. (2021) along with a more robust model.

1.3. Thesis overview

The remainder of the thesis is organized as follows.

Chapter 2 starts with some background information with respect to NLP tasks, knowledge bases and neural networks. Then, it summarizes various approaches that integrate symbolic knowledge and reasoning with deep learning frameworks.

Chapter 3 introduces an extension of the retrofitting algorithm for injecting symbolic knowledge into BERT embeddings. An extensive evaluation of the proposed methods

along with a qualitative analysis of the results are provided.

Chapter 4 presents *Analogy and BERT for target sense verification* (AB4TSV), a framework for solving target sense verification through analogies. The proposed method takes advantage of the axiomatic theory of analogical proportions to integrate analogical reasoning into the model, which in turn maintains its performance and increases its robustness.

Chapter 5 concludes with the main outcomes of the thesis and points out future directions for research.

2. State of the art

The scope of this chapter is to familiarize the reader with the topic of integration of symbolic knowledge and reasoning in neural networks. In Section 2.1 we explain how deep learning is employed in modern NLP with a primary focus on large language models (LLMs) and the NLP tasks that will be tackled in the following chapters. In Section 2.2 we review on some external knowledge sources that are publicly available. Finally, in Section 2.3 we briefly survey various approaches that combine symbolic knowledge and reasoning with deep learning methods.

2.1. Deep learning based NLP

NLP has drastically evolved through the years — from the development of rule-based algorithms for grammar checking, to the deployment of applications that facilitate everyday life tasks such as search engines and automatic translation. Taking advantage of the vast amount of data that are currently available, the learning process now aims to learn optimal features from the data and correlate them with the desired output.

2.1.1. Deep Learning

Inspired from biology, the first artificial neural networks (ANNs) were designed to simulate the perceptual process of human brain cells (McCulloch and Pitts, 1943; Rosenblatt, 1958). Essentially, ANNs can be viewed as an approach to automatically extract useful features from the data, i.e., create a mapping from the inputs to the outputs directly from data. This is accomplished by learning simple, non-linear transformations of the inputs by means of parametric functions called neurons, which are gradually combined within layers in the network, to shape more complex representations at different levels of abstraction (LeCun et al., 2015; Goodfellow et al., 2016). Deep learning has emerged from increasing the number of layers in ANNs. The resulting deep neural networks (DNNs) capture more sophisticated relations in the data, which are not explicitly designed by humans. In the most general case, i.e., supervised learning, training a DNN requires a loss function that measures, in terms of a performance metric, how far the outputs of the model are from the desired output. The goal is to find the optimal set of parameters that minimizes the expected loss on the training data. Backpropagation solves this problem by computing the gradient of the loss with respect to the learnable parameters of the DNN, and using the chain rule to update them accordingly following existing optimization algorithms, such as stochastic gradient descent (Kiefer and Wolfowitz, 1952).

2.1.2. Word embeddings

Prior to deep learning, statistical approaches to NLP were based on other statistical machine learning methods that heavily depend on hand-crafted features (Cortes and Vapnik, 1995; Zhang, 2004). A typical example is the encoding of a text document as a bag-of-words, where the representation is constructed based on the frequency of unique words in a given corpus. Similarly, term frequency-inverse document frequency is another popular technique which measures how important a word is to a document within a corpus. However, such features are often high-dimensional and sparse, and result in a series of problems that is referred to as *curse of dimensionality*, making the learning process inefficient in practice. To deal with the curse of dimensionality Bengio et al. (2003) proposed to learn distributional word representations in low-dimensional spaces. Following the distributional hypothesis — words that appear in the same context tend to be semantically similar — every word in the vocabulary is now viewed as a point or vector in a high-dimensional space, called *word embedding*. This representation is able to capture various linguistic properties from text (syntax, word similarity/relatedness, analogy), according to the training objective that is being optimized by a given algorithm. The prominence of word embeddings was well recognized when word2vec was introduced by Mikolov et al. (2013). It was shown that word2vec outperformed previous methods on a variety of language tasks like semantic relatedness, synonym detection, clustering and analogy completion (Baroni et al., 2014). Most importantly, a learnt set of word embeddings can be transferred to other domains such as knowledge discovery in scientific literature (Tshitoyan et al., 2019) and recommendation systems (Grbovic and Cheng, 2018). Following works (Pennington et al., 2014; Bojanowski et al., 2017) aimed at improving known limitations of word2vec, such as handling out-of-vocabulary words or dealing with word morphology. However, the biggest disadvantage of distributional representations is their inability to model word polysemy. For example the word *mouse* will always have a single embedding regardless of its meaning (animal or computer hardware). Various methods that take into account context have been proposed to address this issue, but in reality, contextualized representations already implicitly existed in recurrent neural networks, such as in the famous Long Short-Term Memory (LSTM) architecture (Hochreiter and Schmidhuber, 1997). One of the first dedicated approaches to generate context based embeddings was TagLM (Peters et al., 2017), that built on the idea of concatenating conventional word embeddings with context-sensitive representations from a neural language model trained on a large amount of unlabeled data. A year later Peters et al. (2018) introduced ELMo, a successor of TagLM that uses the contextual representations of all layers of the model, instead of the top layers as in TagLM. The intuition is that different types of information about a word may be encoded at different layers of the model, hence some could be more useful than others depending on the task. ULMFiT (Howard and Ruder, 2018) is another model that promoted the concept of transfer learning, i.e., training a large model on a lot of a data, and then applying it for particular NLP tasks. ULMFiT reuses the same network architecture and simply trains the model by changing the top layer according to the underlying task. Nowadays, there exist a plethora of approaches that effectively deal with contextuality, namely, BERT

(Devlin et al., 2019), Generative Pre-trained Transformer (GPT) (Radford et al., 2019), XLNET (Yang et al., 2019), T5 (Raffel et al., 2020) and more. These are based on the Transformer architecture (Vaswani et al., 2017) that greatly advanced the state of the art in NLP. Below we briefly explain this concept by focusing on a particular Transformer model that will be extensively used throughout this thesis.

2.1.3. Bidirectional Encoder Representations from Transformers

Released by Devlin et al. (2019), BERT was a breakthrough in NLP since it greatly advanced the state of the art in the field. Unlike its ancestors that processed the input text in a sequential manner – either left-to-right or right-to-left – BERT is bidirectional, allowing the token representations to depend on the full surrounding context. Moreover, it can serve as a basis to solve many NLP tasks with minimal architecture modifications.

2.1.3.1. Input formatting

BERT utilizes the WordPiece tokenizer (Wu et al., 2016) to convert the input text into tokens, using a vocabulary $\mathcal{V}$ that consist of $\sim$ 30K concrete words, subwords and individual characters. Consider the example "*BERT produces contextualized representations.*". The tokenizer begins by checking whether each word is part of the vocabulary or not. If not, it tries to decompose the word into the largest possible subword in the vocabulary, and if that is not sufficient, it ends up splitting the word into individual characters. This way the model can generate representations for out-of-vocabulary words. In this example, all words apart from "*contextualized*" belong to $\mathcal{V}$, thus the tokenization is the following: ['bert', 'produces', 'context', '##ual', '##ized', 'representations']. All words are lower-cased, punctuations are removed and "*contextualized*" is decomposed into "*context*", "*##ual*" and "*##ized*". If a subword occurs at the front of the original word it remains unchanged, whereas if it appears later like in this example, the "##" symbol is used to differentiate. Depending on the downstream task, BERT can either process a single sentence or a pair of sentences that it respectively encodes in the following format:

$$S = \texttt{[CLS]}, t_1, \ldots, t_M, \texttt{[SEP]}$$

or

$$S = \texttt{[CLS]}, t_1, \ldots, t_N, \texttt{[SEP]}, t_{N+1}, \ldots, t_M, \texttt{[SEP]}$$

where S is a sequence of M discrete tokens. The classification token (`[CLS]`) and the separation token (`[SEP]`) are added at the start and end of each sentence respectively. The former is useful for classification tasks, while the latter helps the model to differentiate between the two sentences in the input pair. Next, the input sequence length is fixed to a constant value, thus S is padded or truncated to meet this requirement. In the case of padding, a special `[PAD]` token is repeatedly appended to the end of the input sequence until the specified length is met. For truncation, tokens are sequentially removed either from the end of the first, the second or the longest sentence. Once the input is fully tokenized, it is converted from a list of tokens into a list of indices (input

ids), each mapped to the corresponding token in the vocabulary. Next, a list of zeroes and ones (segment ids) is created to indicate whether an input token belongs to the first (id = 0) or the second sentence (id = 1). Additionally, a list of integers with values from 0 to M (position ids) is used to inform the model of the position of each token in S. For each input, segment and position id the model generates the corresponding token, segment and position embedding vector via $\mathbf{W}^{\text{TE}} \in \mathbb{R}^{d_\mathcal{V} \times d}$, $\mathbf{W}^{\text{SE}} \in \mathbb{R}^{2 \times d}$ and $\mathbf{W}^{\text{PE}} \in \mathbb{R}^{M \times d}$ respectively, where $d_\mathcal{V}$ is the vocabulary size and d is the embedding dimensionality. All three types of embeddings are summed element-wise to produce the final input representation $\mathbf{X} \in \mathbb{R}^{M \times d}$. This is depicted in the example of Figure 2.1.

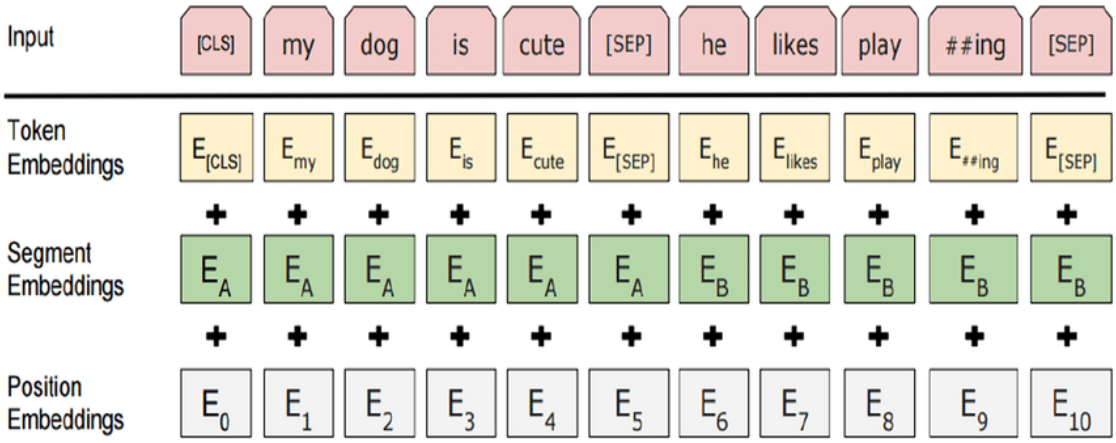


Figure 2.1.: Input representation of BERT as an element-wise sum of the token, segment and position embeddings (Devlin et al., 2019).

2.1.3.2. The Transformer architecture

The original Transformer as proposed by Vaswani et al. (2017) is composed of an encoder and a decoder part as shown in Figure 2.2. The former is useful for creating meaningful representations of text, while the latter makes use of these to perform language generation. BERT only employs the encoder part. This is a stack of L Transformer encoder blocks of the same structure but with different parameters. Each block is broken down into two sub-layers. It starts off with a *self-attention* layer that computes a score for each token in the input sequence S, against all tokens in S. Intuitively, this score reflects the importance that each token assigns to individual tokens in the input, and thus, allows for contextualized token representations. In practice, self-attention is essentially a mapping from a query and a set of key-value pairs to an output. For each token in S the following steps are applied:

1. A query vector $\mathbf{q}_i$ and a key vector $\mathbf{k}_i$ of dimension d_k, and a value vector $\mathbf{u}_i$ of dimension d_v are created by multiplying the embedding $\mathbf{x}_i \in \mathbb{R}^d$ of token i with trainable weight matrices $\mathbf{W}^Q \in \mathbb{R}^{d \times d_k}$, $\mathbf{W}^K \in \mathbb{R}^{d \times d_k}$, $\mathbf{W}^V \in \mathbb{R}^{d \times d_v}$ respectively:

$$\begin{aligned} \mathbf{q}_i &= \mathbf{x}_i \mathbf{W}^Q \\ \mathbf{k}_i &= \mathbf{x}_i \mathbf{W}^K \\ \mathbf{u}_i &= \mathbf{x}_i \mathbf{W}^V. \end{aligned} \tag{2.1}$$

2. Then, the i-th token is scored against all tokens j in S by taking the dot product of the query vector for i and the key vector for j:

$$s_{ij} = \mathbf{q}_i \mathbf{k}_j^T, \ \forall j \in [1, M]. \tag{2.2}$$

3. The obtained score is divided by the square root of d_k for numerical stability during training, and then it is normalized by a softmax operation:

$$s'_{ij} = \frac{s_{ij}}{\sqrt{d_k}} \tag{2.3}$$

$$s''_{ij} = \frac{e^{s'_{ij}}}{\sum_{j=1}^{M} e^{s'_{ij}}}. \tag{2.4}$$

4. Next, the normalized score is multiplied by each value vector of all j-tokens:

$$\mathbf{u}'_{ij} = s''_{ij} \mathbf{u}_j, \ \forall j \in [1, M]. \tag{2.5}$$

5. Finally, the output of self-attention for token i is a sum of weighted value vectors:

$$z_i = \sum_{j=1}^{N} \mathbf{u}'_{ij}. \tag{2.6}$$

At implementation time the self-attention is applied simultaneously on the whole input sequence, by packing all queries, keys and values into corresponding matrices $\mathbf{Q} \in \mathbb{R}^{M \times d_k}$, $\mathbf{K} \in \mathbb{R}^{M \times d_k}$, $\mathbf{V} \in \mathbb{R}^{M \times d_v}$:

$$\text{Attention}(\mathbf{Q},\ \mathbf{K},\ \mathbf{V}) = \text{Softmax}\Big(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\Big)\mathbf{V} = \mathbf{Z}. \tag{2.7}$$

This concept is further expanded to *multi-headed attention* with the queries $\mathbf{Q}$, keys $\mathbf{K}$ and values $\mathbf{V}$ being linearly projected h times using different $\mathbf{W}^Q \in \mathbb{R}^{d \times d_k}, \mathbf{W}^K \in \mathbb{R}^{d \times d_k}, \mathbf{W}^V \in \mathbb{R}^{d \times d_u}$ projection matrices, where $d_k = d_v = d/h$. The resulting heads are concatenated and multiplied by a weight matrix $\mathbf{W}^O \in \mathbb{R}^{d \times d}$ to produce a single matrix $\hat{\mathbf{Z}}$ that contains information from all attention heads:

$$\begin{aligned} \text{MultiHead}(\mathbf{Q},\ \mathbf{K},\ \mathbf{V}) &= \text{Concat}(\mathbf{Z}_1, \ldots, \mathbf{Z}_h)\mathbf{W}^O = \hat{\mathbf{Z}} \\ \text{where } \mathbf{Z}_i &= \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V), \ \forall i \in [1, h]. \end{aligned} \tag{2.8}$$

Effectively, multi-headed attention allows the model to have multiple representation subspaces through the use of the h different $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ matrices, resulting in various attention

patterns associated with each Transformer block and head (Vig, 2019). Examples of these include the delimiter-focused attention pattern, where a lot of attention is centered on the `[SEP]` token since the attention head is unable to associate it with anything else in the input sequence. Next-word attention pattern is another example where each token is focusing on the subsequent one, excluding `[CLS]` and `[SEP]`. The intuition behind this pattern is that adjacent words are often the most relevant for understanding the meaning of a word in context. The second sub-layer in the Transformer block of BERT applies a point-wise, fully connected feed-forward neural network of two linear transformations with a GELU (Hendrycks and Gimpel, 2016) activation function in between them:

$$\begin{aligned} \text{FFNN}(\mathbf{x}) &= \text{GELU}(\mathbf{x}\mathbf{W}_1 + \mathbf{b1})\mathbf{W}_2 + \mathbf{b}_2 \\ \text{where GELU}(x) &= 0.5\,x\Big(1 + \tanh\Big(\sqrt{2/\pi}(x + 0.044715\,x^3)\Big)\Big). \end{aligned} \tag{2.9}$$

The output of this serves as input to the subsequent block, where $\mathbf{W}_1 \in \mathbb{R}^{d\times 4d}, \mathbf{b}_1 \in \mathbb{R}^{4d}$ and $\mathbf{W}_2 \in \mathbb{R}^{4d\times d}, \mathbf{b}_2 \in \mathbb{R}^{d}$ are learnable weights. Furthermore, a residual connection (He et al., 2016) followed by Dropout (Srivastava et al., 2014) and a layer normalization (Ba et al., 2016) is applied after each sub-layer, such that output is:

$$\text{LayerNorm}(\mathbf{x} + \text{Dropout}(\text{Sublayer}(\mathbf{x}))) \tag{2.10}$$

where $\text{Sublayer}(\mathbf{x})$ is the function implemented by the sub-layer itself. An illustration of a Transformer block is shown in Figure 2.2. Throughout this work, we experiment with the $\text{BERT}_{\text{BASE}}$ version of the model, that utilizes 12 Transformer blocks and 12 multi-attention heads. The input sequence length is fixed to 512. We make use of the *uncased* vocabulary of $d_{\mathcal{V}} = 28,996$. Finally the model's hidden dimension is set to $d = 768$.

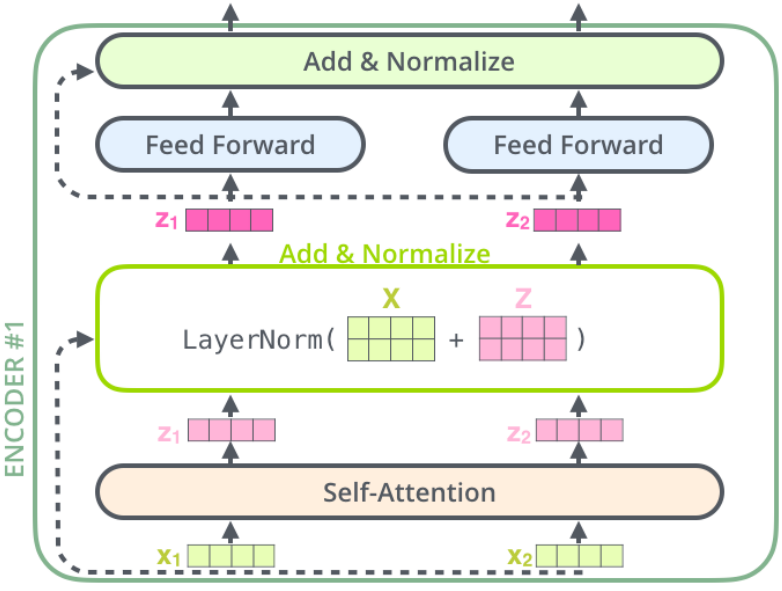


Figure 2.2.: The Transformer encoder block architecture (Alammar, 2018).

2.1.3.3. Pretraining and finetuning

BERT is originally pretrained on documents ($\sim$ 3.3M words) from the BookCorpus (Zhu et al., 2015) and the English Wikipedia. The pretraining objective consists of two unsupervised tasks: masked language modeling and next sentence prediction. Typically, when building a language model the objective is at each step to predict a target word, based on either previously observed words (left-to-right) or words that come after it (right-to-left). In standard conditional language modelling, looking simultaneously on both sides of the context can be problematic, since the model could trivially predict the target word as it has already seen it while predicting a previous word. To deal with this issue, Devlin et al. (2019) randomly replace 15% of the input tokens with a `[MASK]` token, and try to predict those instead. In next sentence prediction, the model takes as input a pair of sentences from the training corpus, and it must decide whether one follows the first one in the original text. Doing so, the model learns how to capture relationships between successive sentences, which is crucial for downstream tasks such as question-answering and natural language inference. The finetuning process allows to BERT to be adapted to a particular downstream task with minimal architecture modifications. More specifically, once the task-specific inputs and the output layer are specified, the entire model is trained end-to-end for a few epochs on labeled data for the downstream task. This is also called finetuning. As previously described, the input consists of either a single sentence or a sentence-pairs, e.g., movie reviews for sentiment analysis or question-answer pairs for question-answering. The output layer is commonly a simple feed-forward layer used to generate the prediction for the downstream task. The inputs to this layer usually depend on the nature of the task. For token-level tasks such as word sense disambiguation and question-answering, the BERT encoder representations of the tokens of interest are fed into the output layer, while for classification tasks like relation extraction or next sentence prediction, the `[CLS]` token representation is used.

2.1.4. Example usages of BERT

As of late, deep learning models tend to replace classical machine learning systems since they require less feature engineering and generally perform better in many NLP tasks. Below we showcase examples of deep learning based NLP applications/problems, and how they are addressed with BERT, discussing at the same time existing limitations of the model.

2.1.4.1. Biomedical relation extraction

Biomedical relation extraction is the task of identifying relations between pairs of entities in a biomedical corpus. Typically, the data consist of individual sentences with annotated relations of named entities, such as drug-to-protein interactions. The objective is to assign to each sentence a relation type from a predefined set. Performance is measured in terms of precision, recall and F_1-score. Peng et al. (2019) pretrained their own BERT variant on approximately 4B words from PubMed abstracts (Fiorini et al., 2018) and

500M words from clinical notes from MIMIC-III database (Johnson et al., 2016). They treated biomedical relation extraction as a sentence classification task, where they masked the named entities with custom tags, e.g., *"Proto-oncogene PIM-1 is a novel estrogen receptor target associated with high grade breast tumors."* becomes *"@CHEMICAL$ PIM-1 is a novel @GENE$ receptor target associated with high grade breast tumors."*. Then, they finetuned their model for sentence classification using the `[CLS]` token embedding of the last encoder layer as input to a feed-forward layer, that produces an estimated posterior probability for each relation type. The relation type with the highest posterior probability is selected as the final estimate. Similarly, Lee et al. (2020) released BioBERT, BERT model pretrained on 4.5B words from PubMed abstracts and 13.5B words of PubMed full-text articles. They used the same finetuning procedure as Peng et al. (2019), however they experimented with different masking strategies for the named entities. SciBERT (Beltagy et al., 2019) is yet another BERT model pretrained on 1.14M biomedical and computer science papers from Semantic Scholar (Ammar et al., 2018). Unlike the previous variants that are pretrained using the standard BERT vocabulary, SciBERT comes with its own WordPiece vocabulary. The authors experimented with standard finetuning and frozen BERT embeddings. For the former, they used the final embedding of the `[CLS]`-token as input to a linear layer. For the latter, they used BERT encoder embeddings of all tokens in the sentence as input to a 2-layer Bidirectional Long Short-Term Memory (BiLSTM) (Schuster and Paliwal, 1997), and then they fed the concatenated first and last BiLSTM vectors to a feed-forward layer. Although these BERT variants achieve better results than the previous state-of-the-art models, there still suffer from several limitations. Direct comparison of these models is not always possible, as for example SciBERT uses the gold-standard annotations of entities for relation extraction, whereas BioBERT performs named entity recognition and relation extraction jointly (hence the use of masking for the named entities). Furthermore, Peng et al. (2019) report that their base version of BlueBERT is less efficient than the large one (12 vs. 24 Transformer encoder blocks) potentially due to the bigger average sequence length in relation extraction tasks. Handling long sequences is a known problem of Transformers related to the attention mechanism (Zaheer et al., 2020). This comes in agreement with the findings of Alimova et al. (2021), showing that BioBERT has a limited capacity to model relations between pairs of entities in a given text span, with an average performance drop of 34.2% in F_1-score in cross-domain evaluation.

2.1.4.2. Sentiment analysis of movie reviews

Sentiment analysis is the task of deciding whether the intended sentiment of a piece of text is positive or negative. Given a collection of movie reviews in the form of sentences along with human annotations of their sentiment, Devlin et al. (2019) treats the problem as a binary sentence classification task. Taking for example the review *"Rarely has so much money delivered so little entertainment."* BERT is finetuned to predict either 0 for the negative class or 1 for the positive. This is done by feeding the `[CLS]` token representation of the last encoder layer of BERT in a feed-forward layer classifier to output a posterior probability of the review being positive or negative. The class with the highest score

is selected. The overall performance is measured in terms of classification accuracy. Jiang et al. (2020a) argue that standard finetuning tends to harm the generalization ability of large pretrained models such as BERT, due to overfitting on the limited data resources for the downstream task. Instead, they propose a modified finetuning method that accounts for the model complexity via regularization, and Bregman proximal point optimization to avoid aggressive updating of the weights at each epoch. Their BERT based model shows improvements on a series of tasks from the GLUE benchmark (Wang et al., 2018), including sentiment analysis. Aghajanyan et al. (2021) introduce a multi-task, pre-finetuning approach that combines gradient optimization of multiple tasks, loss scaling and task sampling schemes, and leads to representations that generalize better across different tasks. They tested their method with RoBERTa (Liu et al., 2019b) showing improvements in sentiment analysis among other tasks.

2.1.4.3. Target sense verification

TSV is a word sense disambiguation task in which the system is provided with a target word in context on the one hand, and a definition and a set of hypernyms of that word on the other hand. The system must decide whether their senses match or not. Consider for instance the context *"home is where the heart is"*, the definition *"where you live at a particular time"* and the set of hypernyms *"residence, abode"*, all corresponding to the target word "home". To disambiguate the meaning of *"home"* in that specific context, one must compare and reason about the underlying relations between these concepts, e.g., infer that *"home"* in this context refers to an environment rather than a place as conveyed by the definition and the hypernyms. Performance is measured in terms of precision, recall and F_1-score. The standard approach of solving TSV with BERT is to patch the inputs into a pair of sentences (context, definition; hypernyms) (Breit et al., 2021). Then, either the `[CLS]` token representation of the last encoder layer of BERT, or a concatenation of the representations of the `[CLS]` token, the target word and the average of all words in the definition and all hypernyms, can be fed as input to a feed-forward network classifier. If the output score is a above a fixed threshold (typically 0.5), the senses are estimated to match, otherwise they are estimated to be distinct. Unlike classical neural networks that use their own training schema, BERT-like architectures are pretrained on generic tasks making it difficult to inject additional features as re-training of the full model is highly expensive. To deal with this effect, Moreno et al. (2021) introduced start and end marker tokens to highlight the target word in context e.g., *"home is where the heart is"* becomes *"[E1] home [/E1] is where the heart is"*, a strategy that helps the model focus on important parts of the input. They finetuned two distinct BERT models on the hypernyms and the definition, using the `[CLS]` token embedding for classification, and they aggregated the two classifier outputs at inference time. Vandenbussche et al. (2021) ran an extensive study of BERT for TSV, including data augmentation, freezing the model parameters during finetuning, applying different pooling strategies to obtain the classifier input, or masking the target word in the context. In contrast to Moreno et al. (2021) that emphasized the target word in context with markers, they set the segment ids of the target word to match those of the

definition showing a slight improvement in accuracy. Additionally, their error analysis shows that BERT often struggles to disambiguate the senses when the context provided for the target word is limited. Liu et al. (2021) argued that off-the-shelf contextualized representations of LLMs are usually dominated by the ones obtained from the finetuned version of the same model. In order to improve on this aspect, they proposed a more generic approach called MIRROWIC, and tested it on various lexical semantic tasks including TSV. This fully unsupervised approach based on contrastive learning aims to extract improved word embeddings from LLMs such as BERT. Specifically, they created positive and negative pairs for a target word by making use of augmentation, masking and dropout techniques, as well as raw samples from Wikipedia. Then, they trained the embeddings such that positive pairs are pulled closer, while negative pairs are pushed apart. To evaluate their method on TSV, they constructed manual templates involving the target word, the definition and/or the hypernyms, and compared the cosine similarities of the target word embeddings in the original context and the template.

2.1.4.4. Fact completion

Fact completion is the task of testing the ability of language models to fill-in missing information about facts of the world. Typically, facts are represented as subject-relation-object triples, e.g., *(Joe Biden, president of, United States)* or question-answer pairs, e.g., *Who is the president of United States? Joe Biden.* At test time, the model has to infer the missing entry of a cloze-type statement concerning a specific fact. In the previous example that could be *"The president of United States is _____"*. Converting a fact into a cloze statement requires a prompt, i.e., a template that transforms the fact into a sentence. There is no standard way of selecting a prompt but the choice significantly impacts the performance of the model. It can be manually created, automatically generated for a specific task (Bouraoui et al., 2020; Jiang et al., 2020b; Gao et al., 2021; Haviv et al., 2021) or optimized for a given task (Shin et al., 2020; Liu et al., 2022). Following the masked language modeling pretraining objective for BERT, Petroni et al. (2019) replaced the relation object in the fact's prompt with the `[MASK]` token. Then, the representation of the last BERT encoder layer corresponding to the `[MASK]` token is fed into a softmax output layer, to produce a probability score over a unified vocabulary. Performance is measured in terms of mean precision at k (P@k), meaning that if the generated token is ranked among the top-k predictions then it is assigned a value of 1, and 0 otherwise. In their work, they only considered single token objects as answers, since multi-token generation greatly complexifies the setting. Generally, compared to language generation models, BERT is limited when it comes to multi-token generation as the exact number of `[MASK]` tokens have to be specified beforehand. Moreover, their findings suggest that although BERT might correctly recall a fact, sometimes the generated answer may be produced by memorization of the information during training rather than factual "understanding".

2.2. Symbolic Knowledge

Most often, the only source of information that is given to a deep learning system is data only tailored for the task at hand. However, this specification can be problematic since the data alone only provide a partial specification of the task to be performed.

2.2.1. Why external knowledge can be useful?

External knowledge sources can enrich the information present in the data as they provide better coverage of the concepts of interest. For example, they can provide complementary information when the training data are sparse, or help improve the expressivity of a language generation model by pointing to semantically equivalent concepts, e.g., paraphrases. Moreover, they can be used for training data augmentation such that the deep learner becomes more robust to input variations. In addition, symbolic knowledge can also help design more interpretable architectures that take into account specific rules to be respected by the model (Camacho-Collados and Pilehvar, 2018). At the same time, being able to impose such constraints to the model is a step towards protecting privacy, e.g., ensuring that the predictions of the model do not contain sensitive information from the training data, and towards eliminating biases that the system reproduces due to errors during the data collection process. Finally, the integration of knowledge graphs (KGs) with deep learning motivates the development of systems that learn concepts in relation with each other rather than treating them independently. Exploiting various relations in the KG could allow the discovery of new concepts, while requiring less training data.

2.2.2. Knowledge graphs

As the availability of data is increasing, so is the necessity of extracting, storing and organizing the important information in the data. However, there is no unified approach for doing so since knowledge can be general or tied to specific domains, and typically comes in different types and formats, e.g., from plain text to specialized symbols. In the following subsections, we review existing symbolic knowledge resources in the form of KGs that are generally considered in deep learning based NLP, with a particular focus on the ones used throughout this work.

2.2.2.1. Formalization of knowledge graphs

Broadly speaking, a KG is a graph whose nodes encode knowledge about concepts or entities, and whose edges represent semantic relationships between them. Concepts typically refer to general instances, while entities point to physical objects in the real world, such as a person or a location. There is no standard formal way to define a KG (Ji et al., 2021). In its most general form, a KG is a directed, labeled graph that comprises a set of "subject-relation-object" triples:

$$\mathcal{G} = \{(s, r, o) | s \in \mathcal{E}, r \in \mathcal{R}, o \in \mathcal{E}\} \tag{2.11}$$

where instances s and o are related by r, with $\mathcal{E}$ being the set of all concepts or entities and $\mathcal{R}$ the set of all relations.

2.2.2.2. Review of existing knowledge graphs

WordNet (Miller, 1995) is among the most popular and widely used public knowledge resources used in NLP. It is a human-constructed lexical database that groups nouns, verbs, adjectives and adverbs based on their meanings (or senses). Synonymous words which refer to the same concept and are interchangeable in many contexts, are clustered together into unordered sets called *synsets*. Each synset comes with its own definition (gloss), and sometimes a few sentence examples that showcase the usage of each synset member. Moreover, polysemous word forms are grouped in as many distinct synsets as their different meanings. For example, the word *bass* has nine senses in WordNet including synsets like {*bass*, *deep*} for the adjective, or {*bass*, *bass voice*, *basso*}, {*bass*, *sea bass*} for the noun. Statistics such as the number of words, synsets, and senses as well as polysemy information are display in Table 2.1. The graph structure of WordNet (Figure 2.3) interlinks synsets with various semantic relations including but not limited to:

- Hypernymy or superordinate: when a synset is a superclass of another synset, e.g., *travel* $\rightarrow$ *fly*.
- Hyponymy or subordinate: when a synset is a subclass of another synset, e.g., *mango* $\rightarrow$ *fruit*.
- Meronymy or has-part: when a synset is a constituent part or member of another synset, e.g., *wheel* $\rightarrow$ *car*.
- Holonymy or part-of: when a synset pertains to another synset, e.g., *chair* $\rightarrow$ *leg*.
- Antonymy: when two synsets have opposite meanings, e.g., *increase* $\longleftrightarrow$ *decrease*.
- Troponymy: when the event characterized by a verb synset is included in that of another verb synset, e.g., *walk* $\rightarrow$ *stroll* or *communicate* $\rightarrow$ *talk* $\rightarrow$ *whisper*.
- Entailment: when the event described by a verb synset entails that of another, e.g., *snore* $\rightarrow$ *sleep*.

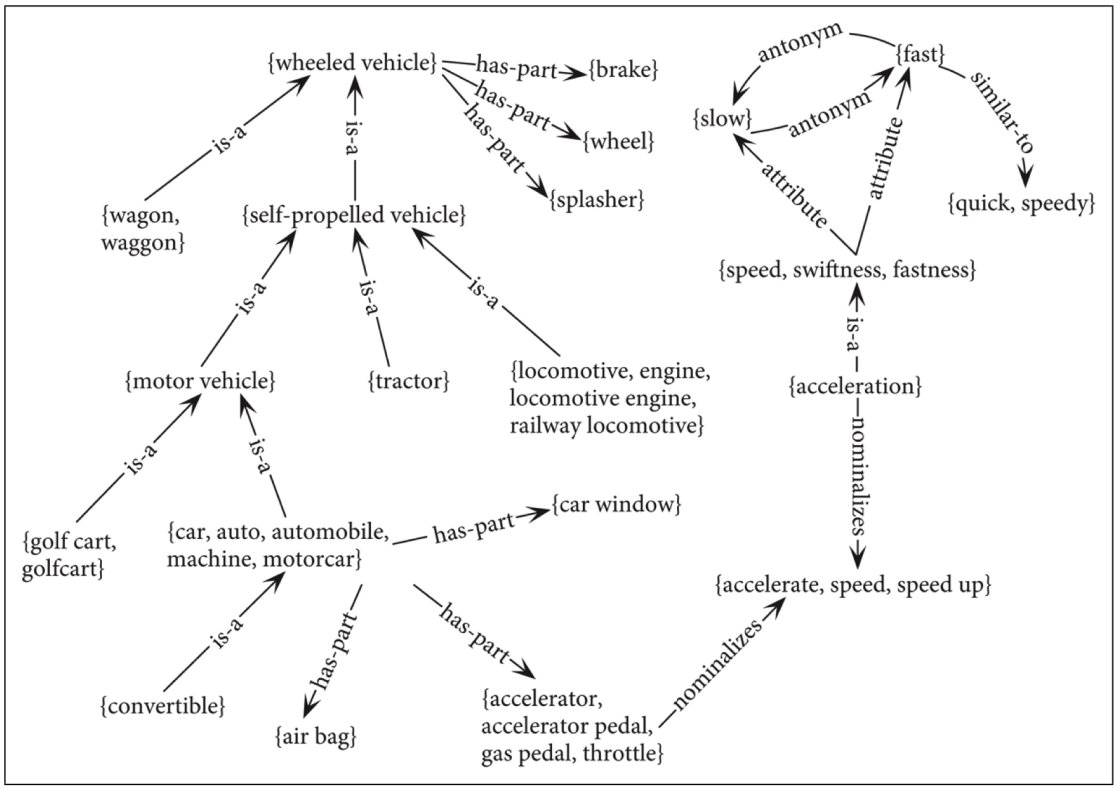


Figure 2.3.: Graph representation snippet of WordNet (Navigli, 2016).

Table 2.1.: Statistics of WordNet 3.0 database.

POS	Unique strings	Synsets	Total word-sense pairs
Noun	117,798	82,115	146,312
Verb	11,529	13,767	25,047
Adjective	21,479	18,156	30,002
Adverb	4,481	3,621	5,580
Total	155,287	117,659	206,941

POS	Monosemous words and senses	Polysemoys words	Polysemoys senses
Noun	101,863	15,935	44,449
Verb	6,277	5,252	18,770
Adjective	16,503	4,976	14,399
Adverb	3,748	733	1,832
Total	128,391	26,896	79,450

POS	Average polysemy incl. monosemous words	Average polysemy excl. monosemous words
Noun	1.24	2.79
Verb	2.17	3.57
Adjective	1.40	2.71
Adverb	1.25	2.50

FrameNet (Baker et al., 1998) is a lexical database that is based on the theory of Frame Semantics (Fillmore and Baker, 2001) — a description of a type of event, entity or relation, also called *semantic frame*, constitutes a good basis for understanding the meaning of most words that occur in it. Consider the example "He was born in 1961". In FrameNet, this sentence is an instance of the [`Being_born`] *frame* along with additional semantic information like "He" marked as [`Child`] and "in 1961" marked as [`Time`], referred to as *frame elements*. Words or lemmas that evoke a particular frame are called *lexical units*, e.g., *born*, *come into the world* for [`Being_born`]. FrameNet consists of over $1,200$ semantic frames, $13,000$ lexical units and $202,000$ example sentences. Statistics are provided in Table 2.2. Figure 2.4 depicts a graph representation of [`Being_born`] along with its relations to existing frames. An explanation of the frame relations is provided below.

- Inheritance or is-a: when the child frame inherits all the features of the parent frame, e.g.,
  [`Revenge`] → [`Rewards_and_punishments`].
- Subframe: when the child frame is part of a complex event represented by the parent frame, and it can be separately described as a frame, e.g.,
  [`Criminal_process`] → {[`Arrest`], [`Arraignment`], [`Trial`], [`Sentencing`], [`Appeal`]}.
- Perspective on: indicates the presence of at least two different possible perspectives on the neutral parent frame, e.g.,
  [`Get_a_job`] and [`Hiring`] → [`Begin_employment`]
  from the perspectives of the employer and employee, respectively.
- Using: when the child frame refers to the parent frame as background, e.g.,
  [`Volubility`] → [`Communication`].
- Causative of and Inchoative of: systematic, non-inheritance relationships between stative frames, e.g.,
  "She had a high salary." — [`Position_on_a_scale`],
  "Her salary increased." — [`Change_position_on_a_scale`]
  and
  "She raised his salary." — [`Cause_change_position_on_a_scale`].
- See Also: relates groups of frames that are similar and should be carefully distinguished, e.g.,
  [`Scrutiny`] → [`Seeking`].
- Precedes: captures a temporal order that holds between child subframes of a complex event represented by the parent frame, e.g.,
  [`Sleep_wake_cycle`] → {[`Being_awake`], [`Fall_asleep`], [`Sleep`], [`Waking_up`], [`Getting_up`], [`Being_awake`]}
  in which all subframes of [`Sleep_wake_cycle`] precede each other in a cycle.
- Metaphor: relation between a source and a target frame where lexical units in the target frame are at least partially explained by the source frame, e.g.,
  [`Hostile_encounter`] → [`Firefighting`].

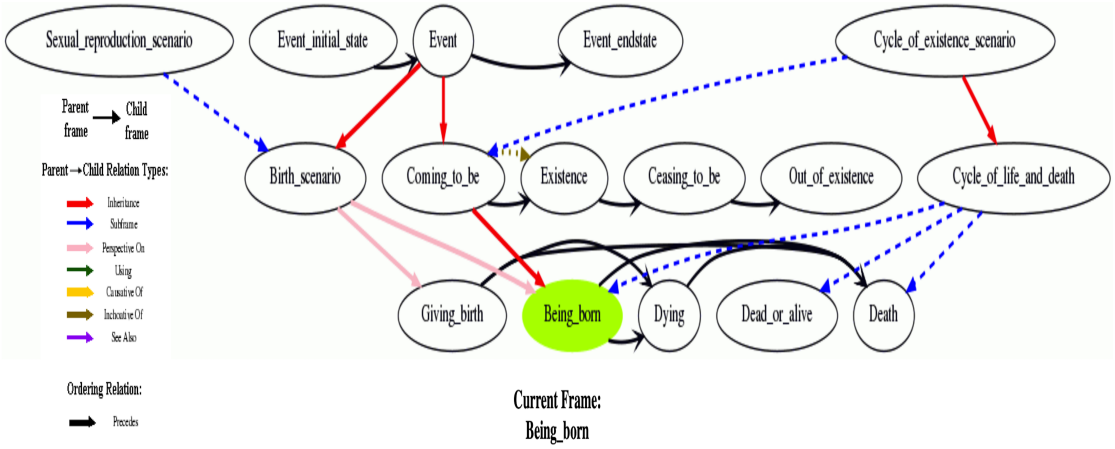


Figure 2.4.: Visualization of the `Being_born` frame of FrameNet using FrameGrapher.

Table 2.2.: Statistics of FrameNet database.

Frames		**Lexical units (LUs)**	**In FrameNet**	**Finished**
Lexical Frames	1,075	Nouns	5,575	2,698
Non-Lexical Frames	149	Verbs	5,214	2,852
FE Relations	10,749	Adjectives	2,407	1,368
Frame Relations	1,878	Other POS	490	67
FEs/Lexical Frame	9.7	LUs/Lexical Frame	12.7	—
FEs in Lexical Frames	10,478	—	—	—
Total Frames	1,224	Total LUs	13,686	6,985

The paraphrase database (PPDB) (Ganitkevitch et al., 2013) is a lexical database with millions of paraphrases automatically extracted from bilingual parallel corpora. It is available in XXXL, XXL, XL, L, M and S sizes with 77M, 23.1M, 7.8M, 3.4M, 1.4M and 0.7M pairs of paraphrases, respectively. The acquisition of paraphrases is based on bilingual pivoting method of Bannard and Callison-Burch (2005) — if two strings from the same language translate to the same string in another language, then they are assumed to be synonymous. Paraphrase pairs in PPDB fall into lexical (single word to single word), phrasal (multi-word to single/multi-word) and syntactic (paraphrase rules containing non-terminal symbols) types. Furthermore, each pair (e_1, e_2) is assigned with entailment relations based on the theory of natural logic defined between pairs of natural language expressions, such as:

- Equivalence: when e_1, e_2 are semantically equivalent, e.g., *distant* $\equiv$ *remote*.
- Forward Entailment or Hyponymy: when e_1 entails e_2, e.g., *glasses* $\sqsubset$ *sunglasses*.
- Reverse Entailment or Hypernymy: when e_2, entails e_2, e.g., *sneaker* $\sqsupset$ *footwear*.
- Exclusion or Contradiction: when e_1, e_2 are either opposite or represent alteration, e.g., *close* $\neg$ *open* or *cat* $\neg$ *dog*.
- Other relation: when e_1, e_2 are related by something other than entailment, e.g., *swim* $\sim$ *water*.
- Unrelated: when e_1, e_2 are independent, e.g, *car* $\#$ *family*.

BioVerbNet (Majewska et al., 2021) is a specialized semantic lexicon of verbs, that aims to accurately represent their meaning when found in biomedical texts, and to improve the performance on biomedical NLP tasks. Initially, the verbs were automatically assigned into groups using a neural classification approach (Chiu et al., 2019b), and then they were further verified and annotated by domain experts. Following VerbNet's (Kipper et al., 2008) guidelines 693 verbs were organized in 22 top-level classes and 117 subclasses (Figure 2.5) based on shared semantic and syntactic properties. Moreover, each subclass is provided with syntactic frames that showcase the possible surface realizations of the member verbs' arguments via usage example sentences. The sentences are annotated with semantic roles. Table 2.3 displays the semantic and syntactic annotations of the subclass verb *repair*, that is part of the *creation and destruction* class.

Table 2.3.: Semantic and syntactic annotations of the subclass verb *repair* in BioVerbNet. Verb class members are underlined.

Example sentences	**Semantic Role**	**Syntactic Frame**
Newts can regenerate their organs	Agent <+plural> {can} V Patient <+plural>	NP.AGENT V NP
Adult zebra fish regenerate their caudal fin following partial amputation	Agent <+plural> V Patient {following} Source	NP V NP PP

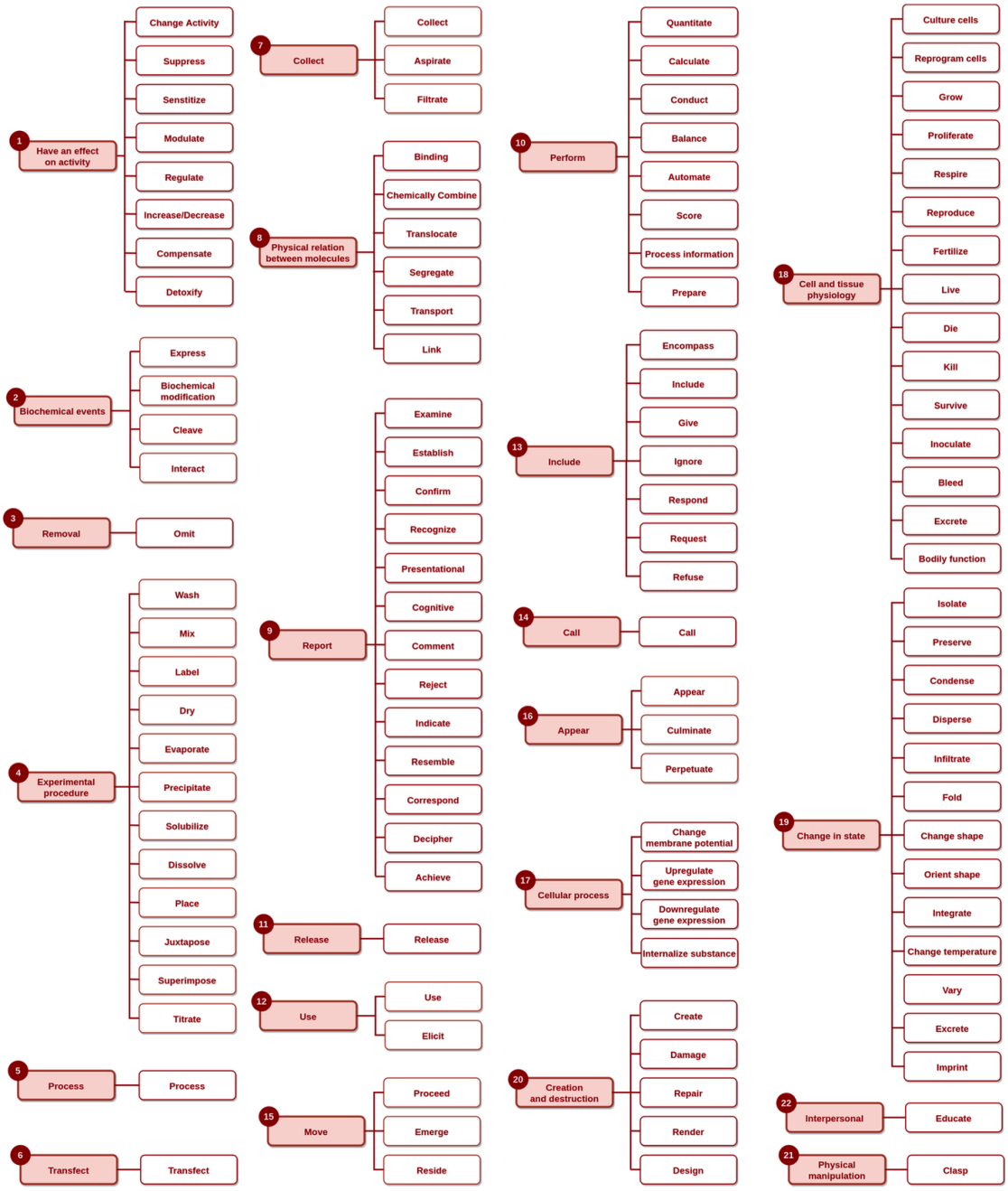


Figure 2.5.: Visualization of BioVerbNet semantic classes (Majewska et al., 2021).

We point to Yan et al. (2018) for an extensive review of existing KGs.

2.2.3. Graph embeddings

To date, encoding information related to the structure of a graph into a machine learning model is an open challenge. Traditional approaches often lean on user-defined heuristics such as kernel functions (Vishwanathan et al., 2010) or graph summary statistics, e.g., node degrees or clustering coefficients (Bhagat et al., 2011). However, these approaches are quite limited as the features they provide cannot be adjusted during training, and their design process can be expensive and time-consuming. Similarly to word embeddings (Section 2.1.2), modern data-driven approaches attempt to embed nodes, relations, sub-graphs or entire graphs as points in a low-dimensional vector space, namely graph embeddings, such that geometric relationships in the latent space reflect the structure of the original graph, e.g, node position in the graph or local graph neighborhood structure. According to Hamilton et al. (2017) node embedding methods can generally be viewed from the *encoder-decoder* perspective, i.e, an encoder function that embeds nodes into low-dimensional vectors, and a decoder function that utilizes these vectors in order to generate information about the graph. Besides the encoder-decoder functions, what makes the difference between these methods is the definition of a pairwise node similarity measure, and a loss function to train the underlying model. Premature techniques are based on matrix-factorization for dimensionality reduction. One of the most popular among these is Laplacian eigenmaps (Belkin and Niyogi, 2001), where the goal is to preserve the local geometry such as nearby nodes in the graph remain close — in terms of Euclidean distance — in the latent space. Follow up works have attempted to learn representations of nodes such that the inner product between node embeddings is approximately close to some deterministic node similarity measure, e.g, the adjacency matrix (Ahmed et al., 2013; Cao et al., 2015) or neighborhood overlap (Ou et al., 2016). Another family of approaches to learn node representations rely on the idea of random walks, i.e., if nodes appear frequently on short random walks over the graph, they should have similar embeddings. In contrast to factorization-based methods, random walks employ a stochastic node similarity measure, e.g., the probability of visiting node j starting from node i on a fixed length random walk, like in Deepwalk (Perozzi et al., 2014) and node2vec (Grover and Leskovec, 2016). However, all of the previous methods share certain limitations. More specifically, the resulting node embeddings are unique and independent of each other, thus there is no parameter sharing between nodes. The encoder part is simply an embedding lookup matrix, which can lead to scaling issues as the size of the graph grows. Furthermore, graph node attribute information, such as node position or description, is not taken into account. Lastly, the encoder is limited in the sense that it cannot generate embeddings for nodes that were not encountered during training. DNN based approaches (Cao et al., 2016; Wang et al., 2016) were introduced to deal with some of these drawbacks, by compressing local neighborhood information of nodes into low-dimensional vectors with the use of autoencoders (Hinton and Salakhutdinov, 2006). A more effective family of approaches, called convolutional, learns node representations by aggregating the information from a node local neighborhood. The embeddings are trained in an iterative fashion by incorporating information from neighbouring nodes at each iteration. These include graph convolutional neural networks

(Kipf and Welling, 2017, 2016; Su et al., 2021a; Schlichtkrull et al., 2018). Graph neural networks (Scarselli et al., 2008; Li et al., 2016; Gilmer et al., 2017) learn sub-graph embeddings, i.e., low-dimensional vector representations over a set of nodes and edges. Finally, there exist approaches for decoding the relations of a graph, which is particularly useful in predicting new edges in the graph. We refer to Hamilton et al. (2017), Nickel et al. (2015), and Dai et al. (2020) for comprehensive reviews on embedding graph nodes and relations/edges.

2.3. Incorporation of symbolic knowledge into neural networks

Although training a deep learning based NLP system on data specific to a domain of interest allows us to obtain representations of concepts that are informative about certain tasks, encoding symbolic knowledge or imposing constraints in the model is not trivial. There is no standard recipe on how to incorporate external symbolic knowledge into a neural network. Examples vary from injecting knowledge at the input level, e.g., by replacing target words in the input sentence with related ones in a given graph, modifying the training objective function to constrain the embedding space, to adapting the architecture of the model or aggregating the embeddings of different types of learners.

2.3.1. Joint and post-hoc methods

We begin by reviewing approaches that integrate knowledge into distributional word embeddings by retraining the model from scratch using a modified training objective. We refer to these approaches as *joint* methods. For example, Levy and Goldberg (2014) proposed to replace the classical bag-of-words contexts in the word2vec skip-gram model by dependency-based contexts, and showed that the resulting embeddings better reflect the syntactic similarities between words. Similarly, Boag and Kané (2017) added pairs of contexts between words and unique concept identifiers taken from the UMLS biomedical ontology (Bodenreider, 2004), to enforce a connection between the embeddings of clinical concepts that are known to be related to each other. Liu et al. (2015, 2018) learned semantic word embeddings by combining word2vec with rank or hierarchical structure rules mined from WordNet. Dict2vec (Tissier et al., 2017) is another extension of the skip-gram model, that builds pairs of contexts between words from word definitions in dictionaries, showing improvements on word similarity and text classification tasks. In another approach, Yang and Mitchell (2017) modified a BiLSTM recurrent neural network (Schuster and Paliwal, 1997) to take into account information coming from the WordNet and NELL (Mitchell et al., 2018) knowledge bases. To this end, they employed an attention mechanism that computes the relevance of candidate concepts from the knowledge base to the current input, and a second component that decides whether to exploit this information or not, and they reported improvements on both entity and event extraction tasks. Jiang et al. (2018) learned word embeddings for readability assessment, i.e., evaluation of readability of texts in terms of scores or levels, by optimizing a hybrid

loss function. One the one hand, the model predicts reading difficulty contexts from a KG, and on the other hand it predicts bag-of-words contexts from a corpus, using the skip-gram model with negative sampling. Nonetheless, joint methods come with the downside that they are model-specific, and often time-consuming since they require retraining the system afresh.

Post-hoc methods surpass these limitations, since knowledge is inserted in the word embeddings after training, regardless of the embedding approach used to obtain them. The most popular technique among these is retrofitting (Faruqui et al., 2015). This is a graph-based approach that, given a semantic lexicon, i.e., a KG whose nodes represent words and edges represent relations between them, tries to reposition the word embeddings in such a way that they become closer — under some distance metric — to neighborhood embeddings in the graph. More formally, given an initial set of distributional word embeddings $\hat{Q} = (\hat{q}_1, \ldots, \hat{q}_n)$, the objective is to obtain new distributional embeddings $Q = (q_1, \ldots, q_n)$ for the same words that minimize the following function:

$$\mathcal{L}(Q) = \sum_{i=1}^{n} \left[a_i ||q_i - \hat{q}_i||^2 + \sum_{(i,j)\in\mathcal{E}} b_{ij} ||q_i - q_j||^2 \right] \tag{2.12}$$

where $\mathcal{E}$ is the set of all related word pairs (i, j) in the graph, and a_i and b_{ij} control the strength of the corresponding terms in $\mathcal{L}$. Equating to zero the partial derivative of $\mathcal{L}$ with respect to q_i results in the following update rule:

$$q_i = \frac{a_i \hat{q}_i + \sum_{j:(i,j)\in\mathcal{E}} b_{ij} q_j}{a_i + \sum_{j:(i,j)\in\mathcal{E}} b_{ij}}. \tag{2.13}$$

Initially, Faruqui et al. (2015) considered a single type of relation between words, namely "similarity". Subsequent approaches have extended retrofitting to account for "dissimilarity" relations (Mrkšić et al., 2016, 2017; Lengerich et al., 2018) and ordering (ranking) between the relations (Ferret, 2017). Along these lines, Vulić et al. (2018) proposed a method that exploits lexical resources to improve the embeddings of all words in the vocabulary, including those which do not appear in the semantic lexicon. Their embeddings showed improvements in dialogue state tracking and lexical text simplification tasks in three different languages. We refer to Roy and Pan (2021) for a comprehensive review of approaches that integrate symbolic knowledge into distributional word embedding systems.

By default, all of the above retrofitting methods can only be applied to distributional word embeddings, i.e., a single representation vector per word. When we shift to contextualized embeddings, each word in the vocabulary can have a different representation in each sentence. An attempt to retrofit contextualized embeddings coming from ELMo (Peters et al., 2018) is presented in the Paraphrase-aware Retrofitting (PAR) (Shi et al., 2019) method. More specifically, PAR learns an orthogonal transformation matrix that pulls closer the embeddings of words in paraphrased contexts, and separates those in unrelated contexts. However, this approach is limited to pairs of paraphrased contexts and cannot benefit from different sources of linguistic information. More recently, Bihani and

Rayz (2021) utilized retrofitting (Faruqui et al., 2015) to inject word sense information into the contextualized embeddings of BERT, among others. Although they empirically show improved word sense disambiguation capabilities, their framework is only applicable to intrinsic evaluation tasks. To our knowledge, there is no existing method for contextualized embeddings that takes full advantage of the benefits of retrofitting. In Chapter 3, we will present our work on refining BERT contextualized embeddings using semantic lexicons, that constitutes a first step to bridging the gap between contextualized embeddings and retrofitting.

2.3.2. Enforcement of logical constraints into neural networks

Neural-symbolic computation is a research field that deals with the integration of logic and symbolic reasoning into neural networks, and has lately received a lot of attention due to the prominence of deep learning (Raedt et al., 2020). In particular, a number of approaches focus on integrating knowledge expressed as logical rules, which can act as a form of regularization, or assist in limited data settings. These approaches can be further categorized based on the type of logic that is being employed.

For example, Hu et al. (2016) adopted a meta-learning approach by simultaneously training a student network to imitate the predictions of a teacher network that are expressed in terms of explicitly first-order logic (FOL) rule constraints as regularization terms in the training loss. Their framework showed improvements applied on sentiment analysis and named-entity recognition tasks. Rocktäschel and Riedel (2017) introduced Neural Theorem Provers, an automated knowledge base completion approach that is able to induce FOL rules using gradient descent, which provide an interpretable representation of what the model has learned, and enable the neural network to perform multi-hop reasoning over facts in various knowledge bases, e.g., if Abe is the father of Homer and Homer is a parent of Bart, infer that Abe is a grandfather of Bart. Similarly, Neural LP Yang et al. (2017) learns FOL rules for knowledge base reasoning building upon a differentiable probabilistic logic framework called TensorLog (Cohen et al., 2020), where inference tasks are compiled into sequences of matrix numerical operations. They employed an LSTM with a differentiable memory component for learning to compose such operations, and computed confident scores for each logical rule via attention. Neural LP shows improvements on knowledge base completion and question-answering tasks. The aforementioned approaches are limited to knowledge base reasoning and do not scale to a large number of complex rules.

Dong et al. (2019) address those limitations in Neural Logic Machines, a more general framework that combines inductive learning and logic reasoning. Tensors are used to represent logic predicates, while neural networks are used as function approximators of logic operations, e.g. logical AND and OR. The resulting system displays promising results in a plethora of tasks ranging from family tree and general graph reasoning, to decision making such as sorting arrays and finding shortest paths. Along the same lines, Dai et al. (2019) proposed Abductive Learning, a system that tries to integrate symbolic domain knowledge as FOL clauses via logical abduction, in order to provide feedback and correct possible mistakes in the predictions of the neural model. Their approach

outperformed the Transformers network and BiLSTM on resolving unknown mathematical operations from image data of hand-written equations, and can be further adapted to more complex problems such as the n-queens task. Moving on from purely logic, there exist probabilistic approaches to integrate knowledge into neural networks. Perhaps one of the most prominent among these is Markov Logic Networks (Richardson and Domingos, 2006), where probabilistic logic determines the strength of the available knowledge in a given domain expressed as FOL formulas. The logic is used as constraints mapped to an undirected graphical model, and the weights indicate how important the constraints are. Rocktäschel et al. (2015) jointly learned embeddings of relations and entity-pairs by casting facts and logical background knowledge as FOL formulas, and then optimized a loss function based on the marginal probabilities that a given formula holds under the model. In the same manner, Demeester et al. (2016) learned distributed representations of relations of facts, used for knowledge base completion, by incorporating implications rules based on the hypernymy relation of WordNet, e.g., $professorAt \rightarrow employeeAt$, into the loss function of the model. In another approach, Xu et al. (2018) focused on multi-class classification with neural networks subject to logic constraints. They defined a semantic loss based on the probability that the constraints are satisfied, and used that as a regularizer whenever the logical theory or constraints were violated. A general framework that integrates logic, neural networks and probability is DeepProbLog (Manhaeve et al., 2018). It supports both symbolic and sub-symbolic reasoning, program induction and probabilistic logic programming by extending the probabilistic logic programming language ProbLog (De Raedt et al., 2007) with neural predicates. Relational Neural Machines (Marra et al., 2020) combine supervised learning with deep neural networks and FOL symbolic reasoning as in Markov Logic Networks into a single graphical model.

Although probabilistic approaches provide fast inference once the training is completed, they often suffer from slow optimization. Therefore, there exist alternative approaches that reside on fuzzy logic, where the logical operators are converted into real valued functions, and the Boolean truth values are translated into confidence values in the continuous $[0, 1]$ interval. However, some properties from probabilistic logic are not necessarily preserved in fuzzy logic, e.g., the transitivity property $A \rightarrow B, B \rightarrow C \Rightarrow A \rightarrow C$ may not be guaranteed. Examples of such frameworks include Probabilistic Soft Logic (Bach et al., 2017), Logic Tensor Networks (Serafini and Garcez, 2016; Donadello, 2018), Semantic based Regularization (Diligenti et al., 2017), LYRICS (Marra et al., 2019a,c), Deep Logic Models (Marra et al., 2019b), or the framework by Roychowdhury et al. (2021).

2.3.3. Learning and reasoning via analogies

As discussed in Section 1, symbolic knowledge exists in scientific theory, taking the form of e.g., axioms, theorems or lemmas, and it is usually expressed with mathematical symbols and formulas. In the case of analogies, such theories can be exploited in different ways in order to teach deep learning models how to reason, adapt, and deal with tasks that require analogy-making. Evidence of analogical reasoning with distributional word embeddings was provided by Mikolov et al. (2013); Levy and Goldberg (2014). It was

shown that these embeddings can model relations in the data through vector differences such that if the objects A, B, C, D are in analogical proportion, then the differences of their respective embeddings (denoted by boldface), i.e., $(\mathbf{B} - \mathbf{A})$ and $(\mathbf{D} - \mathbf{C})$ ought to be similar. A famous example where this was proven to hold true is the analogy *"man is to woman as king is to queen"*. What is really striking about this is that word2vec was not explicitly instructed to model analogies, yet this property came out as a side product of the training.

Sadeghi et al. (2015) attempted to solve visual analogy questions of the form *"image A is to image B as image C is to ?"* or equivalently $A : B :: C : ?$, where the goal is to find an image D that respects the analogy. More specifically, given an analogy quadruple (A, B, C, D) of images, a CNN (LeCun et al., 1998) with Siamese quadruple architecture is used to embed each image. Then, a transformation between image pairs (A, B) and (C, D) learns to map each pair to a vector representation. This is computed using the normalized differences of the image embeddings in the respective pair. The optimization problem utilizes a margin-based contrastive loss function, that encourages the transformations of pairs that are in analogy to be close in the embeddings space, and pulls apart those that are dissimilar. Unlike Sadeghi et al. (2015) who only considered visual inputs, Peyre et al. (2019) developed a system that is able to detect relations in images represented in the form of *(subject, predicate, object)* triplets, e.g., *(person, ride, dog)*. To do so, they learned both visual and language representations of the subject, predicate, object and the whole triplet, using pre-computed appearance features from a CNN object detector, and pretrained word2vec embeddings respectively. These were projected into a common d-dimensional visual-semantic embedding space, where the training objective was to pull together visual-language representation pairs that match, and pull apart those that do not. At inference time, based on a query (target) triplet which has not been seen during training, the model has to retrieve an image described by that triplet. This is done by first finding similar training (source) triplets based on the cosine similarity of their corresponding subject, predicate, and object embeddings. Then, given a source triplet, an estimate of the target triplet is computed using an analogy transformation, inspired by Reed et al. (2015), that indicates how to obtain a latent representation of the target triplet that is analogous to the source triplet. Intuitively, this is similar to the idea of arithmetic operations with word embeddings, e.g., *"king" - "man" + "woman" = "queen"*. But in this case *"person ride horse" - "horse" + "dog" = "person ride dog"*. Essentially, the analogy transformation is a two layer feed-forward network that it is optimized in a similar fashion as the visual-language representations; by pulling closer the visual representation of the target image to its corresponding language representation obtained via analogy transformation, but pushing away pairs that do not match. In a similar line of work, Lu et al. (2019) combined visual and language input representations to solve analogy completion. Given a set of four possible images the system must select an image D that turns the relation $A : B :: C : D$ into a valid analogy. To do so, they used ResNet (He et al., 2016) to embed A, B, C and all four candidate images, and picked the one that results in minimum cosine distance between the differences $(\mathbf{B}-\mathbf{A})$ and $(\mathbf{D}-\mathbf{C})$. Then, a BART model (Lewis et al., 2020) trained to

capture semantic relations using word pairs of pretrained word2vec embeddings, is used to generate a posterior probability vector that a word-pair constitutes a relation given a set of 258 learned relations during training. Using this vector as a representation of the relations $A : B$ and $C : D$ respectively, BART selects the candidate D that minimizes the cosine distance between $A : B$ and $C : D$ for all available candidates D. In the end, the final answer is a weighted average between the similarity measures provided by ResNet and BART.

In another work, Lim et al. (2019) proposed a CNN architecture for detecting semantic analogies, where they framed the task as an image classification problem. CNNs are good at capturing high level features in pictures, hence they detect analogical proportions by stacking the embeddings of **A**, **B**, **C** and **D** into an image and feeding it in the CNN. In the same spirit, Alsaidi et al. (2021a) adapted the previous model to detect morphological analogies in different languages. Training a character-based CNN with data augmentation using the properties of analogical proportions, they outperformed results of state-of-the-art symbolic approaches (Murena et al., 2020; Fam and Lepage, 2018). Moreover, they highlighted the effect of transfer learning using analogy between languages that share commonalities, e.g., the same alphabet. However, these works are restricted to solving analogies of word-pairs using distributional word embeddings. Afantenos et al. (2021) attempted to identify analogical proportions between sentences. To this end, they proposed a more relaxed definition of analogical proportions that is better suited for sentences by substituting the central permutation property ($A : B :: C : D \rightarrow A : C :: B : D$) with that of internal reversal ($A : B :: C : D \rightarrow B : A :: D : C$). However, they leave LLMs for future work. More recently, Ushio et al. (2021a) distilled relation embeddings between word pairs directly from BERT. To that end, they finetuned BERT such that the embeddings of word pairs belonging to the same relation class are closer than those belonging to different classes. Their method outperformed state-of-the-art methods on several analogy and relation classification benchmarks. In a following study, Ushio et al. (2021b) assessed the extent to which LLMs are capable of solving analogies without further finetuning. Their results demonstrate that LLMs can detect analogies but are sensitive to the choice of the hyperparameters. Furthermore, they show that such models are limited when it comes to more complex relations, and often perform worse than traditional word embedding systems. Although both works consider the use of contextualized embeddings, the training datasets consist of pairs of words and thus, manually or automatically generated templates (prompts) are required to transform them into sentences. Garneau et al. (2021) attempted to use analogical reasoning as means to measure global consistency of multilingual BERT (mBERT) (Conneau et al., 2018; Devlin et al., 2019) embeddings, where global consistency is defined as the extent to which the embeddings reflect semantic relations independently of scale. They introduced WiQueen, a multilingual analogy dataset across 11 languages with 78,000 analogies extracted from Wikipedia, and used it to train a four-way Siamese BERT architecture with a contrastive loss function in order to impose global consistency for cross-lingual transfer. They reported improvements on analogy retrieval intrinsic task, and bilingual dictionary induction and sentence retrieval extrinsic tasks. We refer to (Mitchell, 2021) for a comprehensive survey on prominent

AI approaches that attempt to model analogies.

To our knowledge, there is no existing method that combines analogies with contextualized embeddings to address word disambiguation tasks. In Chapter 4, we will combine BERT with the proposed architecture by Lim et al. (2019) to solve target sense verification (TSV) via analogy detection. Unlike previous works that only solve analogies of word-pairs, our setting will be more challenging as we will consider more complex structures such as special tokens, words in context, list of words, full sentences or their combinations.

2.3.4. Transformer knowledge-aware large language models

Transformer based language models have become the norm in NLP, achieving state-of-the-art performance in several benchmarks. An interesting property of such models is that they possess a great amount of unstructured knowledge due to pretraining on large volumes of unlabeled text. However, their ability to reason over that knowledge is quite limited, and they lack interpretability. Therefore, to address those limitations there exist various approaches that try to incorporate structured, symbolic knowledge into Transformer based architectures. Following the categorization by Colon-Hernandez et al. (2021), approaches to inject knowledge into Transformer-based pretrained language models are mainly divided into four types, namely, input, architecture, output based, or their combination, as illustrated in Figure 2.6.

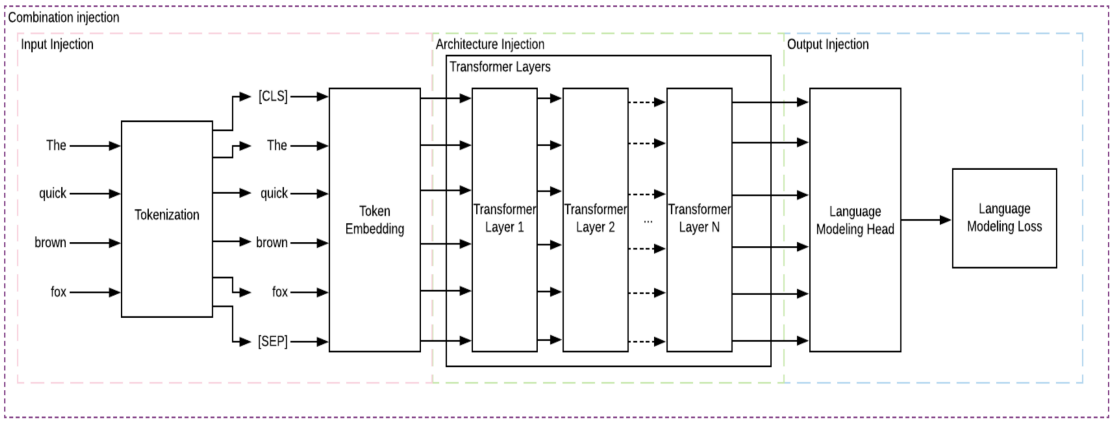


Figure 2.6.: Schematic illustration of the different types of knowledge injection approaches in Transformer-based LLMs by Colon-Hernandez et al. (2021).

Input based approaches typically inject knowledge at the preprocessing stage by modifying the structure or the data themselves, or the layers that come before the Transformer. For example, AMS (Ye et al., 2019) creates a dataset for question-answering by aligning triples from the ConceptNet (Speer and Havasi, 2012) KG, with relevant sentences from Wikipedia. The head or tail entity in each sentence is masked with a special token (`[QW]`), and possible answers are generated by looking at neighbouring triples that share the same masked token and relationship. Next, the sentences are transformed into

questions which are then concatenated with all possible answers to create multi-choice question-answering samples. For example, considering the triple *(population, AtLocation, city)* and the sentence *"The largest city by population is Birmingham, which has long been the most industrialized city."* the resulting sample is **question:** The largest `[QW]` by population is Birmingham, which has long been the most industrialized city? **candidates:** city, Michigan, Petrie dish, area with people inhabiting, country, where the correct answer is underlined. The knowledge which is encoded in the data as question-answer pairs is then injected into BERT by means of standard pretraining on the particular dataset. COMET (Bosselut et al., 2019) is a GPT based model that is able to perform KG completion on commonsense KGs. The model is trained on subject-relation-object triples from ConceptNet and ATOMIC (Sap et al., 2019), where the subject and object entities are represented as natural language phrases, e.g, (throwing a party, causes, having fun). The training objective consists of generating the tokens corresponding to the object entity in the training triples, given the concatenation of the subject and relation tokens as input. According to human evaluation, COMET attains KG completion of novel and high quality commonsense knowledge even on nodes and relations that were not encountered during training. In a follow up work, Hwang et al. (2021) introduced ATOMIC$^{20}_{20}$ a KG meant to be challenging for language models with better coverage of relations compared to existing commonsense KGs. They instantiated COMET with GPT-2 or BART (Lewis et al., 2020), trained it on different commonsense KGs and compared it against GPT-3 and GPT2-XL. The results demonstrate that models trained on ATOMIC$^{20}_{20}$ can generate accurate knowledge on unseen entities and events based on common metrics for text generations and through human evaluation. Along the same lines, Bosselut et al. (2021) used COMET to dynamically build a KG of commonsense answers given some context, that can be used for reasoning in a commonsense question-answering scenario. Poerner et al. (2020) introduced E-BERT, a model that injects factual knowledge into BERT by aligning Wikipedia2Vec (Yamada et al., 2016) entity vectors with BERT wordpiece vectors, without requiring to change the architecture of the encoder nor pretrain the model. Taking for example the BERT tokenized input sentence "*The native language of Jean Mara ##is is* ***`[MASK]`***.", they either concatenate "*The native language of* ***Jean_ Marais*** */ Jean Mara ##is is* ***`[MASK]`***." or replace "*The native language of* ***Jean_ Marais*** *is* ***`[MASK]`***." the wordpiece entity tokens with the corresponding entity ID from Wikipedia (highlighted in bold). E-BERT demonstrates improvements on fact completion on LAMA dataset compared to more sophisticated knowledge-enhanced pretrained language models, such as KnowBert (Peters et al., 2019) and ERNIE (Zhang et al., 2019).

The next type of knowledge injection approaches focuses on architecture modifications of the language model, e.g., modifying existing or adding extra layers to combine knowledge with the contextual representations of the Transformer. For instance, KnowBERT (Peters et al., 2019) incorporates WordNet and part of Wikipedia into BERT, showing the ability of the model to recall facts from the databases, improving downstream relation extraction, entity typing and word sense disambiguation tasks at the same time. This is done by inserting a knowledge attention and recontextualization module in between two

middle Transformer blocks, that is able to enhance entity-span representations in the input text with retrieved graph entity embeddings from the knowledge bases via entity linking. In another approach Lauscher et al. (2020a) introduced Retrograph, a model that incorporates knowledge into BERT using the adapter-based finetuning paradigm, i.e., introducing a set of additional parameters into the encoder and only tuning these, while keeping all the original transformer parameters fixed. To do so, they trained the adapter-augmented BERT using the standard masked language modeling pretraining objective on data from ConceptNet and the Open Mind Common Sense corpus (Singh et al., 2002), showing improvements on natural language inference tasks that require commonsense knowledge.

Moving on, output based approaches integrate knowledge via modifications of the output structure, or by introducing custom loss functions. An example of such model is SemBERT (Zhang et al., 2020), that combines word-level contextual representations from BERT with embeddings of predicate-argument structures derived from a semantic role labelling system. Both types of embeddings are concatenated and fed into a linear projection layer to form a semantics-enriched joint representation used for downstream tasks. Similarly, KEPLER (Wang et al., 2021b) encodes textual entity descriptions and entities from WordNet and Wikidata into the same space, by jointly optimizing a knowledge base completion objective and the standard masked language modeling objective from BERT pretraining.

However, the vast majority of methods fall into the combination of input, architecture, and output based approaches. Shen et al. (2020) integrate relational structured knowledge into BERT and RoBERTa by modifying the masked language modeling pretraining objective to an entity-level masking strategy. Informative entities are selected and then masked following an entity masking scheme guided by ConceptNet. Moreover, an auxiliary training objective based on negative sampling of entities from the KG is employed, in order for the model to learn to distinguish between positive and negative entities. Both tasks are combined to jointly train the model. Similarly, in LIBERT (Lauscher et al., 2020b) the external knowledge is encoded in BERT with the addition of lexical relation classification as a third pretraining objective. The knowledge is represented as word pairs of synonyms or hyponym-hypernym from WordNet and BabelNet (Navigli and Ponzetto, 2012) lexical resources. Positive and negative examples of these pairs are preprocessed by standard wordpiece tokenization and then fed into BERT, e.g., (mended, regenerated) takes the form `[CLS]` men #ded `[SEP]` reg #ener #ated `[SEP]`. Then, a softmax classifier is trained to predict whether the given pair constitutes a valid relation or not, using the embedding of the `[CLS]` token as input. BERT-MK (He et al., 2020) integrates graph contextualized medical knowledge from the UMLS (Bodenreider, 2004) thesaurus into BERT. The knowledge is preprocessed such that subgraph entities and relations are converted into sequences of nodes, that serve as input to a Transformer based model to learn node embeddings by minimizing a margin loss function. The resulting node embeddings are aggregated with language representations from another Transformer, using the same integration method as ERNIE. K-BERT (Liu et al., 2020a) injects triples from a KG into sentences to construct sentence trees that serve as input to BERT. For example, given

the sentence *"Tim Cook is currently visiting Beijing now."* and the set of triples (Apple, CEO, Tim Cook), (China, capital, Beijing), and (Beijing, is_a, City), the sentence tree becomes *"Tim Cook CEO Apple is visiting Beijing capital China is a City now."*. Since the structural information of the sentence is altered, the authors utilize soft-position embeddings to define the position of each token in the Transformer block. Furthermore, they use a visible matrix to control where each token can attend to, as a means to incorporate only relevant knowledge and prevent semantic deviations of the original sentence, e.g., in the previous example ensure that [Apple] and [China] would not interfere as the latter is only meant to modify the representation of [Beijing]. Another approach to dynamically select relevant knowledge of entities from Wikidata is CokeBERT (Su et al., 2021b). The model uses a semantic-driven graph neural network that, given an entity mention, filters out irrelevant information from the KG by assigning scores to neighbouring nodes and relations via an attention mechanism. The resulting entity embeddings are then fused with BERT token embeddings, and the system is trained using an additional denoising entity auto-encoder objective on top of the standard next sentence prediction and masked language modeling objectives. We refer to Colon-Hernandez et al. (2021) for a more extensive survey on knowledge-enriched Transformer based approaches, highlighting limitations and possible future avenues. We also point to Yang et al. (2021) for an in-depth review of knowledge-enhanced LLMs from the point of view of the granularity of knowledge, the method of knowledge injection, and the degree of symbolic knowledge parameterization.

3. On refining BERT contextualized embeddings using semantic lexicons

As discussed in Chapter 2 the introduction of word embeddings was a breakthrough in NLP. Early approaches based on the *distributional hypothesis* (see Section 2.1.2) provide a fixed embedding for each word. Recently, contextualized embedding systems like BERT (Devlin et al., 2019) have allowed the generation of context-dependent word representations, which substantially improve the performance on many downstream NLP tasks.

As discussed in Section 2.3.1, retrofitting (Faruqui et al., 2015) is a popular technique that modifies any set of pretrained distributional word embeddings to account for relational information encoded by a semantic lexicon. This is done as a post-processing step using the graph of relations obtained from the lexicon to update the word vectors. This method was proven to improve performance on various intrinsic and extrinsic evaluation tasks (Mrkšić et al., 2016, 2017; Ferret, 2017; Lengerich et al., 2018; Chiu et al., 2019a). However, these works only deal with non-contextualized word embeddings.

In this chapter, we aim to extend retrofitting to operate with contextualized word embeddings. More specifically, we propose two different methods that, as in the original retrofitting approach, make use of similarity relations between words in order to move the respective embeddings closer to each other in the latent space. The first method combines the embedding of a given test sentence with the embeddings of sentences involving similar words in the training set, while the second method replaces a word in the test sentence by all possible similar words and combines the resulting embeddings. We evaluate the proposed methods with BERT embeddings on three biomedical datasets for a relation extraction task and one movie review dataset for sentiment analysis, and compare them with an oracle topline and two baselines based on weighted majority voting and class posterior averaging respectively. We show that both methods do not substantially impact the performance for this task, and conduct a qualitative analysis to provide further insights on this negative result.

Consequently, the contributions of this chapter are:

1. the development of two approaches that extend the classical retrofitting algorithm to operate with contextualized embeddings,
2. exhaustive experiments with BERT embeddings on three biomedical datasets for relation extraction and one movie review dataset for sentiment analysis,
3. a qualitative study of the obtained results, that gives us some intuition on why the proposed methods do not significantly alter the performance of the overall system for that task on these datasets.

The chapter is organised as follows. We present the proposed methods in Section 3.1.

We describe the experimental evaluation setup in Section 3.2, and we present and analyze the obtained results in Section 3.3. We provide conclusions and discuss future work in Section 3.4.

3.1. Proposed contextualized embedding refinement methods

As in the conventional retrofitting approaches discussed in Section 2.3.1, we assume a vocabulary of words $\mathcal{V} = \{w_1, \ldots, w_n\}$ and an ontology Ω of semantic relations between words in $\mathcal{V}$. We can then represent Ω in the form of an undirected graph $(\mathcal{V}, \mathcal{E})$, where nodes correspond to words in $\mathcal{V}$ and edges $(w_i, w_j) \in \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ to semantic relations between nodes. Suppose that we have a contextualized word representation model $\mathcal{M}$, along with a training corpus $\mathcal{D}_{\text{train}}$ on which it is finetuned, and a test corpus $\mathcal{D}_{\text{test}}$ on which it is evaluated for a particular task.

3.1.1. Method A

The first proposed embedding refinement method, which we refer to as Method A, combines the contextualized embedding of a given word in the test set with the contextualized embeddings of all occurrences of all similar words in the training set. For example, assume that the given word is *better*, which has three neighbours in Ω: *best, improve* and *ameliorate*. By iterating over the training set, we identify all the examples for which a neighbouring word occurs and store the corresponding embeddings. Then, the retrofitting operation combines the original embedding for *better* and the embeddings of all occurrences of all the neighbouring words in a form of a weighted average.

More formally, let $\bar{q}_i \in \mathbb{R}^d$ be the contextualized embedding of word $w_i \in \mathcal{V}$ coming from $\mathcal{M}$ for a given test instance[1]. Let us further denote by $\mathcal{J}_i$ the set of words w_j which are adjacent to w_i according to Ω, and by $\mathcal{K}_j$ the set of training instances where w_j occurs. Then we define $\hat{q}_{jk} \in \mathbb{R}^d$ to be the contextualized embedding computed for all occurrences of w_j in $\mathcal{D}_{\text{train}}$, with index $k \in \mathcal{K}_j$. The index sets $\mathcal{J}_i$ and $\mathcal{K}_j$ vary dynamically for every word.

The goal is to learn a new embedding q_i that is close to $\bar{q}_i$ and to adjacent nodes in Ω under the $\mathcal{L}_2$ norm, by minimizing

$$\mathcal{L}(q_i) = ||q_i - \bar{q}_i||^2 + \sum_{j \in \mathcal{J}_i} \sum_{k \in \mathcal{K}_j} b_{ijk} ||q_i - \hat{q}_{jk}||^2. \tag{3.1}$$

The weights b_{ijk} should naturally depend on the number of neighbours $|\mathcal{J}_i|$ of w_i, and on the number of occurrences $|\mathcal{K}_j|$ of each neighbor w_j in $\mathcal{D}_{\text{train}}$. To this end, we define them as $b_{ijk} = c_{ij} \times d_{jk} = \frac{1}{|\mathcal{J}_i|^\alpha} \cdot \frac{1}{|\mathcal{K}_j|^\beta}$, $\alpha, \beta \in [0, \infty)$ where c_{ij} controls the contribution of each neighbour and d_{jk} controls the contribution of each of its occurrences. For example,

[1] For simplicity, $\bar{q}_i$ does not have a superscript for the test sentence as we only process one test sentence at a time.

$\alpha = \beta = 0$ results in equal weights $b_{ijk} = 1$ for all occurrences, while $\alpha = \beta = 1$ results in weights b_{ijk} that sum up to 1.

Equating the derivative of $\mathcal{L}$ with respect to q_i to zero and expressing the $\sum_k b_{ijk}\hat{q}_{jk}$ in terms of the mean $\mu_{\hat{q}_j}$ of all $\hat{q}_{jk}$ results in the following update rule:

$$q_i = \frac{\bar{q}_i + \sum_j \sum_k b_{ijk}\hat{q}_{jk}}{1 + \sum_j \sum_k b_{ijk}} = \frac{\bar{q}_i + |\mathcal{J}_i|^{-\alpha} \sum_j |\mathcal{K}_j|^{1-\beta} \mu_{\hat{q}_j}}{1 + |\mathcal{J}_i|^{-\alpha} \sum_j \mathcal{K}_j^{1-\beta}}. \tag{3.2}$$

The retrofitting operation therefore takes the form of a weighted average of the original embedding and the embeddings of all occurrences of all similar words in the training set.

3.1.2. Method B

The second proposed method, which we refer to as Method B, does not involve $\mathcal{D}_{\text{train}}$ at all. Instead, everything happens at test time. Using the same example as in Method A, we now check if the given word, e.g., *better*, is present in the test instance. If so, we replace it by a neighbouring word, one at a time, and compute the embedding for each new instance. The original embedding and the embeddings of all neighbouring words are aggregated by a weighted average operation.

Again, we utilise $\mathcal{M}$ to obtain the embedding $\bar{q}_i$ of word w_i for a specific sentence in $\mathcal{D}_{\text{test}}$. In addition, we derive one embedding $\hat{q}_j$ for every word w_j which is adjacent to w_i according to Ω. To do so, we create a new sentence by replacing w_i with w_j in the test sentence, and repeat for every adjacent node of w_i in Ω. The objective is once again to learn a new vector q_i that is close to both $\bar{q}_i$ and all $\hat{q}_j$ under the $\mathcal{L}_2$ norm by minimizing

$$\mathcal{L}(q_i) = ||q_i - \bar{q}_i||^2 + \sum_{j \in \mathcal{J}_i} b_{ij} ||q_i - \hat{q}_j||^2. \tag{3.3}$$

Similarly to the above, we define the weights as $b_{ij} = \frac{1}{|\mathcal{J}_i|^\alpha}$, $\alpha \in [0, \infty)$.

Equating the derivative of $\mathcal{L}$ with respect to q_i to zero and expressing the $\sum_j b_{ij}\hat{q}_j$ in terms of the mean $\mu_{\hat{q}_j}$ of all $\hat{q}_j$ results in the following update rule:

$$q_i = \frac{\bar{q}_i + \sum_j b_{ij}\hat{q}_j}{1 + \sum_j b_{ij}} = \frac{\bar{q}_i + |\mathcal{J}_i|^{1-\alpha} \mu_{\hat{q}_j}}{1 + |\mathcal{J}_i|^{1-\alpha}}. \tag{3.4}$$

Again, the retrofitting operation takes the form of a weighted average of the original embedding and the embeddings of all neighbouring words.

The main difference between both methods lies in the way we exploit the information coming from Ω. Method A typically results in a large number of neighbouring vectors $\hat{q}_{ik}$ that contain noise, since the context around the corresponding words differs from that of the test sentence in general. In contrast, Method B generates fewer neighbouring vectors $\hat{q}_j$ that share exactly the same context as the test sentence being processed.

3.2. Experimental setup

In this section, we first provide information about the data, the semantic lexicons and the contextual word embedding model we used. We then evaluate the proposed methods on two tasks: relation extraction from biomedical data and sentiment analysis of movie reviews. For comparison and consistency purposes, we chose these particular tasks as they have been previously used for evaluating retrofitting (Faruqui et al., 2015; Chiu et al., 2019a). Finally, we describe the experimental evaluation and suggest three alternative strategies for comparison.

3.2.1. Biomedical relation extraction

For the biomedical relation extraction task, we use the two semantic verb lexicons introduced by Chiu et al. (2019a), referred to as **annotated** and **expanded clusters**. The former contains 192 verbs that appear frequently in a corpus of 2,230 biomedical journal articles. It was annotated by domain experts and linguists that hierarchically classified the verbs into 16, 34 and 50 classes to reflect the different granularities of their semantics (Korhonen et al., 2006). The latter is an extended version of the previous that maintains the same hierarchy, but enriches the 50 classes to obtain 1,149 verbs in total. Furthermore, it consists the starting point for the construction of BioVerbNet (Majewska et al., 2021) as discussed in Section 2.2.2.2. We refer to each different version of the verb lexicons simply by adding the number of the verb classes next to its name, e.g., annotated-34.

ChemProt is a manually annotated corpus of relations between drugs/chemical compounds and genes/proteins mentions found in PubMed abstracts. The relations are categorized into ten classes from which only five are used during evaluation. The task is to predict whether a pair of such entities is related or not, and if so, output the type of relation.

The **DDI** corpus aims in the development of systems that can automatically detect drug entities and drug-to-drug interactions in biomedical text. The corpus itself consists of texts from the DrugBank database and abstracts from the MedLine database. Annotations were provided by domain experts that classified drug-drug interactions into four DDI types.

The **i2b2 2010** corpus promotes the study of extraction/classification/relations of medical problems, tests, and treatments. The data consist of discharge summaries collected from Partners Healthcare, Beth Israel Deaconess Medical Center, and the University of Pittsburgh Medical Center, where relations of medical problems-treatments were grouped into eight classes.

All three biomedical datasets are included in the Biomedical Language Understanding Evaluation benchmark (Peng et al., 2019), as well as the preprocessing codes for creating the training, development and test sets[2]. Furthermore, both the annotated and the expanded clusters are derived from biomedical texts, which makes them applicable to

[2] https://github.com/ncbi-nlp/BLUE_Benchmark

these data.

3.2.2. Sentiment analysis of movie reviews

For the sentiment analysis of movie reviews, we use the exact same semantic lexicons as Faruqui et al. (2015), namely, **FrameNet**, **PPDB** and two variants of **WordNet** which we refer to as $\textbf{WordNet}_{\text{syn}}$ and $\textbf{WordNet}_{\text{all}}$ (more details in Section 2.2.2.2). The size of these lexicons is relatively large, since they are general and contain knowledge about words which do not convey any sentiment, e.g., pronouns, prepositions, etc.. In order to focus on relevant words for the task, in conjunction with the semantic lexicons we utilize the **Bing Liu Sentiment Lexicon** (Hu and Liu, 2004), a domain-independent list of 6,786 adjectives that is manually created and that categorizes words as either positive or negative according to their sentiment.

SST-2 (Socher et al., 2013) is a collection of 11,855 sentences from movie reviews including human annotations of their sentiment. The goal is to classify a given sentence as either positive or negative. Since the test labels are not publicly available, we split the training set such that 13% of the sentences are used for testing and the remaining are used for training. The resulting test set has 462 positive and 438 negative reviews, while the training set has 3,148 positive and 2,872 negative reviews. Finally, we use the development set provided by the authors.

We report the performance for each dataset in terms of micro F_1-score for relation extraction, and accuracy for sentiment analysis, as Peng et al. (2019) and Wang et al. (2018), respectively. Some key statistics of all datasets are summarised in Table 3.1.

3.2.3. Retrofitting and BERT architecture

There are different locations within the architecture of BERT where retrofitting transformations can be applied. As described in Section 2.1.3.2, the model consists of 12 Transformer blocks followed by a pooling layer, i.e., a fully connected layer with a dropout layer and a tanh activation. Each block contains a sequence of transformations that is divided into layers. The output layer of each block consists of a linear transformation, followed by dropout and layer normalization. For both approaches we experiment with four different settings:

1. Retrofitting before layer normalization at Transformer block 11
2. Retrofitting after layer normalization at Transformer block 11
3. Retrofitting before layer normalization at Transformer block 12
4. Retrofitting after layer normalization at Transformer block 12

The motivation behind these choices is related to the complex architecture of the model. We hypothesize that the impact of any change into the embeddings would be more noticeable as we get closer to the output space, rather than in earlier layers of the model. Thus, we start experimenting at the pooling layer, which is the closest to the output space, but the results were not promising. Consequently, we move one step back at the output layer of the last Transformer block, and further back to the same place of the preceding Transformer block.

Table 3.1.: Summary statistics of all datasets. (*: corresponds to the non-relational case and it is not considered during evaluation).

Relations	Train	Dev	Test	Relations	Train	Dev	Test
ChemProt				**i2b2**			
CPR:3	768	550	665	PIP	659	96	1,448
CPR:4	2,251	1,094	1,661	TeCP	149	17	338
CPR:5	173	116	195	TeRP	903	90	2,060
CPR:6	235	199	293	TrAP	800	85	1,732
CPR:9	727	457	665	TrCP	171	13	342
False*	15,306	9,404	13,485	TrIP	49	2	152
Total	19,460	11,820	16,943	TrNAP	52	10	112
DDI				TrWP	24	0	109
DDI-advise	663	193	221	False*	17,226	1,910	36,707
DDI-effect	1,212	396	360	Total	20,033	2,223	43,000
DDI-int	146	42	96	**SST-2**			
DDI-mechanism	946	373	302	Positive	3,148	444	462
False*	15,842	6,240	4,782	Negative	2,872	428	438
Total	18,779	7,244	5,761	Total	6,021	872	900

In the retrofitting equations (3.1) or (3.3), we initially consider as $\bar{q}_i$ the embedding corresponding to the word token in the test sentence, but preliminary experiments show that this does not have an impact on the final performance. To verify this, we replace the embeddings of these individual words with random numbers, or even zeroes. Both cases do not affect the performance, indicating that the output classifier is not very much dependent on single word embeddings. Instead, we focus on the `[CLS]` token embedding which is a weighted linear average of all word embeddings in the test sentence, it is closer to the output space, and has a bigger impact on the final result. All $\hat{q}_{jk}$ in (3.1) correspond to the activations of the word token in training sentences, whereas all $\hat{q}_j$ in (3.3) correspond to the activations of the `[CLS]` token in modified test sentences.

3.2.4. Technical details

For the biomedical relation extraction, we choose BlueBERT (Peng et al., 2019) a specific variant of BERT that is further pretrained on PubMed abstracts (Fiorini et al., 2018) and clinical notes from the MIMIC-III database (Johnson et al., 2016), while for sentiment analysis of movie reviews, we experiment with the classical BERT (Devlin et al., 2019). In particular, for both tasks we select the $\text{BERT}_{\text{BASE}}$ release of the model, which makes use of the exact same configurations as in the original BERT, and we further finetune it on the downstream task for each dataset. We treat both tasks as sentence classification problems. The input sentence is fed into BERT which makes use of the `[CLS]` token

of that sentence to perform the classification. In particular, the `[CLS]` representation of the last Transformer block is passed into a feed-forward output layer that produces an estimation for each class. For biomedical relation extraction, the named entities are anonymized with predefined tags (e.g., `@GENE`, `@CHEMICAL` for ChemProt) as done by Lee et al. (2020). Regarding the process of selecting the appropriate verbs to retrofit, if the same verb appears more than once in the input sentence, we randomly select one. Since we have no way of knowing which instance of the verb is more appropriate given a sentence, we prefer the stochastic choice. Alternatively, we could have selected all verbs in a sentence but for consistency we simply pick one. Moreover, by default, BERT treats out-of-vocabulary words by splitting them into wordpiece sub-tokens, e.g., ['hemorrhage'] → ['hem', '##or', '##rh', '##age'], resulting into multiple representations for a single verb. Favouring a particular sub-token over the others, or selecting a specific pooling strategy to obtain a single representation of the verb is ambiguous, therefore we choose to ignore such cases. Finally, all verbs in the lexicons are present in their infinitive form, meaning that alternative forms of the verbs are not considered, e.g., activate ≠ activated. This could be treated by either expanding the lexicon to include other forms, or performing some sort of stemming to fuse all variations into a single word. However, questioning the lexicon resource is out of the scope of this work.

3.2.5. Grid search optimization

In order to find a good set of values for the retrofitting hyperparameters α, β, we perform a grid search using the development sets. For biomedical relation extraction, we use both annotated and expanded clusters, and we search for α and β in $[0, 2]$ with a step of 0.2. We do not proceed on testing Method A for SST-2, as it turns out to be inferior to Method B. A grid search plot of micro F_1-scores for Method A is provided in Figure 3.1. The white colour corresponds to the baseline score and the red asterisk indicates the best (α, β)-pair performance on the development set.

For sentiment analysis of movie reviews, we use all four lexicons in conjunction with Bing Liu's sentiment lexicon (see Section 3.2.2), while for relation extraction, we only used the annotated-34 and annotated-50 versions of the annotated clusters. This is due to the extensive amount of neighbouring verbs on the annotated-16 and the expanded clusters, which significantly increases the computational cost. Once again, we perform a grid search on the development sets, where we search for α in $[0, 2]$ with a step of 0.2. A grid search plot of accuracy scores for Method B is provided in Figure 3.2. The green colour bar indicates the best α-values on the development set, while the horizontal lines show the top performance of all proposed strategies (see Section 3.2.6).

3.2.6. Alternative classification strategies

In order to assess the ability of our method to leverage the information in the semantic lexicons, we perform the following experiment. First, we augment each dataset by adding all modified sentences that are generated by replacing the underlying word in the original sentence, with a neighbouring one from the semantic lexicon. This results to a

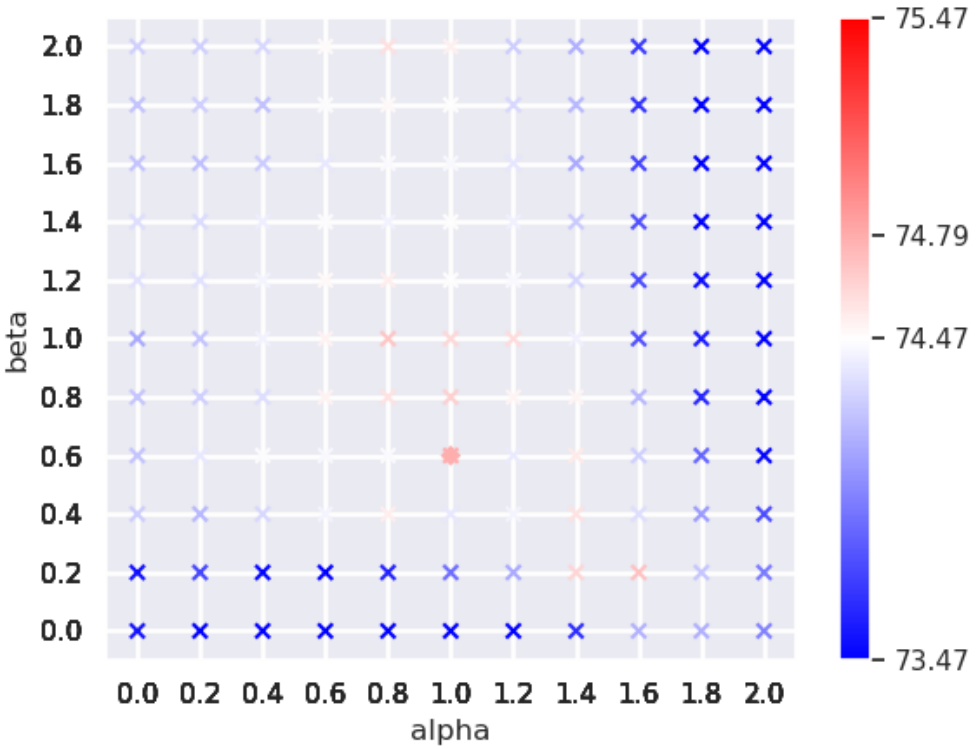


Figure 3.1.: Grid search plot of micro F_1-scores for Method A while retrofitting after layer normalization at Transformer block 11 using the expanded-16 clusters on ChemProt.

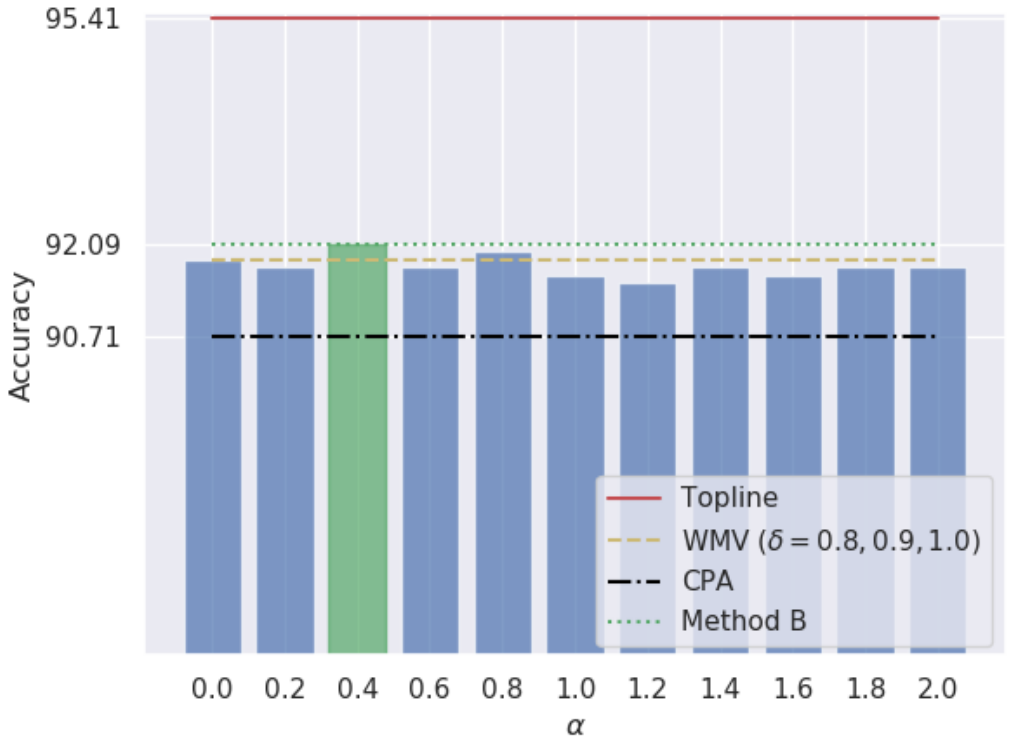


Figure 3.2.: Grid search plot of accuracy scores for Method B while retrofitting before layer normalization at Transformer block 12 using the PPDB semantic lexicon on SST-2. The horizontal lines indicate the top accuracy score for Method B and the alternative classification strategies.

number of augmented versions of the original datasets, equal to the number of semantic lexicons per task. Then, we use the originally finetuned BERT on the downstream task — BlueBERT for biomedical relation extraction or vanilla BERT for sentiment analysis of movie reviews (Section 3.2.4) — and score both original and modified sentences in the augmented dataset. Once we obtain a probability score for each sample in the augmented dataset, we compare it with the following alternative strategies:

- **Topline**: Always selecting the true class of a test sentence as the final prediction, if it was predicted by at least one of the original or the modified sentences.

- **Weighted majority vote (WMV)**: Picking the predicted class with the most occurrences as the final prediction out of the original and the modified test sentences. Here, we assigned a weight of 1 to the original and a weight of $\frac{1}{|S|^{\delta}}, \delta \in [0, 1]$ to each modified sentence, where $|S|$ is the total number of sentences for the current test input. We experimentally noticed that choices of δ outside $[0, 1]$ did not affect the final prediction.

- **Class posterior averaging (CPA)**: Averaging the probabilities of the predicted classes for both the original and the modified test sentences, and taking the class with the maximum probability as the final prediction.

To further clarify the dataset augmentation process and the use of the alternative strategies, we give the following example. Assume we are tackling sentiment analysis of movie reviews on the SST-2 dataset, and that we make use of WordNet as our semantic lexicon. Given the original sentence S_0: *"A sometimes tedious film."* that conveys negative sentiment, we first identify that *tedious* has 4 neighbouring words in the lexicon, namely, *wordy*, *long-winded*, *verbose*, and *windy*. After replacing *tedious* with every neighbour one by one, we construct a set of 4 modified sentences, $S_{01}, \ldots, S_{04}$, corresponding to S_0. We repeat the same process for each sentence in SST-2. Once the augmented dataset is built, we use the finetuned BERT on SST-2 to score each sentence in it. Since each label can be either positive or negative, the model outputs a set of binary probability scores (p_{pos}, p_{neg}) for each sentence. In the example above, we obtain $S_0 : (0.9997, 0.0003)$, $S_{01} : (0.9979, 0.0021)$, $S_{02} : (0.9992, 0.0008)$, $S_{03} : (0.9997, 0.0003)$, and $S_{04} : (0.4072, 0.5928)$. Based on these scores, topline will correctly select the negative class since it was predicted by at least one of the original or the modified sentences, i.e., S_{04}. However, weighted majority vote will output the positive class for all $\delta \in [0, 1]$ since on average it appears more frequently in the original and the modified sentences (4/5 times). Similarly, class posterior averaging will predict the same as weighted majority vote since the maximum mean average score per class $(\frac{0.9997+0.9979+0.9992+0.9997+0.4072}{5}, \frac{0.0003+0.0021+0.0008+0.0003+0.5928}{5}) = (0.88074, 0.11926)$ belongs to the positive class.

3.3. Results and qualitative study

In this section we present the results obtained from the grid search, and conduct additional experiments that give further insights on the reasons why the proposed methods yield a similar performance to the baseline model.

3.3.1. Grid search experimental results

After finding the best performing set of hyperparameters amongst all combinations of lexicons, Transformer blocks, and positions that were tested on the development set, we evaluate the corresponding model on the test set. The results for both retrofitting approaches are displayed in Table 3.2.

At first sight, both approaches seem to have no significant impact compared to the baseline performance. More specifically, Method A results in a decrease of performance on all datasets, while Method B slightly improves it for ChemProt and SST-2. We assume that the poor performance is related to the way we selected the best hyperparameters α and β for the final model. Based on the grid search results, we observe that quite frequently the overall highest score is too localized on a specific selection of α and β, even though neighbouring points significantly worsen the performance. This is an indication that generalization over the test data is not likely. Furthermore, we notice that in many cases the alternative strategies we propose work better than our retrofitting approaches. This suggests that i) the use of the lexicons is meaningful, but ii) we have not yet found the correct way of exploiting this knowledge. It is also worth highlighting the abrupt decrease in dev and test performances on the i2b2-2010 dataset for class posterior averaging. After manually inspecting the output probability scores of the model, we sometimes observe that while initially the predictions for the original sentences are correct, many of the modified sentences favour the same class, which is different from the one predicted in the original sentence. Since averaging equally favours each class, the final prediction will then be wrong. This is also evident when we compare with the score obtained from weighted majority vote for low values of δ. In that case, every sentence gets a weight of ~ 1 ending up in standard majority voting. This indicates that the original sentence is more important for the classification on this dataset than the modified ones, implying in turn that we should assign it a higher weight. Furthermore, it suggests that many neighbouring words are not relevant.

3.3.2. Neighborhood based hyperparameter selection

Since performance may be too localized on a single hyperparameter value, a slightly intuitive way is to examine the behaviour over a neighborhood of points instead. More specifically, in the whole $[0, 2] \times [0, 2]$ grid, we consider a square window of width 2, 4 and 6 respectively. For example, for a width of 4, the neighborhood contains $5 \times 5 = 25$ (α, β)-pairs for approach A and 5 α-points for approach B. We then move around this window to cover the whole grid area, while we average across all F_1-scores within the neighborhood to obtain a single estimate for all settings on the validation data. Finally,

Table 3.2.: Performance achieved by Methods A and B and some retrofitting approaches for static word embeddings on all datasets. Baseline corresponds to BERT base finetuned on each dataset for the specific task. Method A, B denote the proposed retrofitting approaches. Topline, class posterior averaging and weighted majority vote were discussed in Section 3.2.6, where for the last we select the weight (δ) based on the best performance on the validation set.

Corpus	**Model**	**Semantic lexicon**	**Dev set** miF_1/Acc (%)	**Test set** miF_1/Acc (%)
ChemProt	Baseline	—	74.47	72.61
	Method A	expanded-16	74.86	72.56
	Method B	annotated-50	74.59	72.63
	Topline	annotated-50	75.54	73.67
	CPA	annotated-50	72.92	72.07
	WMV ($\delta = 1.0$)	annotated-50	74.47	72.61
	Chiu et al. (2019a)	expanded-34	—	71.00
DDI	Baseline	—	71.34	80.11
	Method A	expanded-34	79.35	78.78
	Method B	annotated-34	72.33	79.43
	Topline	annotated-34	73.04	80.97
	CPA	annotated-34	71.97	79.40
	WMV ($\delta = 0.1$)	annotated-34	72.02	79.60
i2b2-2010	Baseline	—	71.34	72.69
	Method A	expanded-16	72.92	72.52
	Method B	annotated-34	71.83	72.63
	Topline	annotated-34	73.71	74.18
	CPA	annotated-34	60.79	58.50
	WMV ($\delta = 1.0$)	annotated-34	71.34	72.69
SST-2	Baseline	—	91.86	92.00
	Method B	$\text{WordNet}_{\text{syn}}$	92.09	92.11
	Topline	$\text{WordNet}_{\text{syn}}$	94.95	94.55
	CPA	$\text{WordNet}_{\text{syn}}$	90.37	90.11
	WMV ($\delta = 1.0$)	$\text{WordNet}_{\text{syn}}$	91.86	92.00
	Faruqui et al. (2015)	$\text{WordNet}_{\text{syn}}$	—	82.40

we select the center of the sliding window that yielded the maximum overall score, and evaluate the model on the test data. The results are displayed in Table 3.3.

Overall, we notice that our hyperparameter heuristic selection can sometimes improve the performance compared to the baseline, but it can also degrade it. More specifically, for every selection of width, Method A results in a better performance on ChemProt, whereas Method B degrades it. Furthermore, both approaches perform poorer than the

baseline on DDI, while for i2b2-2010, the results are contrasted. It is worth mentioning that our heuristic slightly improves over the original scores for Method A in Table 3.2, across all datasets. This also holds true for Method B on DDI and i2b2-2010 datasets, but not on ChemProt.

Ideally, we would hope for a single choice of parameters (hyperparameters α and β, lexicon, before or after layer normalization and Transformer block) that would generalize on any dataset. To test if such setting exists, we repeat the previous procedure for Method A, but we additionally average, each time, the scores obtained from each distinct setting of retrofitting parameters (lexicon, layer, etc.) across all datasets, to pick a single (α, β)-pair. We do not conduct the same experiment for Method B, since we only use annotated-34 and annotated-54 clusters as semantic lexicons. Taking a look at those results – marked as *Global* in Table 3.2 – we can observe that the performance consistently improves for ChemProt, however it degrades for all other datasets compared to the baseline.

3.3.3. Euclidean distance ranking of retrofitted vectors

In order to deeper understand how our proposed methods change the embeddings in space, let us focus on a single test case: Method A on ChemProt, using the expanded-16 clusters, and retrofitting after layer normalization at Transformer block 12, with $\alpha = 0.4$ and $\beta = 1.4$ (second row of Table 3.2) — where the proportion of disagreements between the baseline model and the test case model is statistically significant based on McNemar's test. This statistical test is particularly useful in the presence of limited amount of data, and it is suited for large deep learning networks whose training is time consuming (Dietterich, 1998).

Looking at the contingency Table 3.4, we observe that the proportion of errors for the baseline and the proposed approach are different. In particular, the baseline model made 134 predictions that the learned model got wrong, and similarly 85 correctly predicted samples from the learned model were wrongly classified by the baseline model.

This points out that both models behave differently, but on average they result in similar performances. To further analyse how Method A affects the embeddings in the latent space, we randomly select $5,000$ (out of $18,014$) test sentences where we apply our method, and we compute the corresponding activation of the `[CLS]` token before and after retrofitting. Next, we compute the Euclidean distance between every retrofitted vector and every `[CLS]` vector before retrofitting. This results in a $5,000 \times 5,000$ matrix, where each row contains the distances of one retrofitted vector to all original vectors (before retrofitting). We then rank from 1–5,000 each retrofitted embedding by sorting each row in the matrix in ascending order. By doing so, we can check how far our method is moving the embeddings in the latent space. The distribution of the resulting rankings across all vectors is summarized in Figure 3.3.

In this plot, we observe that a large proportion of vectors has a relatively low ranking (around $[0, 80]$), but there is also a considerable amount of vectors with high ranking (around $[950, 1000]$), suggesting that potentially the vectors do not move as far as they should, or sometimes they move too far. We can take a closer look by visualizing the

Table 3.3.: Performance of the model for both retrofitting approaches on each dataset after applying our heuristic hyperparameter selection with a neighbourhood width of 6. Global corresponds to the maximum average (α, β)-pair across all datasets for Method A.

Corpus	**Model**	**Semantic lexicon**	**Neighbourhood width**	**Dev set** *Acc* (%)	**Test set** *Acc* (%)
ChemProt	Baseline	—	—	74.47	72.61
	Method A	annotated-16	2	74.69	72.66
	Method A	annotated-16	4	74.69	72.63
	Method A	annotated-16	6	74.64	72.63
	Method B	annotated-50	2	74.57	72.59
	Method B	annotated-50	4	74.58	72.54
	Method B	annotated-50	6	74.58	72.60
	Global	annotated-34	2	74.47	72.77
	Global	annotated-34	4	74.55	72.99
	Global	annotated-34	6	74.64	73.00
DDI	Baseline	—	—	71.34	80.11
	Method A	annotated-16	2	79.16	79.18
	Method A	annotated-16	4	79.16	79.27
	Method A	expanded-34	6	78.45	78.79
	Method B	annotated-34	2	72.21	79.47
	Method B	annotated-34	4	72.21	79.47
	Method B	annotated-34	6	72.12	79.79
	Global	annotated-34	2	75.81	75.96
	Global	annotated-34	4	71.10	79.29
	Global	annotated-34	6	71.27	79.64
i2b2-2010	Baseline	—	—	71.34	72.69
	Method A	expanded-16	2	72.42	72.80
	Method A	expanded-16	4	72.18	72.74
	Method A	expanded-16	6	72.31	72.65
	Method B	annotated-34	2	71.64	72.64
	Method B	annotated-34	4	71.54	72.73
	Method B	annotated-34	6	71.54	72.67
	Global	annotated-34	2	71.85	71.48
	Global	annotated-34	4	71.54	72.46
	Global	annotated-34	6	71.43	72.60

embeddings (before/after retrofitting) and focusing on a single update. For the test case model parameters, we visualize in Figure 3.4 all vectors before and after retrofitting as well as the respective neighbouring embeddings and their mean using t-SNE (van der

Table 3.4.: Contingency table of the predictions of the baseline and the test case model on the ChemProt test data.

	Test case Correct	**Test case** Wrong
Baseline Correct	15,095	134
Baseline Wrong	85	1,629

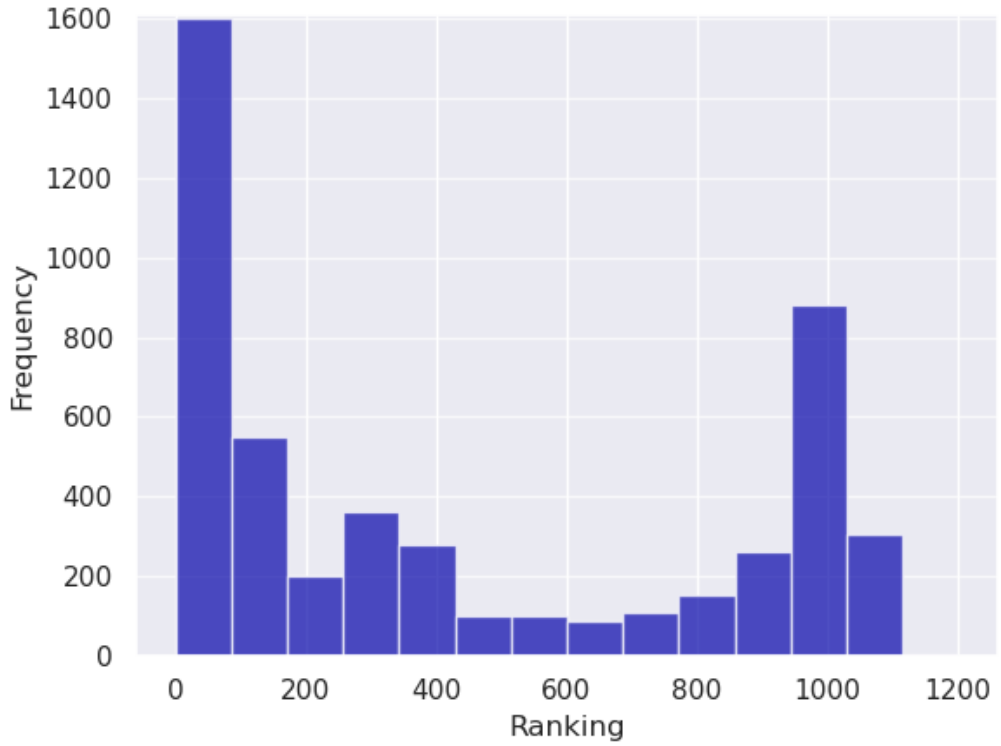


Figure 3.3.: Histogram of the ranking across `[CLS]` token retrofitted vectors for all 5,000 ChemProt test sentences where Method A is applied.

Maaten and Hinton, 2008), for 30 sentences from the development set. More specifically, the highlighted vectors correspond to the verb "study" with the initial activation of the `[CLS]` token before retrofitting q_i (red cross), its 897 neighbouring embeddings $\hat{q}_{jk}$ (blue points) and their mean $\mu_{\hat{q}_j}$ (yellow cross), and the retrofitted vector q_i (green point). According to Method A's update rule in Equation (3.2), since $\alpha = 0.4$ and $\beta = 1.4$, we expect q_i to end up closer to $\bar{q}_i$ than $\mu_{\hat{q}_j}$ which is the case. However, it is clear that $\hat{q}_{jk}$

are too many in size, and distributed in several places across the latent space. This is an indication that there is a lot of variation in the neighbouring embeddings, and therefore not all words in the lexicon are relevant for the task at hand.

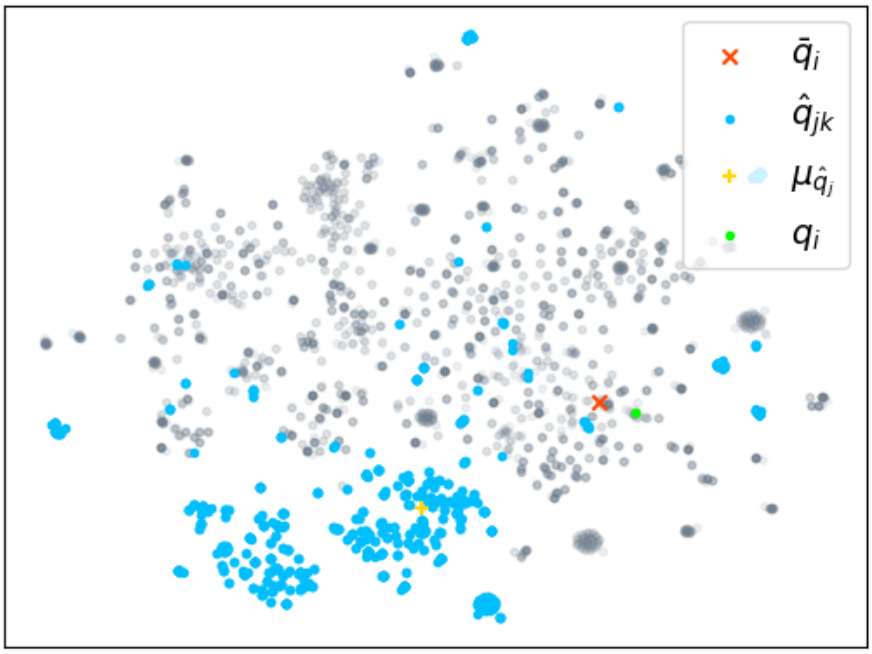


Figure 3.4.: T-SNE visualization for Method A while retrofitting after normalization at Transformer block 12 using the expanded-16 clusters on ChemProt.

3.3.4. Neighbouring word filtering

Bing Liu's list of adjectives allows us to focus on appropriate words in the semantic lexicons for the task of sentiment analysis. The next question we want to answer is: which neighbouring words are relevant for the underlying word, and which are not? It is obvious that not all neighbouring words for a given word in the lexicons are actual synonyms in the context of movie reviews. Replacing single words in the input sentence in Method B forces the same context between the original and the modified sentence. Consequently, we restrict the lexicons to the domain by selecting neighbours that are "good" replacements instead of using the whole list. This is done by inspecting the predictions of BERT for every original and modified sentence on the augmented development set for a given lexicon (see Section 3.2.6). Then, we can distinguish between the following cases:

Table 3.5.: Results for the best performing lexicons derived from our neighbouring word selection strategy for Method B and the proposed alternative strategies. Baseline corresponds to BERT base finetuned on SST-2 for sentiment analysis.

Lexicon	**Model**	**Dev** *Acc* (%)	**Test** *Acc* (%)
—	Baseline	91.86	92.00
$\text{FrameNet}_{10\%}$	Method B	92.09	92.00
	Topline	92.09	92.11
	CPA	92.09	92.00
	WMV ($\delta = 0$)	92.09	92.00
$\text{WordNet}_{syn_{10\%}}$	Method B	92.09	92.00
	Topline	92.66	92.00
	CPA	92.09	91.89
	WMV ($\delta = 0$)	92.09	92.00

A) the original sentence was wrongly classified but the modified sentence was correctly classified (good case),
B) the original and the modified sentence were both correctly or wrongly classified (neutral case),
C) the original sentence was correctly classified but the modified sentence was wrongly classified (bad case).

Next, we compute the counts that correspond to good, neutral and bad cases for every pair of original-neighbouring words. These show on average whether a neighbour is a good replacement or not for a given word. Then, using McNemar's statistical test, we create three reduced versions, one for each semantic lexicon, by selecting a neighbour for a given word with a 10%, 50% and 90% confidence level[3]. The higher the confidence level the more certain we are about replacing a word by another one, but the smaller the lexicon becomes (and vice versa). Finally, we repeat the grid search optimization (see Section 3.2.5) and present in Table 3.5 the results for the best settings.

Overall, there is some gain in performance compared to the baseline on the development set as expected. For example, Method B reaches topline performance for $\text{FrameNet}_{10\%}$, which suggests that retrofitting in the sense of averaging embeddings can be meaningful. Moreover, we can see that the topline performance is almost identical to that of the baseline model on the test data. This is due to the limited size of the reduced lexicons. For example, FrameNet originally consists of $1,700$ words and $90,140$ relations, while its largest reduced version, $\text{FrameNet}_{10\%}$, has only 1 word and 5 relations, as shown analytically in Table 3.6. Ideally, if the dataset were bigger, we would have selected lexicons with higher confidence level that would also be large enough to improve over the baseline, i.e., the topline score would significantly outperform the baseline.

[3]We use the confidence level percentage as a subscript to denote the reduced lexicon, e.g., $\text{FrameNet}_{90\%}$.

Table 3.6.: Approximate size of the graphs obtained from the original four lexicons intersected with Bing Liu's sentiment lexicon (top left) along with their 10%, 50% and 90% reduced versions we created.

Lexicon	**# Words**	**# Edges**
FrameNet	1,700	90,140
PPDB	4,893	44,829
$\text{WordNet}_{\text{syn}}$	5,481	29,848
$\text{WordNet}_{\text{all}}$	5,481	113,792
$\text{FrameNet}_{10\%}$	1	5
$\text{PPDB}_{10\%}$	1	6
$\text{WordNet}_{syn_{10\%}}$	4	6
$\text{WordNet}_{all_{10\%}}$	6	9
$\text{FrameNet}_{50\%}$	—	—
$\text{PPDB}_{50\%}$	1	1
$\text{WordNet}_{syn_{50\%}}$	2	2
$\text{WordNet}_{all_{50\%}}$	1	1
$\text{FrameNet}_{90\%}$	—	—
$\text{PPDB}_{90\%}$	—	—
$\text{WordNet}_{syn_{90\%}}$	1	1
$\text{WordNet}_{all_{90\%}}$	—	—

3.3.5. How does averaging compare to majority voting?

A simple way of assessing whether averaging is a good way of combining the knowledge coming from the semantic lexicons is to analyze if it behaves similarly, better, or worse, than majority voting. To answer this question, we count how many times Method B yields the correct answer, when the predictions of the modified sentences are 0–10% correct, up to 90–100% correct. For example, we can see the distribution of these counts for our best performing lexicons on the development set in Figure 3.5. For both systems, Method B gives exactly the same answers as majority voting, which in their largest portion fall into the 90–100% interval of the correct answers. Therefore, averaging preserves the majority vote so there is hope in retrofitting provided the lexicon can help.

3.3.6. Further remarks

Impact of the hyperparameters (α, β). The neighbourhood based selection experiments discussed in Section 3.3.2, are contrasted across datasets, meaning that this strategy can sometimes help but other times be harmful as well. This leads to the conclusion that the choice of the hyperparameters α, β, is dependent on each dataset. Therefore, we cannot find a single pair that globally results in a good performance, as in the original retrofitting Equation (3.1), where α and β are fixed.

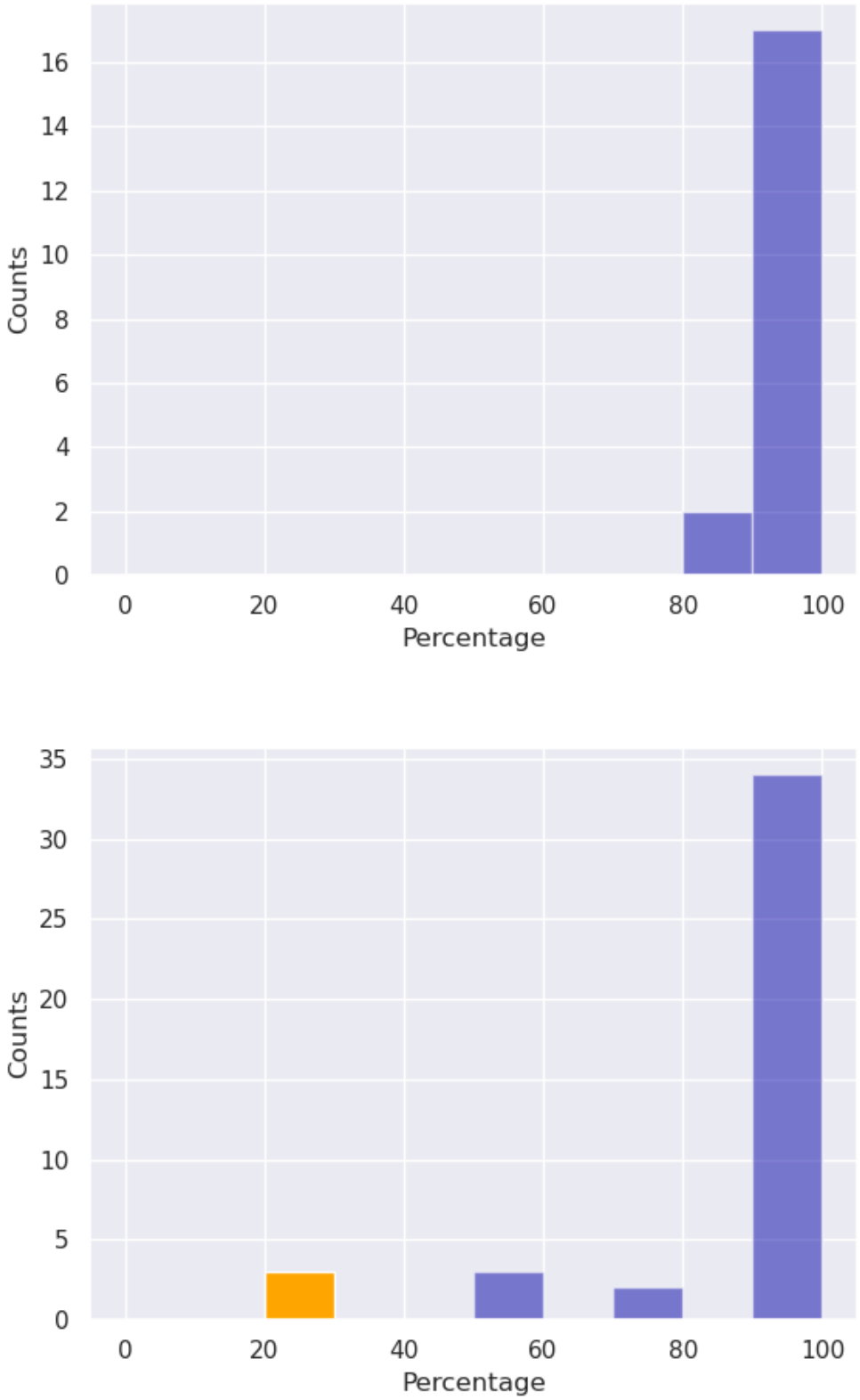


Figure 3.5.: Histogram of the cases where Method B yields the correct (blue) and incorrect (orange) answer compared to the percentage of correctly classified modified sentences for Framenet$_{10\%}$ (top) and WordNet$_{syn_{10\%}}$ (bottom). Top bins (0.5 – 1.0) indicate when majority voting succeeds while lower bins (0 – 0.5) show when it fails.

Impact of BERT parameters. Based on the results obtained from the grid search for Method A, the behaviour of the system with respect to the choice of retrofitting before or after layer normalization in the output layer of BERT, is different for each biomedical dataset. However, we could consistently observe that after layer normalization at the last Transformer block, the majority of (α, β)-pairs lead to bad performances on the validation data across all datasets. Another interesting point is that we come across multiple best scores when we retrofit before layer normalization at Transformer block 11, which is the farthest position from the output layer. As we move closer to the output layer, performance starts to be more localized around individual hyperparameters α and β. The same behaviour is observed for Method B. Illustrations of such cases can be found in Appendix A.1.

Annotated vs. expanded clusters. The results from grid search experiments show, in most cases, a similar distribution of F_1-scores if we compare lexicons with the same number of classes, e.g., annotated-16 vs. expanded-16, with the difference that the expanded comes with more points of worse performance. Consequently, and based on our neighborhood based hyperparameter selection, the annotated clusters are favoured. It is worth mentioning that the amount of verbs contained in the expanded clusters is considerably larger than the one in the annotated, resulting in a less precise knowledge overall. This may also be the reason why the performance drops for more (α, β)-pairs.

Alternative strategies. As expected, topline yields the best score across tasks, datasets, and lexicons, implying that the use of semantic lexicons can help in improving performance. In contrast, class posterior averaging is the worst among the alternative classification strategies, mainly because it equally favours the predictions derived from the original and the modified sentences. Based on the grid search results, a better alternative is weighted majority vote with a weight $\delta = 1$, which essentially means that the predictions on the modified sentences have less impact on the final outcome than the prediction on the original sentence.

3.4. Summary

In this chapter, we have proposed two approaches that extend the original retrofitting technique to operate with contextualized embedding systems. More precisely, the proposed technique incorporates external knowledge coming from semantic lexicons into BERT contextualized representations. After conducting a large-scale series of experiments on three biomedical datasets for relation extraction, and one movie review dataset for sentiment analysis, we observed that both approaches do not substantially affect the performance on these downstream tasks. Our test results show that the lexicons can be a useful source of information to further improve the results. However, the current experimental setting did not make it viable. This is demonstrated in our qualitative study, where we show that, when we improve the quality of the semantic lexicons by selecting only relevant neighbours for a given word, the resulting lexicons are not sufficiently large to be able to generalize at test time. Future work and perspectives are provided in Chapter 5.

4. An analogy based approach for solving target sense verification

Recent efforts have focused in understanding how semantic and syntactic knowledge could be encoded in large language models (LLMs) (Peters et al., 2018; Tenney et al., 2019; Liu et al., 2019a; Reif et al., 2019; Hernandez and Andreas, 2021). Despite their great success in a plethora of downstream tasks, the ability of these models to perform reasoning is limited and understudied (Talmor et al., 2020; Rogers et al., 2020). Designing LLMs that are able to reason over the knowledge they possess may not necessarily translate into better performance, however it is a prerequisite for interpretability (Zhou et al., 2020). The type of reasoning varies depending on the task at hand, e.g., answering chronological questions and summarizing events (temporal reasoning), selecting the most plausible explanation given a set of observations and hypotheses (abductive reasoning), understanding whether the meaning of a given text entails that of another (semantic inference), or finding common relations between pairs of words or phrases (analogical reasoning).

In this chapter, we tackle target sense verification (TSV) in terms of analogical reasoning by combining BERT with a CNN architecture that models analogical proportions by design and that is used for detecting analogies (Lim et al., 2019; Alsaidi et al., 2021a). Our experiments demonstrate that the position of the definition and hypernyms in the BERT input sequence, as well as the emphasis on the hypernyms, significantly impact the final performance. Moreover, we achieve competitive results on the WiC-TSV evaluation benchmark (Breit et al., 2021). Finally, promoting analogical reasoning by using the axiomatic properties of analogical proportions explicitly during training yields comparable performance and alleviates the dependence on the input encoding of BERT, which makes our model more robust.

The main contributions of this work are the following:

- we reformulate TSV as an analogy detection problem,
- we propose AB4TSV, a hybrid approach for solving TSV,
- we optimize the input encodings for AB4TSV,
- we demonstrate that AB4TSV achieves competitive performance on the WiC-TSV evaluation benchmark,
- we show empirically that enforcing the axiomatic properties of analogies during training yields a more robust model.

The chapter is organized as follows. The problem formulation is presented in Section 4.1. The model architecture is described in Section 4.2, and the experimental setup in Section 4.3. In Section 4.4 we analyze the results, and we discuss conclusions and perspectives for future work in Section 4.5.

4.1. Problem formulation

Analogical reasoning is one of the most used inference approaches in everyday life since it can be easily adapted to many common-sense applications involving reasoning: problem solving, modeling, planification, etc. Analogies also constitute a natural approach to modeling medical reasoning as practiced by physicians and medical staff. Further applications are found in natural language processing in tasks such as machine translation (Langlais et al., 2009), visual question-answering (Peyre et al., 2019), semantic (Lim et al., 2019) and morphological (Alsaidi et al., 2021a) problems.

Solving analogy-based problems requires the system to learn how to reason over relations of the form $A : B :: C : D$, which reads as "A *is to* B *as* C *is to* D". Following the axiomatization from Lepage (2004), a quaternary relation constitutes an analogical proportion if and only if the following properties hold true: $\forall A,\ B,\ C,\ D$,

1. $A : B :: C : D \Rightarrow C : D :: A : B$ (symmetry)
2. $A : B :: C : D \Rightarrow A : C :: B : D$ (central permutation)

Analogies combined with data-driven methods were proven to be beneficial in a variety of tasks, ranging from transfer learning to data augmentation and explainable AI (Keane and Smyth, 2020; Hüllermeier, 2020; Alsaidi et al., 2021b; Afantenos et al., 2021; Ushio et al., 2021a). However, to the best of our knowledge, they have not yet been applied to word sense disambiguation tasks.

Target sense verification (TSV) (Breit et al., 2021) is a word sense disambiguation task in which the system is provided with a target word in context on the one hand and a definition and a set of hypernyms of that word on the other hand, and it must decide whether their senses match or not. Consider for instance the context *"home is where the heart is"*, the definition *"where you live at a particular time"* and the set of hypernyms *"residence, abode"*, all corresponding to the target word *home*, as shown in Figure 4.1. To disambiguate the meaning of *home* in that specific context, one must compare and reason about the underlying relations between these concepts, e.g., infer that *home* in this context refers to an environment rather than a place as conveyed by the definition and the hypernyms.

Although TSV is not originally viewed as an analogy problem, analogical reasoning can be applied to solve it. To illustrate this, let us focus on two contexts — in the first one, the target word *home* matches the target sense given by the definition and the hypernyms, whereas in the second one it does not (Figure 4.1). Observe that the definition and the hypernyms always correspond to the same sense S_T (target sense). We denote this relation by R'. Solving TSV requires us to find whether the sense of *home* in context S_I (intended sense) corresponds to S_T. Let R be the relation that *home* points to S_I and hypernyms[1] point to S_T. We can now reformulate the problem in the form of analogical proportions such as *home : hypernyms :: definition : hypernyms*, and check whether it constitutes a valid analogy in each context or not (analogy detection). If R is approximately the same as R', then their senses match and we output True, otherwise we output False. This modification essentially allows us to reuse LLMs that

[1] Or the definition, but for the sake of this example we use the hypernyms.

are suitable for solving TSV, and at the same time it gives us access to existing tools for tackling analogies. This way we can test if the combination of both is beneficial in terms of performance on the task and/or in terms of robustness of the hybrid model. Furthermore, we focus on TSV because it is more complex than classical word analogy or lexical relation classification tasks (Ushio et al., 2021a). Indeed, TSV does not compare isolated words but words in context, lists of words and full sentences.

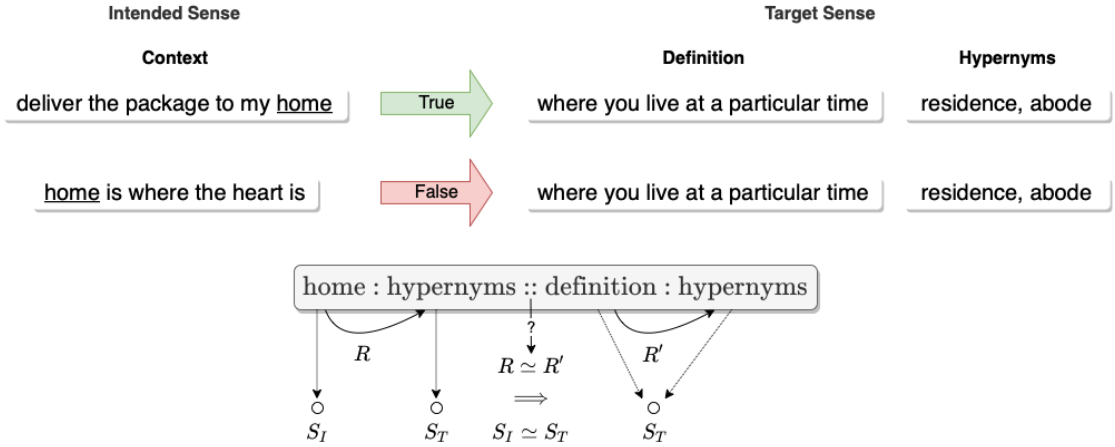


Figure 4.1.: Illustration of translating TSV into analogy detection.

4.2. AB4TSV architecture

Following Lim et al. (2019); Alsaidi et al. (2021a) we use a CNN classifier that explicitly models the relations *"is to"* and *"as"* in the analogy, as an interpretable alternative to the black-box classifier of Breit et al. (2021). The architecture of the model is composed of two main parts. First, we encode the words or sentences to be compared (target word, context, definition, hypernyms, etc.) into 4 embeddings **A**, **B**, **C**, **D** using BERT final encoder layer. Then we stack them into an $n \times 4$ matrix, where n is the embedding size. This matrix serves as input to a CNN classifier that consists of the following layers:

- A convolutional layer with 128 filters of size 1×2 with stride $(1, 2)$ with ReLU activation. The output of this operation is an $128 \times n \times 2$ matrix. Intuitively, this layer models "*is to*" in *"A is to B"* and *"C is to D"*.
- A convolutional layer with 64 filters of size 2×2 with stride $(2, 2)$ with ReLU activation. The output of this operation is flattened into a vector of length $64 \times (n-1)$. Intuitively, this layer models the relation "*as*" in *"A is to B as C is to D"*.
- A fully-connected layer with sigmoid activation, resulting in a scalar output.

The main difference with Lim et al. (2019) and Alsaidi et al. (2021a) is the shifting from static to contextualized embeddings. While they use pretrained GloVe vectors or train a character-based CNN to extract word representations, we utilize BERT to extract **A**, **B**, **C**, **D**. Together with our original input encoding optimization (Section 4.3.2), this

allows us to efficiently compare objects of different structure such as words, list of words, full sentences and/or their combinations, rather than just isolated words. The proposed *Analogy and BERT for target sense verification* (AB4TSV) architecture is depicted in Figure 4.2.

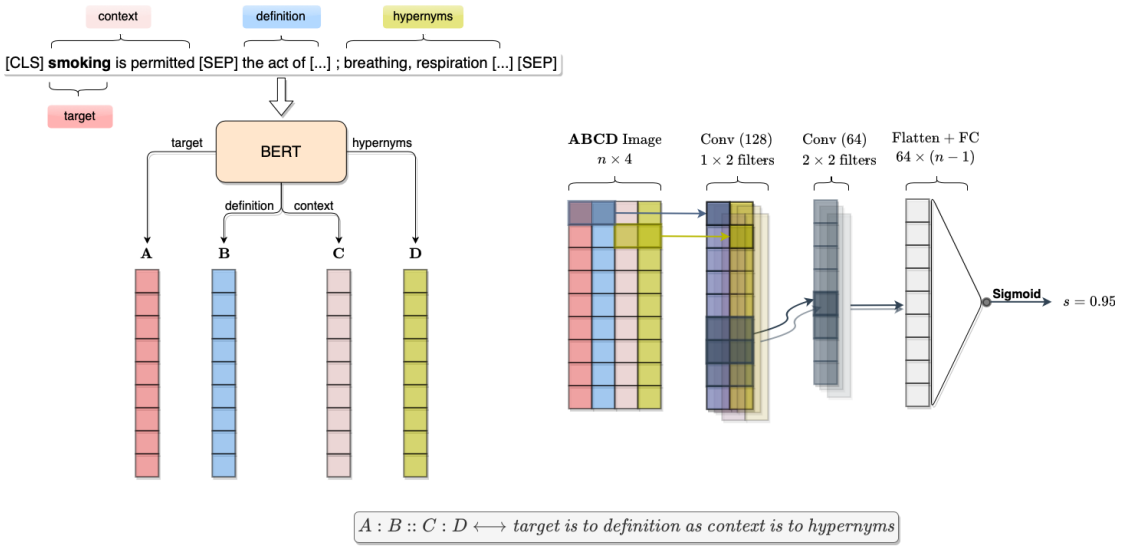


Figure 4.2.: Overview of AB4TSV. In this example we want to test whether $A : B :: C : D$ is a valid analogy when A = target, B = definition, C = context *and* D = hypernyms. The inputs are patched into a pair of sentences (context, definition-hypernyms) and fed into BERT. The embeddings of **A**, **B**, **C**, **D** are extracted from the last encoder layer of BERT and stacked into an $n \times 4$ matrix and input to the CNN classifier. Here, the classifier outputs *True* since the output value $s = 0.95 > \delta$, where the decision threshold δ is set to 0.5.

4.2.1. Choice of analogical relation

Tackling TSV based on analogical reasoning requires us to select A, B, C, $D \in S$ such that the relation $A : B :: C : D$ yields good classification performance. Let $S = \{cls, tgt, ctx, def, hyps, descr\}$ be the set of set of tokens that can be obtained from BERT as follows: cls corresponds to the `[CLS]` token, tgt denotes the target word in the context, ctx includes all tokens in the context, def represents all tokens in the definition, $hyps$ contains all tokens in the hypernyms, and $descr$ is the concatenation of all tokens in the definition and the hypernyms. There are numerous pooling strategies for obtaining a fixed representation for every set of tokens in S, such as summing the corresponding hidden states of all 12 Transformer blocks of BERT, selecting the hidden states of the second-to-last Transformer block, concatenating the hidden states of the last four Transformer blocks, and others (Devlin et al., 2019). Typically, the choice is related to the performance on the downstream task and is empirically justified (Devlin et al., 2019). However, exploring different pooling strategies is outside the scope of this work,

hence we stick to the standard selection for this task as proposed by Breit et al. (2021). Accordingly, we denote by **cls** the embedding of the `[CLS]` token, **tgt** the embedding of the target word, **ctx** the average of the embeddings of all words in the context, **def** the average of the embeddings of all words in the definition, **hyps** the average of the embeddings of all hypernyms, and **descr** the average of the embeddings of all words in the definition and all hypernyms, where each embedding is the corresponding hidden state of the last BERT Transformer block. The selection of $tgt, ctx, def, hyps$ as possible candidates for A, B, C, D is essential, since these are the main components that carry the senses to be compared according to the task. Additionally, the embedding of *cls* can generally be seen as a representation of the whole input, therefore it may capture key information both from context, and definition/hypernyms. Finally, inspired by the authors of WiC-TSV (Breit et al., 2021) we also test for *descr*, which essentially treats definition and hypernyms as a whole rather than separate units. As said, this selection of S allows us to compare relevant instances of different structure (special tokens, words in context, list of words, full sentences or their combinations), and thus, makes the task more challenging than typical word analogy or lexical relation classification.

4.2.2. Input encoding selection

The order in which the context, definition, and hypernyms are fed into BERT has a direct impact on the embeddings of the A, B, C, D candidates in S. The default input encoding format (Breit et al., 2021) structures the BERT input sequence into a pair of sentences, such that the context comes first, and the concatenation of the definition and the hypernyms in the second sentence, as illustrated at the top left of Figure 4.2. Preliminary experiments using the default input encoding format show that most relations seem to work except when *hyps* is included. This may be due to the way hypernyms are encoded: they are simply a set of words separated by commas, appended to the definition. BERT was originally trained on syntactically correct sentences, hence it might find such definition/hypernyms "sentence" difficult to interpret. Therefore, we introduce alternative ways of encoding the definition and the hypernyms into embeddings based on the following operations:

- **swap**: exchanging the position of the definition and the hypernyms in the default input encoding;
- **fc**: enclosing the hypernyms with focus characters, e.g., *residence, abode* becomes *`$` residence, abode `$`*;
- **em**: enclosing the hypernyms with entity markers, e.g., *residence, abode* becomes *`[H]` residence, abode `[/H]`*.

Following Breit et al. (2021) we always apply **fc** to the target word in the context. The use of focus characters/entity markers works as a form of weak supervision, pointing out important terms in the sentence (Huang et al., 2019; Baldini Soares et al., 2019; Zhang et al., 2019; Wang et al., 2021a; Zhou and Chen, 2022). This does not directly address the syntactic correctness issue caused by the hypernyms, however it instructs BERT to treat them in a special way. That is, all hypernyms will now share a common characteristic in the data that will, ideally, alter their embeddings in such a way that they become

more meaningful for solving the task. Moreover, as shown by Baldini Soares et al. (2019) part of the properties of employing such strategies can be captured in the embeddings of the focus characters/entity markers themselves. Therefore, we include them in the computation of **hyps**, as a means to explicitly transfer these properties directly to the representation of hypernyms.

Based on these three operations, we test the following 6 input encodings:

1. **default**:
   `[CLS] context [SEP] definition; hypernyms [SEP]`
2. **default+fc**:
   `[CLS] context [SEP] definition; $ hypernyms $ [SEP]`
3. **default+em**:
   `[CLS] context [SEP] definition; [H] hypernyms [/H] [SEP]`
4. **swap**:
   `[CLS] context [SEP] hypernyms; definition [SEP]`
5. **swap+fc**:
   `[CLS] context [SEP] $ hypernyms $; definition [SEP]`
6. **swap+em**:
   `[CLS] context [SEP] [H] hypernyms [/H]; definition [SEP]`

4.3. Experimental setup

In this section, we first present the data used to train and evaluate AB4TSV. Then, we describe the experimental procedure along with some technical details concerning implementation, evaluation and baselines for comparison.

4.3.1. Data

We use the WiC-TSV (Word in Context - Target Sense Verification) dataset, which was designed specifically for TSV (Breit et al., 2021). The data are pairs of context and definition-hypernyms, where the definition and hypernyms correspond to the same sense of the target word. General-domain instances are extracted from WordNet and Wiktionary (WNT/WKT). Domain-specific instances for Cocktails (CLT) and Medical Subjects (MSH) were taken from "All about cocktails"[2] and MeSH[3] thesauri respectively, while Computer Science (CPS) examples were manually constructed. The training and development sets include general-domain sentences only, while the test set includes domain-specific sentences too (see Tables 4.1 & 4.2). The task is divided into three sub-problems taking into account only the definition (sub-task 1), only the hypernyms (sub-task 2), or both (sub-task 3). In the following we focus on sub-task 3 only, since we are interested in comparing the underlying relations between combinations of instances of different structure, including that of the definition with the hypernyms.

[2] http://vocabulary.semantic-web.at/cocktails
[3] https://www.nlm.nih.gov/mesh/meshhome.html

Table 4.1.: Statistics of the WiC-TSV dataset. The $\mathcal{P}_+$ column refers to the proportion of positive examples.

		Total	$\mathcal{P}_+$
Train	**WNT/WKT**	2137	0.56
Dev	**WNT/WKT**	389	0.51
Test	**WNT/WKT**	717	0.54
	Domain-specific	589	0.47
	MSH	205	0.52
	CTL	216	0.43
	CPS	168	0.46
	All	1306	0.51

Table 4.2.: WiC-TSV samples taken from the development set. The target word in the context is highlighted in bold. Notice that in some examples the meaning of the target word can be easily disambiguated solely from the context, whereas in others the definition and/or hypernyms are necessary.

Context	Definition	Hypernyms	Label
A **marriage** of ideas.	A close and intimate union.	union, unification	True
A **fight** broke out at the hockey game.	The act of fighting; any contest or struggle.	conflict, struggle, battle	True
My neighbor was the lead **role** in last year's village play.	The actions and activities assigned to or required or expected of a person or group.	duty	False
They went bankrupt during the economic **crisis**.	A crucial stage or turning point in the course of something.	juncture, occasion	False

4.3.2. Analogical proportion optimization

Regarding the number of combinations of $A,\ B,\ C,\ D \in S$, there are $|S|^4 = 6^4 = 1,296$ possible relations in total. To ensure that we test whether the intended sense of the target word in the context corresponds to the target sense in the definition/hypernyms, we first distinguish between two sets: $S_1 = \{cls, tgt, ctx\}$ and $S_2 = \{cls, def, hyps, descr\}$ where $S_1 \cup S_2 = S$. The former set includes embeddings which primarily contain information coming from the context, while the latter involves embeddings which reflect the

information found in the definition-hypernyms. *cls* belongs to both since it represents the whole input. Next, we define the following rules: For all A, B, C, D in S,

1. $A \neq B \vee C \neq D$
2. $\neg\Big[\big[(A \in S_1 \setminus S_2) \wedge (B, C, D \in S_1)\big] \vee \big[(A \in S_2 \setminus S_1) \wedge (B, C, D \in S_2)\big]\Big]$

The first rule ensures that we avoid relations where embeddings on either side are identical, e.g., *A is to A as C is to D*. The second rule makes sure that each relation contains embeddings from both sources of information. In other words, A, B, C, D cannot be instantiated from S_1 or S_2 exclusively. These rules reduce the number of relations to be tested to 768. For each choice of input encoding and relation, we train our system 4 times using different random seeds, resulting in $6 \times 768 \times 4 = 18,432$ runs. A single run takes approximately 35 minutes on 1 Nvidia GTX 1080 Ti 11GB.

4.3.3. Assessing and promoting permutation invariance of analogical proportions

A key part of our experiments is to employ the permutation invariance properties of analogical proportions (see Section 4) to assess (i) whether analogical reasoning is beneficial in terms of performance on the task, and (ii) whether the model naturally learns to be invariant to these permutations or they must be explicitly enforced at training time. To assess whether the model is invariant, we compute the embeddings $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$ of the given relation to be tested, and simply compare the performance achieved when feeding the initial relation vs. the relations obtained by permuting analogical proportions to the classifier. To promote permutation invariance at training time, we include both the initial and the permuted relations in each minibatch. We distinguish training without/with permutation invariance by adding a subscript "pi", i.e., AB4TSV and AB4TSV$_{\text{pi}}$, respectively.

4.3.4. Technical details

We finetune our model for a maximum of 5 epochs using a binary cross-entropy loss. We use `bert-base` checkpoint weights, the Adam optimizer with a learning rate of 0.00005, a linear warmup scheduler, and gradient clipping with norm 1 using Hugging Face's Transformers library (Wolf et al., 2020).

As baselines we consider the BERT variants published by Breit et al. (2021), namely HyperBertCLS and HyperBert3. Both models are composed of BERT with a linear layer on top as a classifier. The key difference between them is that the former takes the embedding of the `[CLS]` token as input to the classifier, while the latter takes not only the embedding of `[CLS]` but also the embedding of the target word and the average of all words in the definition and all hypernyms.

Performance is measured in terms of accuracy and $F1$-score. Unless otherwise stated, all tables report statistics (mean and standard deviations) of these two measures over several independent runs (exact number stated in the text for each experiment). Since the test set labels of WiC-TSV are not publicly available, all reported scores on the test set are computed by the organizers (Breit et al., 2021).

The WiC-TSV data, scripts for tranining/evaluating AB4TSV and the baseline models, and source code for reproducing our experimental results are available at our GitHub repository[4].

4.4. Results

In this section we present the results obtained from the optimization of the input encoding and the analogical relation, compare those with the baselines on the development set, and with existing approaches on the test set. Next, we discuss the effects of utilizing (or not) the analogical properties explicitly during the training process.

4.4.1. Impact of the input encoding

Figure 4.3 shows the mean accuracy achieved across all 4 runs sorted in ascending order for each input encoding and each analogical relation (Section 4.3.2). Overall, for all input encodings there exist some A, B, C, D combinations that result in good performance. However, looking at the general trend of these curves, we can observe that some are more sensitive than others to the selection of the analogical relation. More specifically, **fc**/**em** outperform the **default**/**swap** encodings, showing that enclosing the hypernyms with focus characters or entity markers reduces the sensitivity to the selection of the A, B, C, D, and improves performance. In addition, entity markers appear to be consistently better than focus characters. One possible explanation is that the same focus characters are also enclosing the target word in the context. For example, in "`[CLS] ... $ tgt $ ... [SEP] def; $ hyps $ [SEP]`" part of the context and the definition are enclosed with `$`, while in "`[CLS] ... $ tgt $ ... [SEP] $ hyps $; def [SEP]`" part of the context is enclosed with `$`. Hence, the part of the input that comes after the target word and before the hypernyms is also enclosed in these special characters, possibly bringing some confusion to the model. Moreover, swapping or not the position of the definition and the hypernyms in the sentence (plain lines), results in a large amount of bad accuracies for several A, B, C, D combinations ($0 \sim 400$ on the x-axis). Taking a closer look at these particular combinations — highlighted in bold in Figure 4.3 — we observe that they all share a common characteristic. That is, at least one of the A, B, C, D in the analogical relation is instantiated as `hyps`. This demonstrates that BERT is struggling to make sense out of the hypernyms probably because of their structure — a list of words that comes before or after the definition, and does not form a syntactically correct sentence. However, this effect is not present when we apply the **fc**/**em** strategies (dashed and dotted lines).

[4] https://github.com/gonconist/ab4tsv

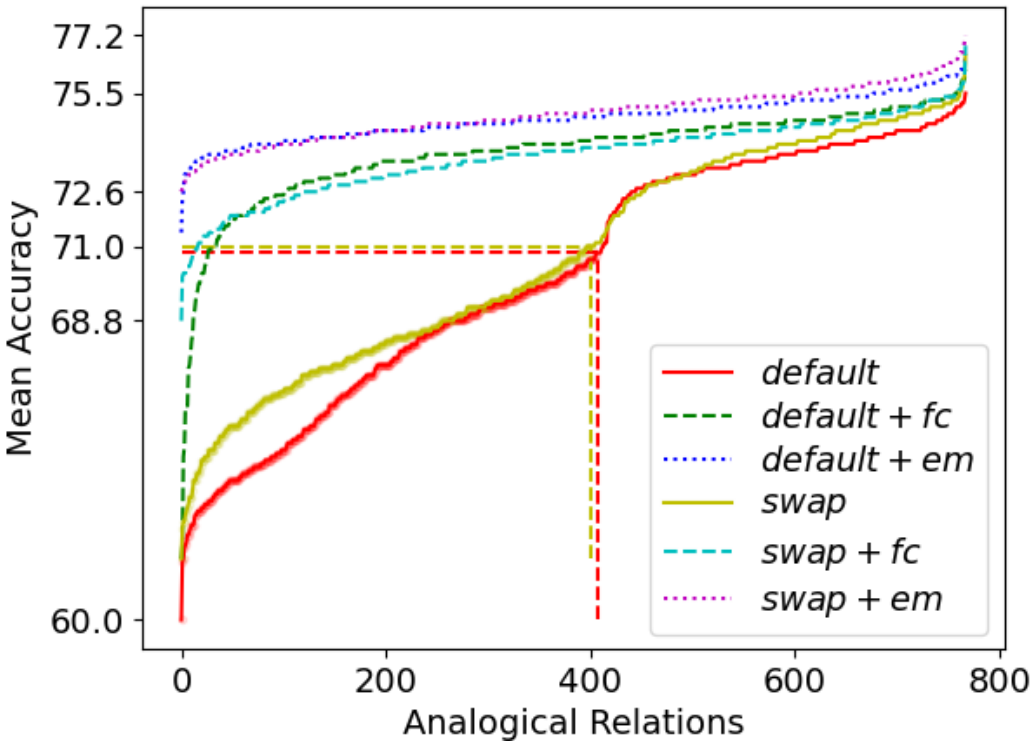


Figure 4.3.: Mean accuracy achieved on the development set. Each curve represents one possible input encoding, and the horizontal axis represents the 768 possible relations $A : B :: C : D$ sorted in order of increasing accuracy. The highlighted area shows the portion of analogical relations that result in bad performance for default and swap input encodings.

To further investigate this phenomenon we attempt to apply **fc**/**em** strategies only at the input level, without including the embeddings of the focus characters/entity markers in the averaging of **hyps**. We define these operations as ⟨**fc**⟩ and ⟨**em**⟩ respectively. Consequently, this results in the following 4 new input encodings:

1. **default**+⟨**fc**⟩:
   `[CLS] context [SEP] definition; $ hypernyms $ [SEP]`
2. **default**+⟨**em**⟩:
   `[CLS] context [SEP] definition; [H] hypernyms [/H] [SEP]`
3. **swap**+⟨**fc**⟩:
   `[CLS] context [SEP] $ hypernyms $; definition [SEP]`
4. **swap**+⟨**em**⟩:
   `[CLS] context [SEP] [H] hypernyms [/H]; definition [SEP]`

We relaunch the exact same experiment (Section 4.3.2) for these input encodings, and we plot the new curves in Figure 4.4. Based on the results, we notice that simply enclosing

the hypernyms with focus characters or entity markers at the input level is not sufficient to alleviate this problem. What really makes a difference, is the inclusion of the embeddings of these special characters when computing the vector representation of the hypernyms. Furthermore, ⟨**em**⟩ is superior to ⟨**fc**⟩ showing once again that the use of entity markers is more instructive for BERT on this particular task. Additional figures showing how the focus characters compare to entity markers are provided in Appendix B.1.

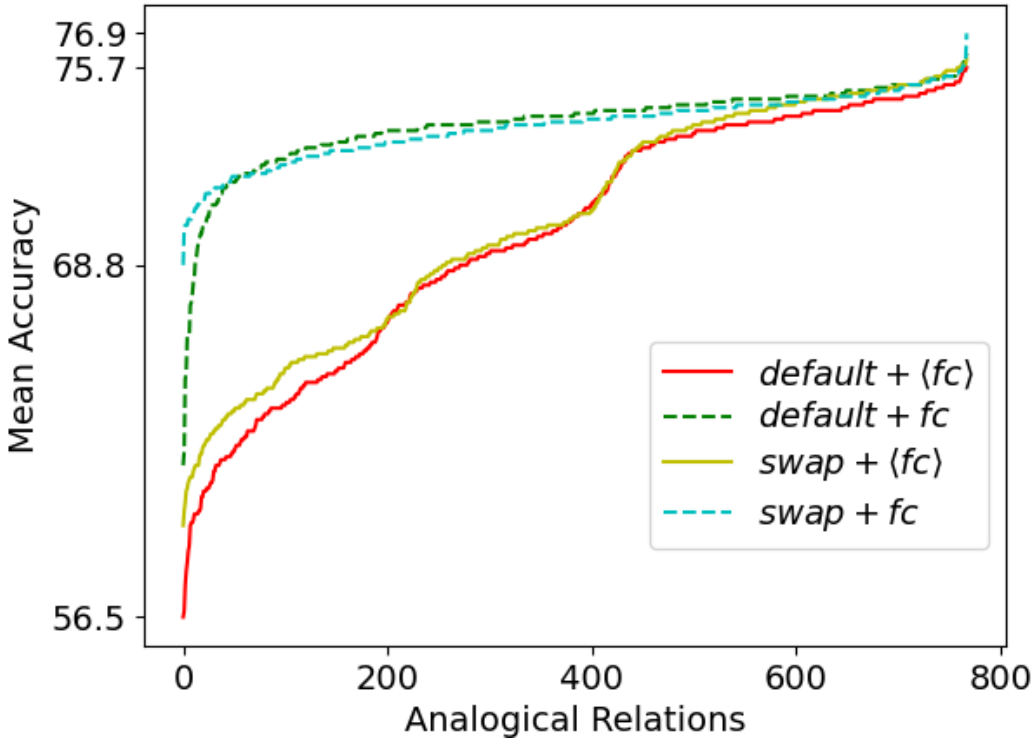


Figure 4.4.: Comparison between including the embeddings of the focus characters (**fc**) or not (⟨**fc**⟩) in the averaging of **hyps**. The vertical axis represents the mean accuracy achieved on the development set, and the horizontal axis represents the 768 possible relations $A : B :: C : D$ sorted in order of increasing accuracy.

Finally, one of the encoding strategies presented by Baldini Soares et al. (2019) is to represent the relation between two marked entities in the text, using the concatenation of their respective start entity marker token embeddings. Inspired by this, we introduce **em**† where we choose the embedding of the last entity marker, `[/H]`, is used as a representation for the hypernyms. Notice that the concatenation of both entity markers is not feasible in our case, as the CNN classifier expects a fixed length representation for each **A**, **B**, **C**, **D**. This modification results in 2 additional input encodings:

1. **default**+**em**†:

`[CLS] context [SEP] definition; [H] hypernyms [/H] [SEP]`

2. **swap**+**em**†:
   `[CLS] context [SEP] [H] hypernyms [/H]; definition [SEP]`

We repeat the same experiment and plot the results in Figure 4.5. Overall, this strategy works similarly to **em** which suggests that the last entity marker embedding alone contains enough information to help BERT disambiguate the hypernyms. Additional figures showing how **default**+**em**† compares to **swap**+**em**† are provided in Appendix B.1.

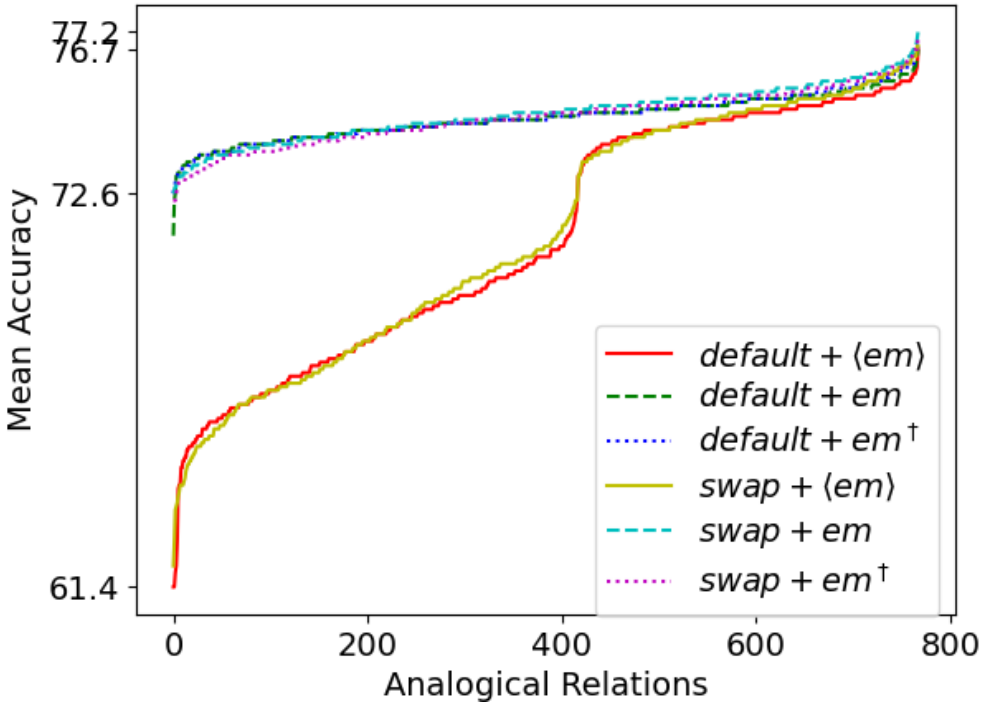


Figure 4.5.: Comparison between including the embeddings of the entity markers (**em**) or not (⟨**em**⟩) in the averaging of **hyps** or just using the embedding of the first entity marker (**em**†) to represent **hyps**. The vertical axis represents the mean accuracy achieved on the development set, and the horizontal axis represents the 768 possible relations $A : B :: C : D$ sorted in order of increasing accuracy.

4.4.2. Comparison with other methods for TSV

We retrain the AB4TSV models with the best relation for each selected encoding, and report the results over 10 runs, as well as those of the baselines, in Table 4.3. The results clearly show that AB4TSV outperforms the baseline models. This holds true even for one-to-one comparisons on the respective encodings, further demonstrating the importance of the input encoding for BERT. Specifically, exchanging the position of the definition and the hypernyms (**swap**) and using focus characters/entity markers (**fc**/**em**)

yields the best performance in terms of accuracy. Note also that `cls` is present in all analogical relations. This suggests that the `[CLS]` token is particularly important for solving this task.

Table 4.3.: Accuracy and $F1$-score achieved on the development set by the proposed method and the two baselines. The Analogy column shows the relation that yielded the best performance for a given encoding.

Encoding	**Analogy**	**Dev** *Acc*	**Dev** $F1$
default	$cls : descr :: cls : ctx$	74.5 ± 0.015	77.0 ± 0.016
default+fc	$cls : def :: ctx : cls$	74.9 ± 0.010	77.3 ± 0.006
default+em	$tgt : descr :: cls : def$	75.4 ± 0.027	**77.8** ± 0.023
swap	$def : cls :: cls : ctx$	75.4 ± 0.016	77.7 ± 0.016
swap+fc	$def : ctx :: cls : hyps$	**75.8** ± 0.013	77.7 ± 0.013
swap+em	$hyps : def :: cls : ctx$	**75.8** ± 0.017	77.7 ± 0.012
	Baselines		
default		74.4 ± 0.014	**77.2** ± 0.009
default+fc		73.5 ± 0.027	75.2 ± 0.035
default+em	HyperBertCLS	74.0 ± 0.022	76.1 ± 0.019
swap		72.6 ± 0.028	74.4 ± 0.031
swap+fc		73.1 ± 0.028	75.2 ± 0.031
swap+em		**74.6** ± 0.024	76.6 ± 0.022
default		74.0 ± 0.014	76.9 ± 0.007
default+fc		73.9 ± 0.018	76.3 ± 0.018
default+em	HyperBert3	73.1 ± 0.031	75.2 ± 0.032
swap		73.8 ± 0.015	76.3 ± 0.015
swap+fc		73.5 ± 0.011	75.6 ± 0.013
swap+em		74.4 ± 0.011	75.7 ± 0.024

Based on the results of the previous experiment we select the relation that performs best for each encoding, and train the AB4TSV model 10 times by promoting invariance to the permutations of analogical proportions at training time. The reported results in Table 4.4 show a slight decrease in performance compared to the systems trained without the permutations of analogical proportions in Table 4.3, in the order of 1% absolute. However, notice that after integrating analogical reasoning, the results are consistent across all encodings. This suggests that the model learns to be less dependent on the input encoding, and thus, is more robust.

Table 4.4.: Accuracy and $F1$-score achieved on the development set by promoting invariance to the permutations of analogical proportions at training time. The Analogy column shows the relation that yielded the best performance for a given encoding in the previous experiment.

Encoding	**Analogy**	**Dev** *Acc*	**Dev** $F1$
default	$cls : descr :: cls : ctx$	74.3 ± 0.016	76.1 ± 0.014
default+fc	$cls : def :: ctx : cls$	74.6 ± 0.008	76.6 ± 0.008
default+em	$tgt : descr :: cls : def$	**75.1** ± 0.014	**77.3** ± 0.013
swap	$def : cls :: cls : ctx$	74.2 ± 0.010	76.1 ± 0.011
swap+fc	$def : ctx :: cls : hyps$	74.8 ± 0.012	75.9 ± 0.024
swap+em	$hyps : def :: cls : ctx$	75.0 ± 0.009	76.4 ± 0.011

To assess the generalization capabilities of AB4TSV, we evaluate the performance of the best systems of the previous experiments on the test set. The results in Table 4.5 show that AB4TSV can outperform previously reported results on WiC-TSV both in terms of accuracy and $F1$-score, according to whether the axiomatic properties of analogies are enforced at training time or not. This illustrates the usefulness of analogical reasoning to solve the task, even on specific instances that lie outside the training domain. Interestingly, **swap+em** that was one of the most dominant strategies (see Figure 4.3) results in poorer performance compared to other approaches.

The WiC-TSV test set includes general-domain instances from WordNet and Wiktionary (WNT/WKT) as well as domain-specific examples about cocktails (CLT), medical subjects (MSH) and Computer Science (CPS). According to Breit et al. (2021) this data formation aims at testing the generalization capabilities of the underlying system on unseen instances and incomplete data, and evaluating transfer learning from general into specific domains. To get a better insight of how AB4TSV behaves on those instances, we report the detailed per-domain results for all AB4TSV configurations on the test set in Table 4.6. For clarity, we refer to AB4TSV using **swap+em** as input encoding and $hyps : def :: cls : ctx$ as the relation to be tested as system 1, AB4TSV using **swap+fc** as input encoding and $def : ctx :: cls : hyps$ as the relation to be tested as system 2, and AB4TSV$_{\text{pi}}$ using **default+em** as input encoding and $tgt : descr :: cls : def$ as the relation to be tested as system 3. Overall, system 1 is the least performing of the three and its decrease in performance is present both in general and domain-specific test examples. This could possibly mean that the particular selection of entity markers for the hypernyms does not generalize well on the test set. However, notice that system 3 using the same entity markers outperforms existing approaches, demonstrating once more that when trained using the axiomatic properties of analogies, AB4TSV becomes less dependent on the input encoding selection, and thus more robust. In addition, system 2 achieves the highest total $F1$-score with system 3 being really close. The main difference is that system 2 scores better on general domain instances, while system 3 is more accurate on CTL. Interestingly, all systems exhibit similar performance on CPS.

Table 4.5.: Test set results of our best performing systems trained with and without the permutations of analogical proportions, compared to previously reported results. All results are calculated by the authors of WiC-TSV benchmark (Breit et al., 2021).

Approach	**Test** *Acc*	**Test** *F*1
Supervised		
CTLR (Moreno et al., 2021)	78.3	78.5
Vandenbussche et al. (2021)	71.9	76.2
BERT-B (Breit et al., 2021)	76.6	78.2
BERT-L (Breit et al., 2021)	76.3	77.8
FastText (Breit et al., 2021)	53.4	63.4
AB4TSV+swap+em	75.7	77.5
AB4TSV+swap+fc	**78.6**	**79.8**
AB4TSV$_{pi}$+default+em	**78.6**	79.4
Unsupervised		
U-dBERT (Breit et al., 2021)	61.2	51.3
U-BERT (Breit et al., 2021)	60.5	51.9
MIRRORWIC (Liu et al., 2021)	73.7	—

Table 4.6.: Per domain performance of our AB4TSV models on the test set. Results are reported in terms of accuracy, precision, recall and *F*1-score, and are calculated by the authors of WiC-TSV benchmark (Breit et al., 2021).

	Test *Acc*	**Test** *Pre*	**Test** *Rec*	**Test** *F*1
AB4TSV+swap+em				
WNT/WKT	72.7	75.5	72.8	74.1
MSH	73.7	67.3	95.3	78.9
CTL	81.0	70.3	96.8	81.4
CPS	84.5	77.1	94.9	85.1
Total	75.7	73.2	82.4	77.5
AB4TSV+swap+fc				
WNT/WKT	75.0	78.1	74.6	76.3
MSH	79.5	73.5	94.3	82.6
CTL	84.7	76.3	93.6	84.1
CPS	85.1	77.3	96.2	85.7
Total	78.6	76.8	83.0	79.8
AB4TSV$_{pi}$+default+em				
WNT/WKT	74.5	77.4	74.4	75.8
MSH	79.5	74.6	91.5	82.2
CTL	87.0	84.2	86.0	85.1
CPS	84.5	77.1	94.9	85.1
Total	78.6	77.8	81.2	79.4

4.4.3. Invariance to the permutations of analogical proportions

In order to measure the invariance of the model w.r.t. permutations, we focus on the relation that yields the best accuracy on the analogical proportion optimization experiment (5^{th} row of Table 4.3). In this case, we compare the performance obtained using the initial and the permuted analogical relations as input to the classifier, depending on whether we explicitly promote invariance or not. Table 4.7 reports the results over 4 runs. As expected, the model is less dependent on both permutations when we explicitly enforce them during training. Conversely, when we train on a single non-permuted relation, the performance for symmetry greatly degrades at test time, while that of central permutation decreases by a smaller margin. This makes sense for the specific relation ($def : ctx :: cls : hyps$) since central permutation ($def : cls :: ctx : hyps$) resembles more the original than symmetry ($ctx : hyps :: def : cls$).

Table 4.7.: Results on the development set for the **swap+fc** encoding and the relation $def : ctx :: cls : hyps$. Permute column refers to promoting (✓) or not (✗) invariance to the permutations of analogical proportions at training time.

Property	**Permute**	**Dev** *Acc*	**Dev** $F1$
base	✗	**76.2** ± 1.927	**78.0** ± 1.932
	✓	75.1 ± 1.611	76.8 ± 1.891
sym	✗	53.2 ± 17.10	61.4 ± 18.53
	✓	74.5 ± 1.949	76.4 ± 2.210
cp	✗	72.9 ± 3.596	73.4 ± 5.760
	✓	74.7 ± 2.104	76.5 ± 2.315

4.4.4. Interpreting AB4TSV via explanation methods

For our last experiment we aim to get an intuition of how AB4TSV is making sense of the different parts of the input text to solve TSV. Specifically, we employ a set of feature attribution methods using the Captum library (Kokhlikyan et al., 2019) to measure the *importance* of each token in the input sequence, and in all 12 Transformer blocks of BERT. For the former, we utilize Input X Gradient (Shrikumar et al., 2017) an extension of Saliency (Simonyan et al., 2014), that computes input attribution based on the gradients of the output w.r.t. the input multiplied by the input feature values. Intuitively, given a linear model the gradients represent the coefficients of each input, and the input × coefficient product gives the total contribution of the feature w.r.t. the output of the linear model. For the latter, we utilize Layer Gradient X Activation — which is the equivalent of the Input X Gradient for hidden layers — that performs element-wise multiplication of the activation of the layer with the gradients of the output w.r.t. the given layer. In both cases, the input is the final representation for each token in the input sequence of BERT, i.e., the element-wise sum of the token, segment and position

embeddings (see Section 2.1.3.1), while the output corresponds to the unnormalized prediction score (logit) of the model for that input sequence.

We visualize the token attributions obtained from Input X Gradient in a saliency map, and attributions across all tokens and Transformer blocks obtained from Layer Gradient X Activation in a heat map, as shown in the example in Figure 4.6. In the top left corner, the saliency map provides the true label, the predicted label of the model along with the logit, the attribution label, and the attribution score. The explanation methods answer the question of how important each input token is towards the model's prediction. Therefore, the attribution label sets the target class for which we want to compute attributions based on the model's prediction. In this case it is set to `None`, since for TSV the model always outputs a scalar value per example. If instead the output of the model was binary, i.e., two confidence scores for predicting True or False, respectively, then we would have to specify the attribution label for which we would like to compute attributions, i.e., the one corresponding to the True or the False class. The attribution score corresponds to the normalized sum of all attributions of all tokens in the input. The colours of each token in the saliency map are associated with the importance of each token, i.e., red/white/green means that they are unimportant/neutral/important for that specific prediction, or in other words they are negatively/not/positively correlated with the predicted score. The meaning of importance can be interpreted as how much each token is contributing/steering the model towards producing this particular score. The heat map summarizes the attributions of all tokens across all 12 Transformer blocks of BERT. Higher colour intensity means higher importance, while darker means the opposite.

As our test models we choose HyperBertCLS and AB4TSV with $def : ctx :: cls : hyps$ as analogical relation and **swap+fc** as input encoding. After manually inspecting the WiC-TSV development set, we select a few examples for which we apply the explanation methods. For the sake of clarity, we display the results for the examples presented in Table 4.8, while we list the rest in Appendix B.2. To facilitate the interpretation, we focus on examples where the context clearly assists in the disambiguation of the target word, since otherwise the surrounding words do not contribute in understanding the sense of the target word. Furthermore, we restrict ourselves to examples where HyperBertCLS correctly predicts the sense for the target word while AB4TSV does not, and vice versa.

Table 4.8.: Samples from the WiC-TSV development set for which we apply the explanation methods. The target word in the context is highlighted in bold.

Context	**Definition**	**Hypernyms**	**Label**
They went bankrupt during the economic **crisis**.	A crucial stage or turning point in the course of something.	juncture, occasion	False
My neighbor was the lead **role** in last year's village play.	The actions and activities assigned to or required or expected of a person or group.	duty	False

In the first example of Table 4.8, HyperBertCLS correctly predicts that the sense are unrelated, while AB4TSV fails to verify them. The saliency map in Figure 4.6 illustrates that HyperBertCLS relies a lot on the hypernym `occasion` to make the decision. Moreover, the heat map below shows that the `[CLS]` token gains high attribution in early Transformer blocks but it diminishes in the last block. In general, most activity is concentrated in the hypernyms, and a few tokens from the context such as the target word `crisis` and the token `during`. In contrast, the saliency map in Figure 4.7 shows that AB4TSV puts a lot of emphasis in token `bankrupt` and sub-token `##cture` of the hypernym `juncture`. Furthermore, the heat map again reveals high attribution for the `[CLS]` token that gets eliminated in the last Transformer block. Moreover the tokens `bankrupt`, `during` and the sub-token `jun` have high contribution throughout various Transformer blocks. In both cases, the hypernyms seem to be more dominant for making the decision followed by a few tokens in the context, while the definition seems not to be so useful.

In the second example of Table 4.8, HyperBertCLS falsely verifies the senses, while AB4TSV succeeds in distinguishing them. The saliency map in Figure 4.8 shows that HyperBertCLS is focusing on seemingly irrelevant tokens for making the prediction, such as the separation tokens `[SEP]`. In addition, the heat map does not reveal a lot of high activity across the tokens and the Transformer blocks, possibly meaning that the model is having trouble at reasoning on that particular example. Looking at the equivalent picture for AB4TSV in Figure 4.9, we clearly see that the model is heavily relying on more useful tokens to disambiguate the sense of `role` in the context, such as the hypernym `duty`, and the tokens `village`, `play`. Regarding the attributions inside the BERT-part of the model, we observe that `duty` gains high attribution in the first four Transformer blocks that gets reduced as we proceed to later blocks. Therefore, we hypothesize that the hypernym `duty` is what guides AB4TSV to correctly verify the senses on that particular example.

Overall, and in the absence of recognized general explainability quality metric, it is not an easy task to arrive in conclusive findings concerning the interpretation of BERT-like models, as they are highly complex with many parameters that affect the final prediction. Nonetheless, the explanation methods can provide some useful insights. More specifically, based on the heat maps we notice that the `[CLS]` token always gets assigned with high or low attributions throughout the Transformer blocks, especially in the last, whereas the rest of the tokens in the last block get zero attribution. This makes sense particularly for HyperBertCLS as it was finetuned using the `[CLS]` token embedding of the last Transformer block as input to a linear classifier. This pattern is also true for AB4TSV, with the important difference that the respective objects in the analogical relation, i.e., $def : ctx :: cls : hyps$ also get assigned with non-zero attribution.

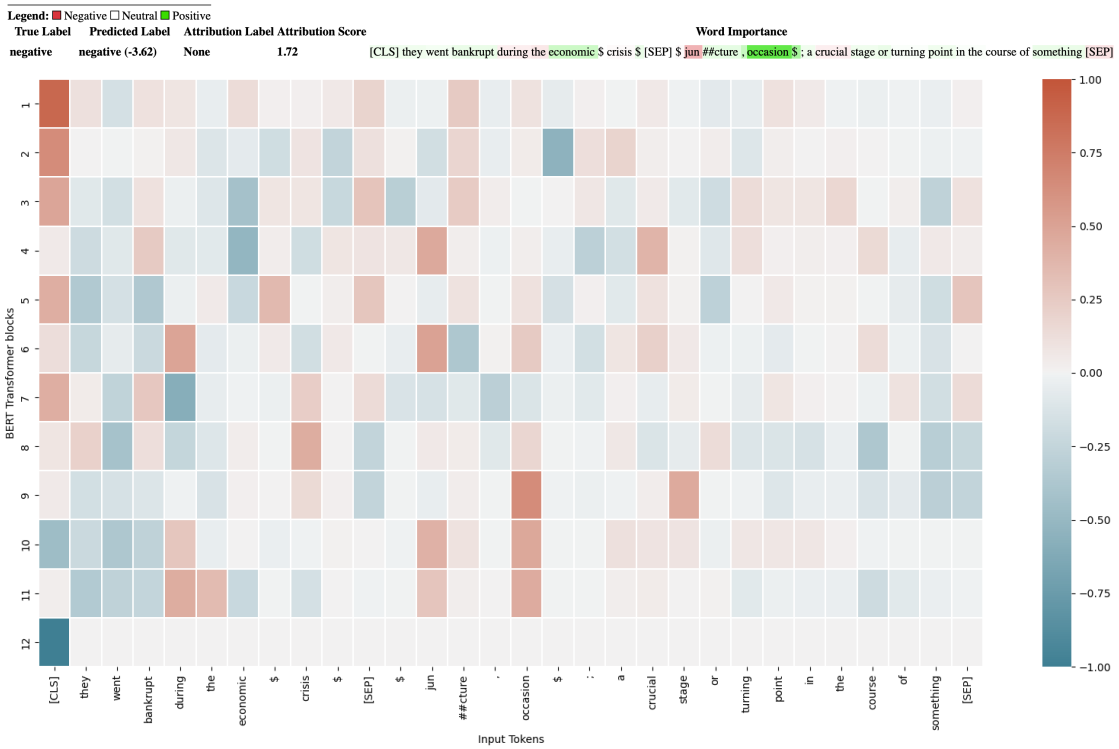


Figure 4.6.: Saliency map of input token attributions (top) and heat map of attributions across all 12 Transformer blocks (bottom) of HyperBertCLS on a correctly predicted example.

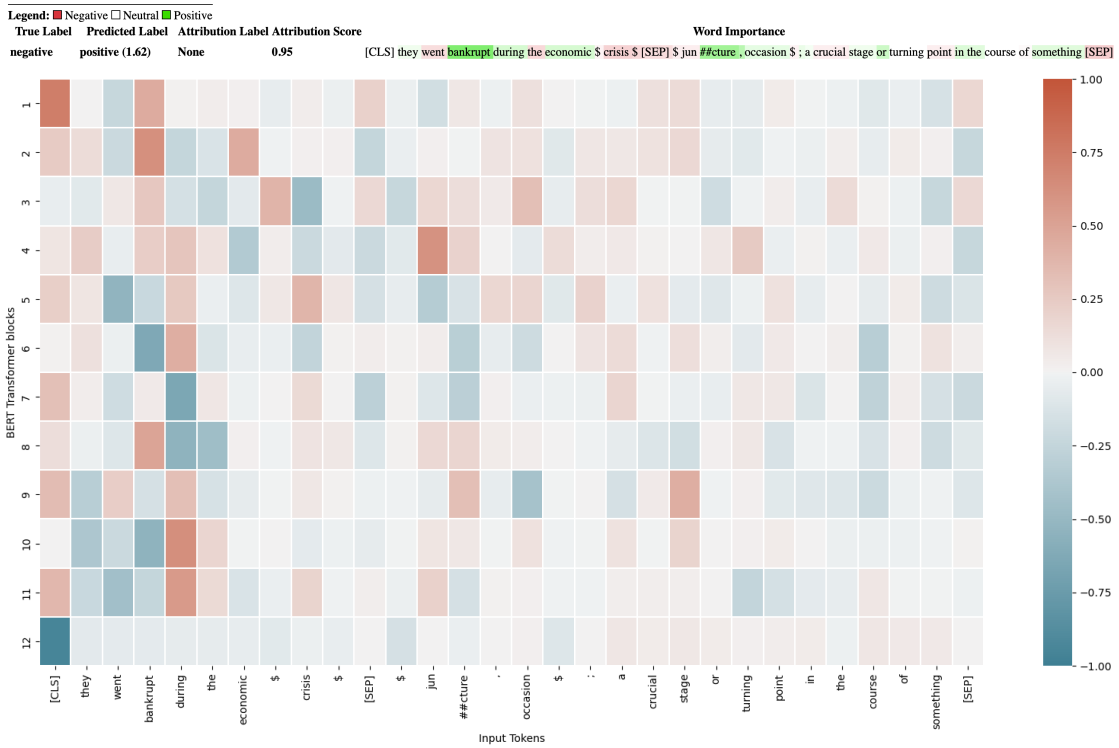


Figure 4.7.: Saliency map of input token attributions (top) and heat map of attributions across all 12 Transformer blocks (bottom) of AB4TSV on a falsely predicted example.

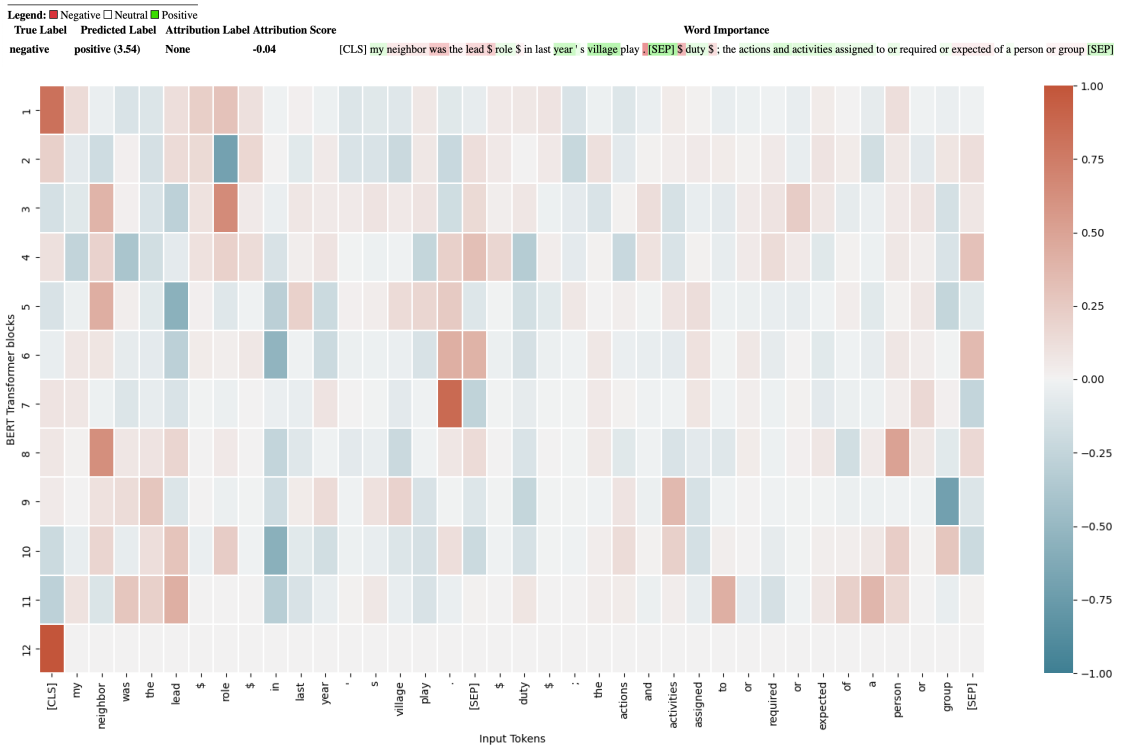


Figure 4.8.: Saliency map of input token attributions (top) and heat map of attributions across all 12 Transformer blocks (bottom) of HyperBertCLS on a falsely predicted example.

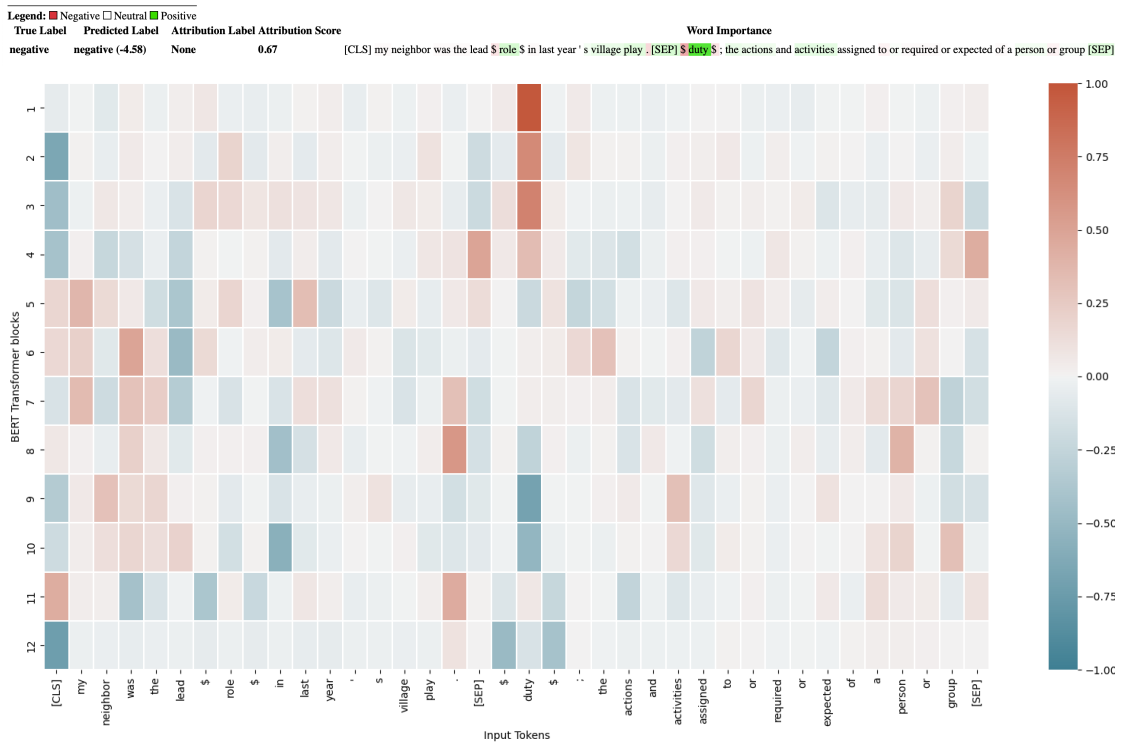


Figure 4.9.: Saliency map of input token attributions (top) and heat map of attributions across all 12 Transformer blocks (bottom) of AB4TSV on a correctly predicted example.

4.5. Summary

In this chapter, we have proposed an alternative formulation for TSV based on analogical reasoning. More precisely, we directly compared the underlying relations between several components of the input text (target, context, definition, hypernyms), by developing a Transformer-based architecture combined with a CNN classifier previously used for detecting analogies. The experimental results demonstrate the importance of the input encoding, suggesting that BERT is better at handling well-structured sentences or text that is specifically marked with special characters. Moreover, promoting invariance w.r.t. the permutations of analogical proportions during training result in a more robust system, that behaves consistently irrespective of the input encoding, and performs comparably to its initial version (without promoting permutation invariance). Both approaches achieve competitive results on the WiC-TSV evaluation benchmark, displaying some generalization capabilities even for domain-specific examples outside of the training data. Future work and perspectives are provided in Chapter 5.

5. Conclusion and perspectives

The general scope of the thesis is the integration of symbolic knowledge and reasoning with large language models (LLMs), in an attempt to address several limitations that such systems exhibit. More precisely, we designed methods for incorporating semantic knowledge and analogical reasoning within BERT based architectures, and evaluated both their impact in terms of performance on various downstream tasks and the robustness of the resulting models. This last chapter concludes the thesis with a summary of our contributions, in Section 5.1, and a discussion of future directions and perspectives, in Section 5.2.

5.1. Synopsis

Word vector representations play a fundamental role in many NLP applications. Exploiting symbolic knowledge was proven to improve the quality of word embeddings and their performance on many downstream tasks (Mrkšić et al., 2016, 2017; Ferret, 2017; Vulić et al., 2018; Lengerich et al., 2018). Retrofitting (Faruqui et al., 2015) is a simple and popular technique for refining distributional word embeddings based on relations coming from a semantic lexicon. Inspired by this technique, in Chapter 3 we present two methods for incorporating knowledge stored in semantic lexicons into BERT contextualized embeddings. Both methods are designed in order to encourage the learnt word embedding not to deviate too much from its original position in the latent space, while getting closer to the embeddings of its neighbouring words in the semantic lexicon. However, the main difference between both proposed methods lies in the way that the neighbouring information is encoded — by storing the contextualized embeddings of all occurrences of all similar words in the training set, or by replacing the underlying word in the sentence with every neighbouring word, one-by-one, and computing the contextualized embeddings on these new instances. In practice, both operations are applied within the architecture of BERT and take the form of a weighted average of the original and the neighbouring embeddings. After optimizing the hyperparameters of the retrofitting operations via grid search, we evaluate them on three biomedical datasets for relation extraction, and one movie review dataset for sentiment analysis. The retrofitted vectors do not substantially impact the performance for these tasks, so we conduct a qualitative analysis to provide further insights on this negative result. The grid search results indicate that the optimal choice of the hyperparameters becomes more localized as we retrofit closer to the output layer of BERT. After implementing a neighbourhood-based hyperparameter heuristic, we observe that performance is contrasted across datasets, suggesting that the optimal choice of these parameters is dataset-dependent. In addition, the proposed alternative

classification strategies point out that the use of the lexicons can significantly improve the performance across all tasks and datasets. Furthermore, ranking the Euclidean distances between a subset of vectors before and after the retrofitting operation reveals that a large proportion of these vectors either do not substantially move, or move too far with respect to their original position in the latent space. Consequently, there is a lot of variation in the neighbouring embeddings and therefore, not all words in the lexicon are relevant for the tasks at hand. To address this issue, we rely on McNemar's statistical test and reduce the size of the lexicons by selecting only relevant neighbours for a given word, with varying levels of confidence. This experiment shows that retrofitting in the sense of averaging embeddings can be meaningful, yet the size of the resulting reduced lexicons is not sufficiently large to be able to generalize at test time. Finally, we demonstrated that our method preserves the majority vote, which implies that retrofitting has potential provided that the lexicon can help.

LLMs such as BERT acquire a vast amount of knowledge during pretraining. Nonetheless, their ability to solve tasks that require reasoning over this knowledge is limited (Talmor et al., 2020; Rogers et al., 2020). Certain tasks can be improved by analogical reasoning over concepts (Keane and Smyth, 2020; Hüllermeier, 2020; Afantenos et al., 2021; Ushio et al., 2021a), e.g., understanding the underlying relations in *"man is to woman as king is to queen"*. Solving target sense verification (TSV) requires to decide whether the sense of a given a target word in a context matches or not that of a definition and a set of hypernyms for that particular word. In Chapter 4, we propose a way to formulate TSV as an analogy detection task, by transforming the input data into quadruples of the form $A : B :: C : D$. We present *Analogy and BERT for target sense verification* (AB4TSV), a model that uses BERT to represent the objects in these quadruples combined with a CNN classifier (Lim et al., 2019; Alsaidi et al., 2021a) to decide whether they constitute valid analogies. After motivating the set of possible choices for A, B, C, D, and introducing alternative ways to encode the input of BERT, we perform a grid search to find the best combinations $A : B :: C : D$ in terms of performance. The results demonstrate that for some input encodings, the representation of the hypernyms can degrade the end performance. However, this problem is not present when we enclose the hypernyms with focus characters or entity markers. Repeating the optimization experiment with additional input encodings shows that it is not sufficient to enclose the hypernyms with focus characters or entity markers at the input level. What really makes a difference is the inclusion of the embedding of these special characters in the computation of the hypernyms embedding. In fact, using solely one of the embeddings of these special characters to represent the hypernyms is sufficient to alleviate this issue. Following the axiomatization of Lepage (2004), we further incorporate symmetry and central permutation properties of analogies during training to assess whether analogical reasoning is beneficial in terms of performance on the task, and whether the AB4TSV naturally learns to be invariant to these properties, or if this must be explicitly enforced at training time. The results show a slight decrease in performance compared to the model trained without the axiomatic properties of analogies in the order of 1% absolute. However, they are consistent across all input encodings. Therefore, AB4TSV gains in ro-

bustness as it becomes less dependent on the choice of the input encoding. Furthermore, after giving as input the symmetry and central permutation quadruples to the analogy classifier, we notice that AB4TSV is not naturally invariant w.r.t. these properties, but they must be explicitly enforced during training. Next, we test our system on the WiC-TSV evaluation benchmark, and show that it can outperform existing approaches both in terms of accuracy and F1-score, irrespectively of whether the axiomatic properties of analogies are enforced at training time or not. Finally, feature attribution methods reveal that the `[CLS]` token embedding always gets assigned with extremely positive or negative attribution, meaning that it is a decisive factor for the final prediction. Although the rest of the tokens get nearly zero attribution for our BERT baseline model, AB4TSV assigns non trivial attributions to the components defined by the input quadruple.

In summary, throughout this thesis, we attempted to incorporate symbolic knowledge and reasoning with BERT-like models in order to boost their performance on downstream tasks and increase their robustness. For the former objective, we designed two retrofitting methods that inject information from semantic lexicons into BERT embeddings, and evaluated them on biomedical relation extraction and sentiment analysis of movie reviews. The qualitative analysis results demonstrate that the retrofitting operations have potential in improving the results, however more sophisticated mechanisms of selecting relevant neighbours for a given word in the semantic lexicons are needed, in order to avoid integrating noisy information in the embeddings. For the latter objective, we formulated TSV as an analogy detection problem, and we proposed AB4TSV, a hybrid architecture that achieved competitive results on the WiC-TSV evaluation benchmark. After including the axiomatic properties of analogies in the training phase of the model, we showed that it maintains a good performance, and nevertheless gains in robustness, as it becomes less dependent on the selection of the input encoding. We firmly believe that combining symbolic knowledge and reasoning with LLMs is the way forward for addressing the various limitations that modern AI systems exhibit, and we hope that this thesis provides useful material and encourages further research into this topic.

5.2. Perspectives

There are still many open research questions and challenges within the topic of symbolic knowledge and reasoning integration with LLMs. In this Section, we highlight a few of them, and provide future directions in the light of the contributions presented in this thesis.

5.2.1. Retrofitting with large language models

Below we pinpoint existing challenges with respect to the contributions presented in Chapter 3, and propose research avenues as a first step to addressing them.

5.2.1.1. Word importance

One of the main challenges when incorporating symbolic knowledge with LLMs is to decide what piece of information is relevant for the task at hand. As shown in Chapter 3, considering off-the-shelf semantic lexicons may result in neighbouring words that are unrelated under that particular domain. To deal with this issue, we proposed to reduce the size of the lexicons by keeping words based on some level of statistical confidence. However, it turns out that this approach is limited as it heavily relies on word statistics, and requires that words in the lexicon appear several times in the dataset.

Alternatively, techniques for measuring word importance could be a research avenue worth exploring to address this phenomenon. Specifically, in Section 4.4.4, we employed feature attribution methods, e.g., Saliency (Simonyan et al., 2014), in order to compute how much each token in the input sequence contributes to the final prediction of the model. Therefore, at test time, someone could compute the attribution scores for all tokens in the input, and choose to retrofit the embeddings of the ones that are assigned a high score and are present in the semantic lexicons.

Attention (Bahdanau et al., 2015) is another form of measuring word importance. Specifically, for Transformer architectures, as explained in Section 2.1.3.2, the normalized attention scores in equation (2.5) — also referred to as attention weights — intuitively express the importance that each token assigns to every individual token in the input sequence. Furthermore, based on the study of Vig (2019), multi-headed attention shows various attention patterns across the different attention heads and Transformer blocks of BERT. Therefore, using Method B that sequentially replaces the underlying word with neighbouring words from the semantic lexicons, it could be worth exploring how attention is distributed for each neighbour across the different heads and Transformer blocks. For example, we could imagine that we observe a delimiter-focused attention pattern, i.e., the attention is mostly concentrated in the separation tokens `[SEP]`, which, according to Vig (2019), indicates that the model cannot find any meaningful tokens to focus on in the input sequence. After replacing the word of interest with a neighbouring word, we might observe that attention gets distributed to other parts of the input, meaning that most likely this replacement is meaningful.

5.2.1.2. Linking BERT word embeddings and the classifier output

Early experiments in Chapter 3 indicate that retrofitting BERT embeddings at the word level does not impact the performance of the downstream task. Since BERT is finetuned using the `[CLS]` token representation as input to the classifier, we chose to retrofit this token instead. However, the link that finetuning creates between the `[CLS]` embedding and the final prediction is not explicit enough, as it is not clear how the trained system encodes relevant information in `[CLS]`.

A different approach would be to entirely avoid finetuning, which is possibly one of the reasons why there is no strong link between word embeddings and the classifier outcome. However, the pre-finetuning `[CLS]` token embedding of BERT is not a good proxy of the input sequence, meaning that most likely the downstream task performance will be

poor. One possible solution to this would be to retrofit word embeddings using pretrained BERT variations that were tuned for text/sentence embedding generation (Kapočiūtė-Dzikienė et al., 2021). The `[CLS]` token embedding of such models can provide a reliable generic sentence representation.

Besides the effects of finetuning, one probable cause of failing to retrofit BERT word embeddings may be related to the representation degeneration problem in contextual word representations (Gao et al., 2019). Unlike static word representations that are uniformly distributed, contextualized word vectors tend to only occupy a narrow cone in the embedding space (anisotropy), which can limit their representational power. Mu et al. (2018) propose a postprocessing method that creates more isotropic distributional word representations, by removing their top principal directions. To deal with representation degeneration, Bihani and Rayz (2021) apply this technique with BERT contextualized embeddings and then retrofit (Faruqui et al., 2015) in order to bring representations of same word senses closer in the embedding space. Their intrinsic evaluation reveals that the resulting representations are more isotropic with increased word sense disambiguation capabilities. It would be worth exploring whether isotropic BERT contextualized embeddings are a better fit for our proposed methods, both in the presence or the absence of finetuning.

5.2.1.3. Towards lexical systems

As previously described, a source of error in the proposed retrofitting methods is associated with the semantic lexicons themselves. Lexical databases, such as WordNet, mainly organize information in a hierarchical fashion and are usually centered around a specific relation, e.g., synonymy, while ignoring others like paradigmatic and syntagmatic that are also present in natural language (Mel'cuk, 1996). In contrast, non-ontological lexical systems emphasize the relational nature of lexicons where nodes represent complex entities of well specified word senses, and links between them are provided through lexical functions (Polguère, 2009). An interesting research avenue would be to assess how retrofitting methods can be integrated with lexical systems such as the French Lexical Network (Polguère, 2014), in more fine-grained tasks, e.g. anaphora/co-reference resolution, where we are certain that the symbolic knowledge is useful and we would not need to heavily rely on word statistics. This would require to account for multiple type of relations, thus the works of Mrkšić et al. (2016, 2017), Ferret (2017) and Lengerich et al. (2018) could be a good starting point.

5.2.2. Analogical reasoning with pretrained language models

Below we pinpoint existing challenges with respect to the contributions presented in Chapter 4, and propose some research avenues that could serve as a first step to addressing them.

5.2.2.1. Elimination of contextualized dependence

In Chapter 4, we formulated TSV as an analogy detection task and developed a hybrid architecture that uses BERT to represent the objects in the relation $A : B :: C : D$, and a CNN classifier to detect the absence or presence of the analogy. The model is finetuned for the task by processing all sources of information — context, definition and set of hypernyms for a given target word — at once in a single input sequence. This choice implies that the embeddings of all objects in $A : B :: C : D$ will share a lot of common information due to contextualization. We hypothesize that this dependence may not fully allow the classifier to model the underlying relations between the different objects, thus limiting its analogical reasoning capabilities. To verify this assumption, we propose to eliminate the contextual dependence of the objects of interest by disjoining the input sequence into separate subsequences and feeding them to BERT independently. For example, to extract the desired representations, one could input the context into BERT, and then feed it with the definition and the set of hypernyms in two individual runs. Alternatively, the use of a siamese network structure like SBERT (Reimers and Gurevych, 2019) could serve this purpose as well. It is interesting to notice that splitting the input sequence into distinct subsequences allows us to deal with TSV sub-tasks 1 and 2 (Breit et al., 2021), where either the definition or the set of hypernyms is given as source of information for the target sense, respectively. Particularly for sub-task 2, since hypernyms are simply a set of words separated by commas, feeding them solely into BERT might be problematic, as it was pretrained on well structured sentences. A possible workaround could be to replace the target word in the context by a single hypernym, and use the generated sentence as input to the model.

5.2.2.2. Beyond target sense verification

Analogical reasoning is key in designing AI systems that can generalize, learn efficiently in the absence of large amounts of training data, and perform sophisticated transfer learning across domains (Mitchell, 2021). In Chapter 4, we empirically demonstrated the benefits of combining LLMs and analogies by solving TSV. A natural extension would be to adapt our methodology to the more general task of word sense disambiguation. The main difference lies in that the latter requires the model to select the relevant sense for the target word in context, based on a set of possible candidates from a sense inventory, e.g., WordNet. For example, Huang et al. (2019) address this problem by finetuning BERT using positive and negative context-gloss pairs. Given that the information about the gloss and the hypernyms is already provided by the sense inventory, AB4TSV is directly applicable on this task.

Overall, the use of analogies with LLMs is an understudied area in AI. It would be worth investigating the potential of teaching analogical reasoning to LLMs, and evaluating it on different types of problems like recent works do on relation classification (Ushio et al., 2021a) and bilingual dictionary induction and sentence retrieval (Garneau et al., 2021).

5.2.3. Towards a unified knowledge and reasoning integration architecture

In a realistic scenario, the required type of symbolic knowledge and reasoning needed to guide the model into making the right decision for the task at hand, varies depending on the context, e.g., commonsense, encyclopedic, grammar, etc. Therefore, the ultimate goal is to design a single joint approach that could correctly exploit various knowledge bases and employ different reasoning mechanisms at the same time.

Towards this goal we focus on the editability of factual knowledge in LLMs. Petroni et al. (2019) claim that LLMs acquire factual knowledge as they are exposed to large amounts of text during pretraining. This is evident from their ability to recall facts — by probing them for world knowledge via cloze-type queries — without having to explicitly tailor them for this particular task. However, facts can change over time and retraining the whole model in order to reflect those changes is often prohibited. Moreover, the factual knowledge stored in the parameters of the black-box LLM is hard to identify, interpret, and modify. Finding clever ways to edit the knowledge inside LLMs would allow us to fix mistakes due to factual knowledge stored by the model becoming stale over time, overwrite unintentionally memorized sensitive information, eliminate biases to ensure a fair application of such models in real-world, and make a step towards understanding huge opaque neural networks. We categorize existing approaches to updating the factual knowledge in LLMs into finetuning (Chen et al., 2021; Zhu et al., 2020), meta-learning (Sinitsin et al., 2020; De Cao et al., 2021; Mitchell et al., 2022), causal intervention (Dai et al., 2022; Meng et al., 2022) and end-to-end (Logan et al., 2019).

To this end, our starting point is KGLM (Logan et al., 2019) a language generation model that maintains a dynamically growing local knowledge graph, from which it can select and copy information at inference time using COPYNET (Gu et al., 2016). Giving an example on factual completion, the authors claim that KGLM is directly controllable via modifications to the knowledge graph. Encoding the fact *(Barack Obama, birthDate, 1961-08-04)* to *"Barack Obama was born on _____"* leads to "August", "4" and "1961" as top-3 predictions of KGLM. After manually changing the *birthDate* to "2013-03-21" the top-3 predictions of the model automatically changed to "March" and "21", "2013". Although this approach changes the final prediction instead of the parameters of the model, its hybrid architecture and copying mechanism constitutes a good starting point towards a unified knowledge and reasoning integration approach. In the future, we would like to integrate the dynamic local knowledge graph and copying mechanisms from KGLM with LLMs such BERT or GPT, and explore the capabilities of the hybrid model on fact completion.

5.3. Epilogue

Large language models may currently be at the forefront of technological advancements, however they are far from reaching human-like perception when it comes to understanding, learning, and reasoning with natural language. Their lack of transparency, fairness, and privacy renders them unreliable for certain applications, and consequently makes it

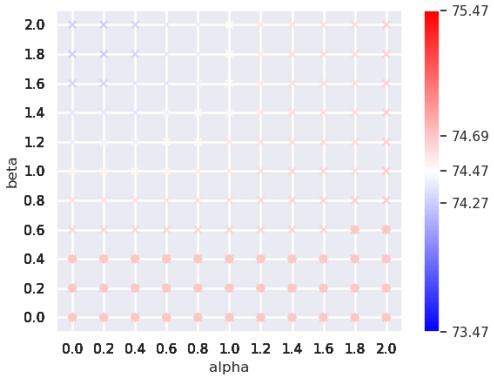
hard to enable trust from humans. Natural language text is discrete, and therefore there is a lot of missing information in the data used to train these models, from syntactical and grammatical rules to knowledge about words/concepts of interest. Knowledge bases are typically rich with that type of information. Designing hybrid approaches that incorporate symbolic knowledge with LLMs not only bridges the gap between symbolic and subsymbolic AI, but also encourages the development of novel reasoning mechanisms that can exploit this knowledge and advance the state of the art in NLP. Through this thesis, we position in favour of the hybridization of symbolic knowledge and reasoning with LLMs, as we believe it to be key towards a more powerful and transparent AI.

Appendices

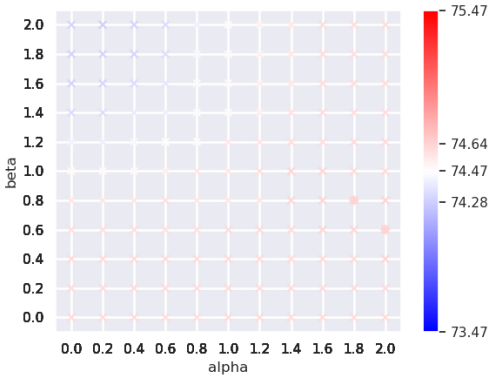
A. Supplementary material from Chapter 3

A.1. Grid search visualizations

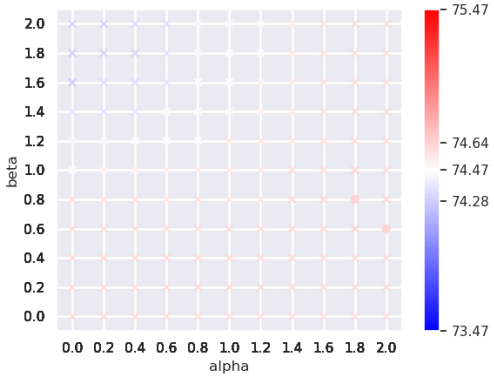
Here we provide the grid search plots for the proposed retrofitting approaches across all different settings and datasets. The white colour corresponds to the baseline score and the red asterisk indicates the best (α, β)-pair performance on the development set for Method A. The green colour bar indicates the best α-values on the development set for Method B, while the horizontal lines show the top performance of all proposed strategies (see Section 3.2.6).



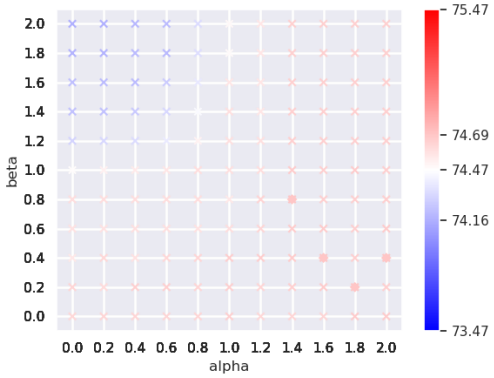
annotated-16



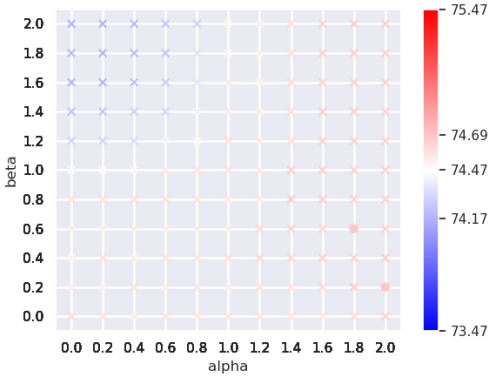
annotated-34



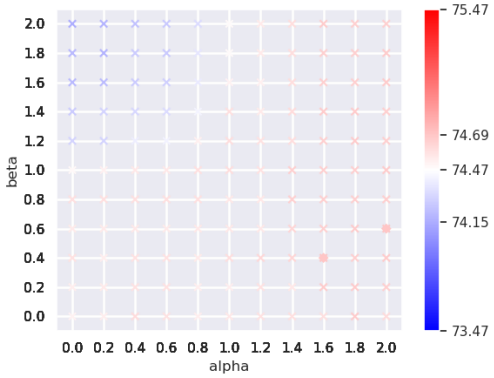
annotated-50



expanded-16

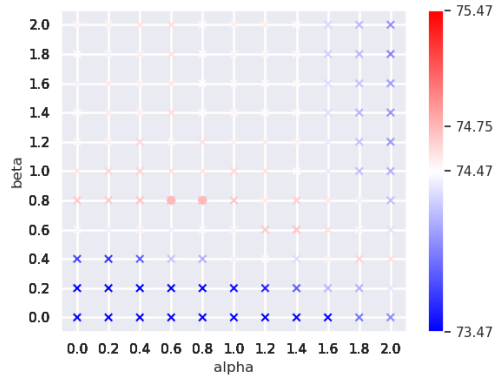


expanded-34

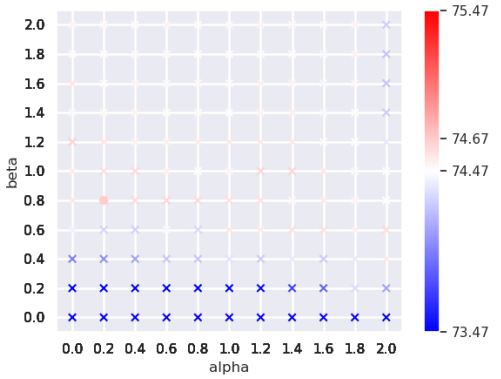


expanded-50

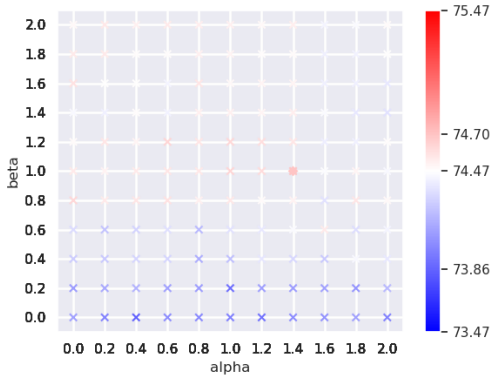
Figure A.1.: Grid search plots of micro F_1-scores for Method A while retrofitting before layer normalization at Transformer block 11 on ChemProt.



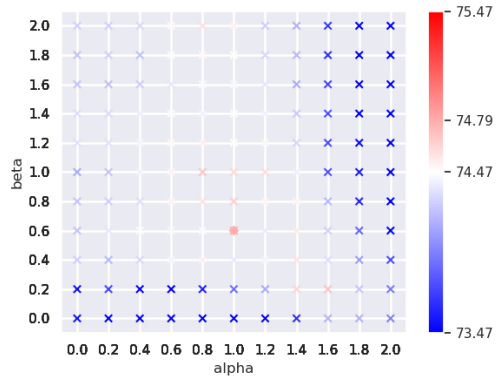
annotated-16



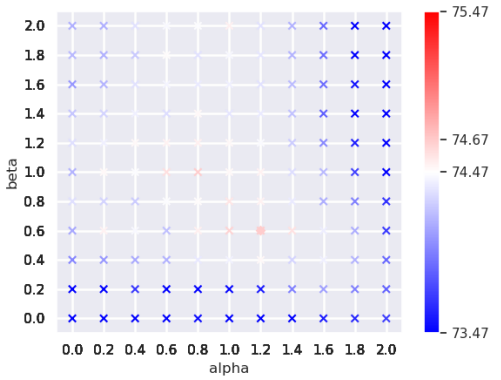
annotated-34



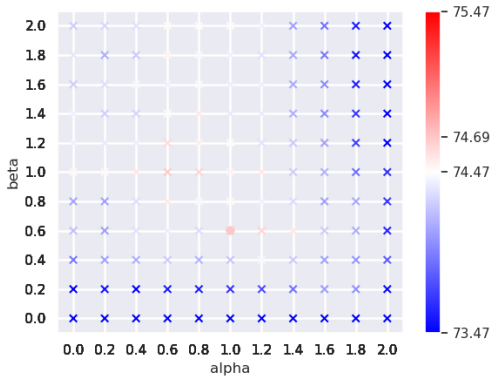
annotated-50



expanded-16

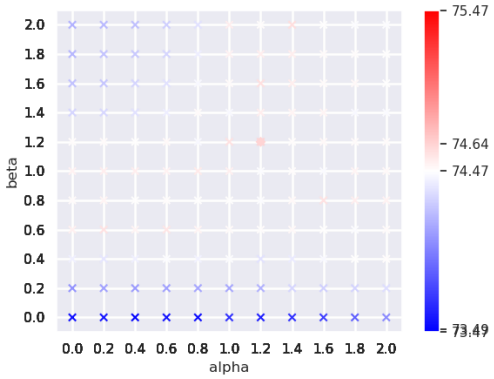


expanded-34

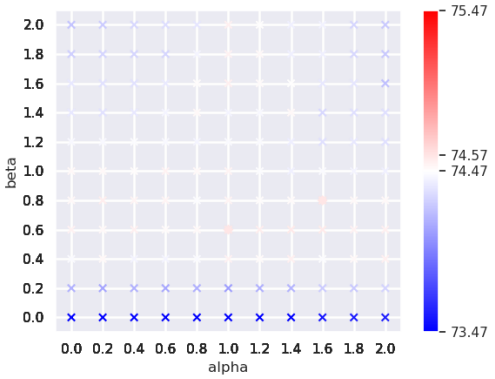


expanded-50

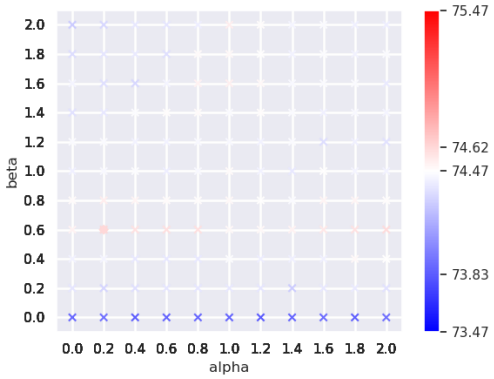
Figure A.2.: Grid search plots of micro F_1-scores for Method A while retrofitting after layer normalization at Transformer block 11 on ChemProt.



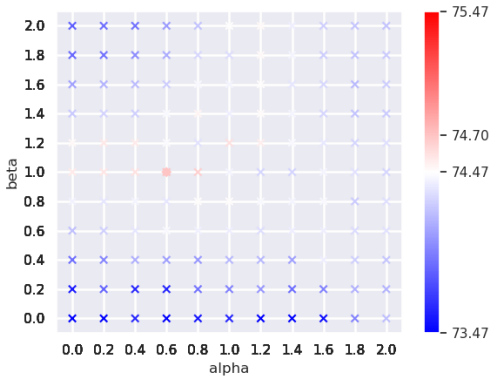
annotated-16



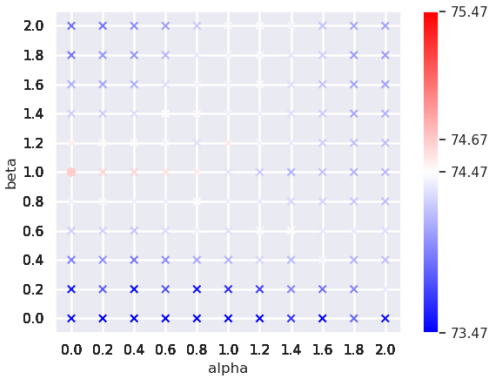
annotated-34



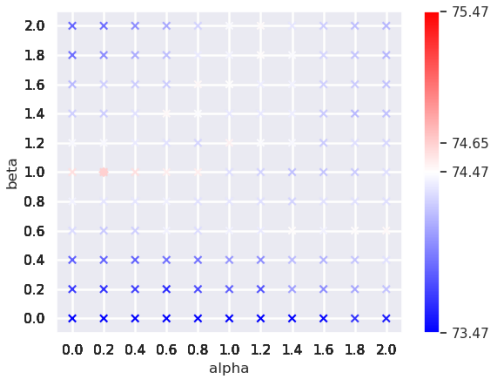
annotated-50



expanded-16

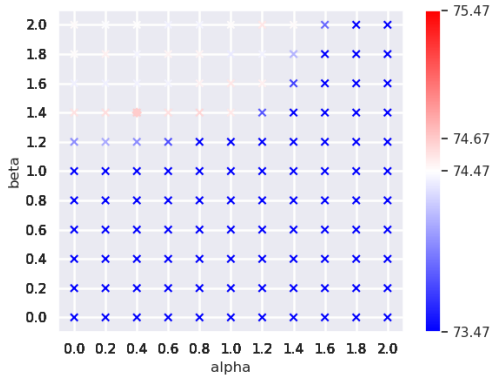


expanded-34

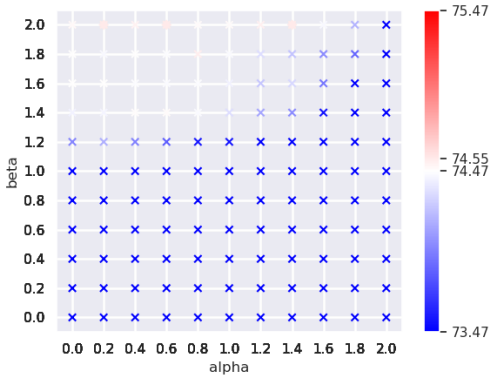


expanded-50

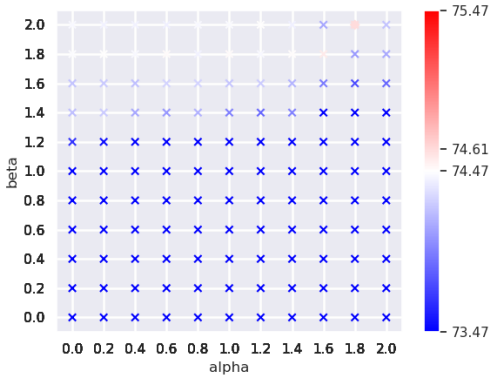
Figure A.3.: Grid search plots of micro F_1-scores for Method A while retrofitting before layer normalization at Transformer block 12 on ChemProt.



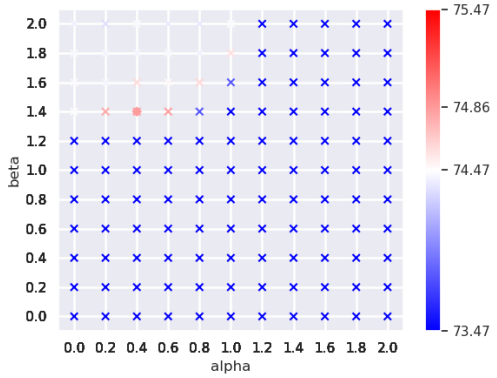
annotated-16



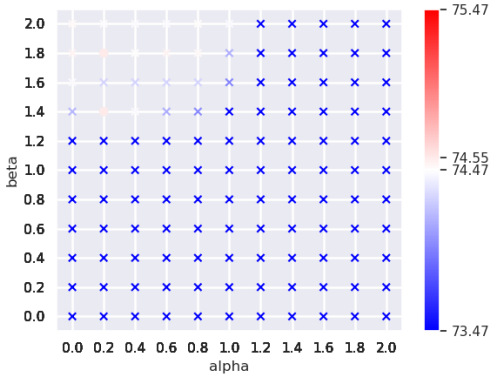
annotated-34



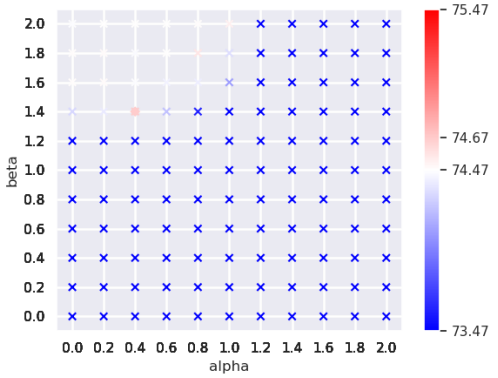
annotated-50



expanded-16

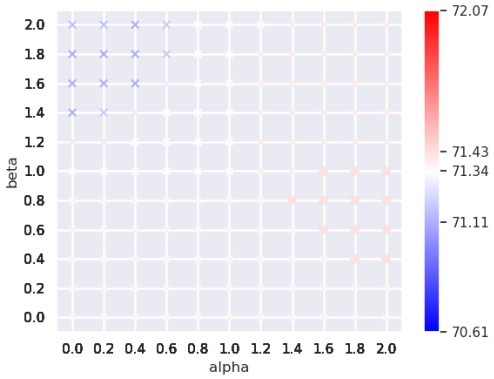


expanded-34

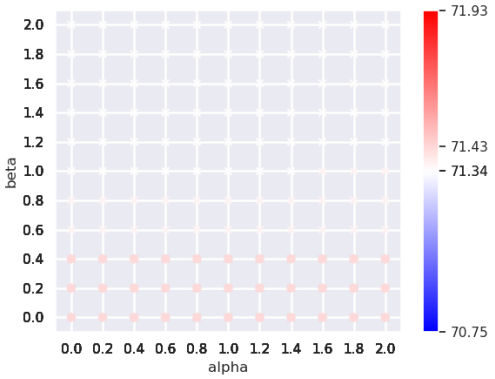


expanded-50

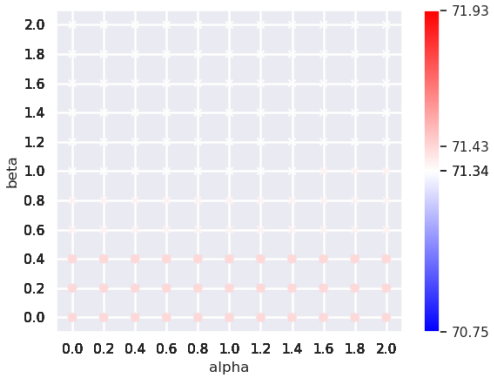
Figure A.4.: Grid search plots of micro F_1-scores for Method A while retrofitting after layer normalization at Transformer block 12 on ChemProt.



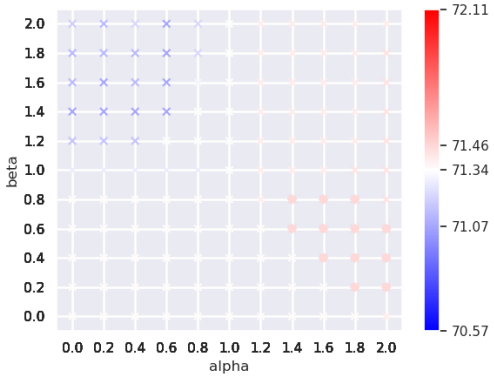
annotated-16



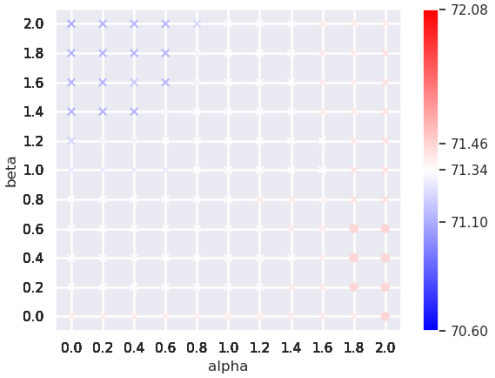
annotated-34



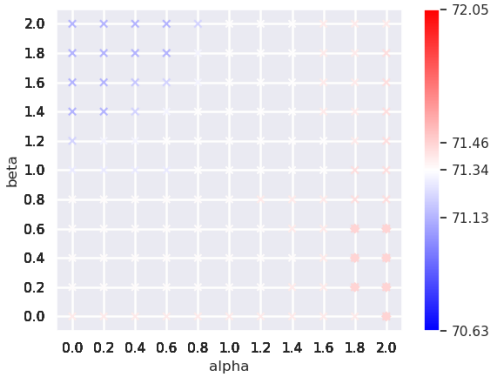
annotated-50



expanded-16

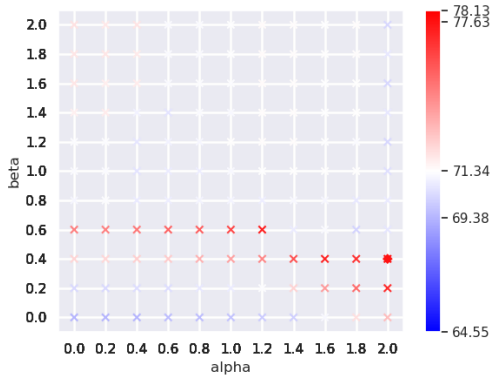


expanded-34

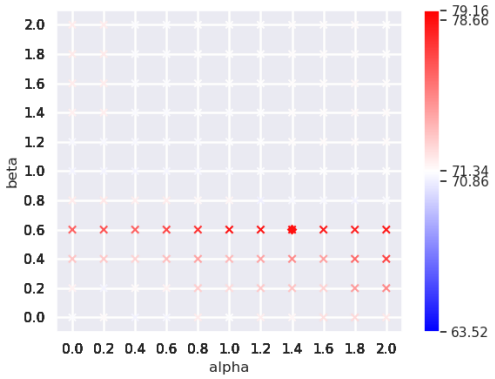


expanded-50

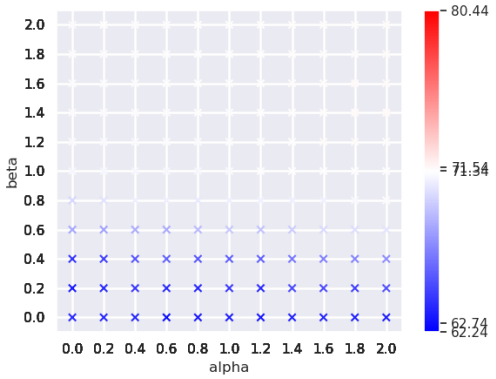
Figure A.5.: Grid search plots of micro F_1-scores for Method A while retrofitting before layer normalization at Transformer block 11 on DDI.



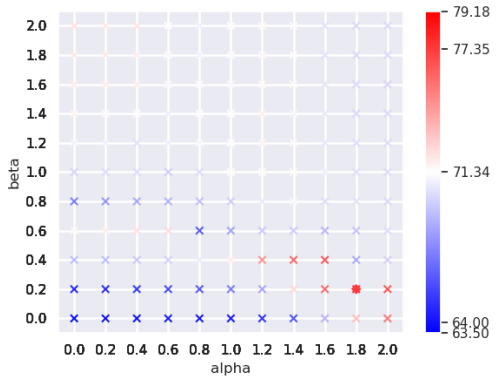
annotated-16



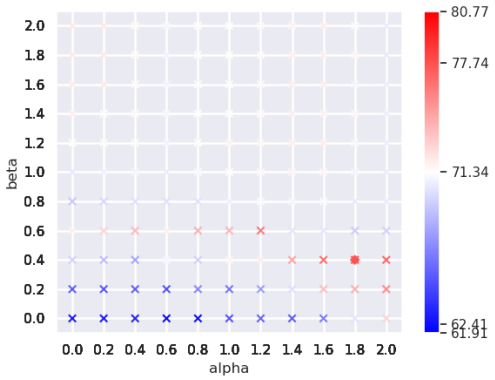
annotated-34



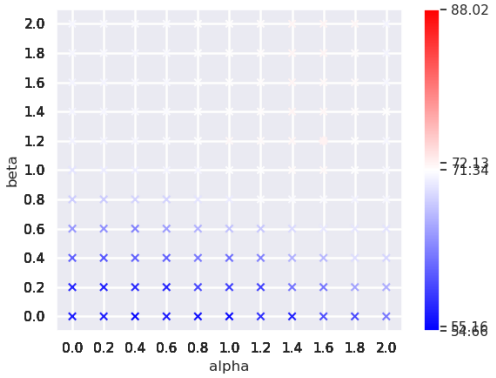
annotated-50



expanded-16

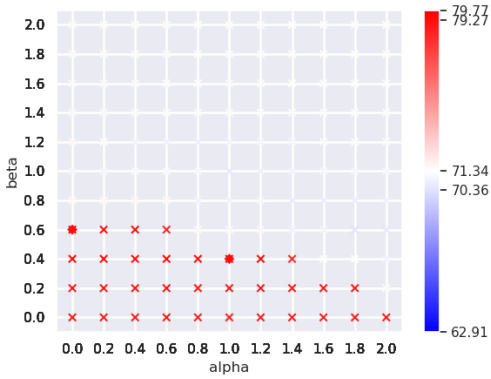


expanded-34

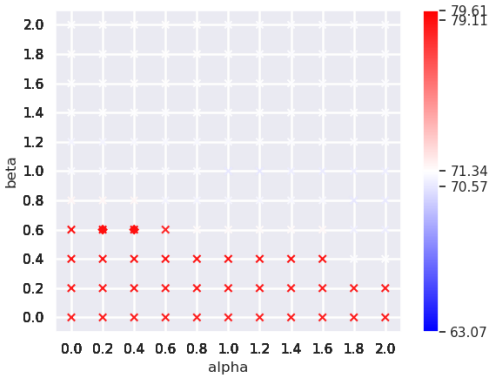


expanded-50

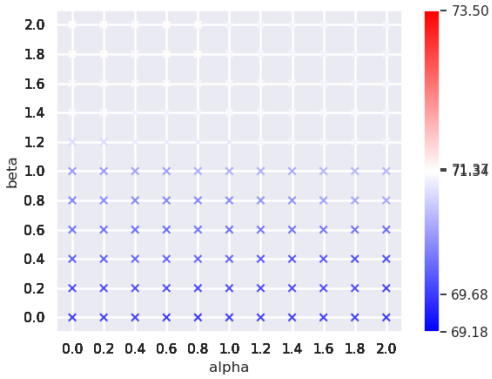
Figure A.6.: Grid search plots of micro F_1-scores for Method A while retrofitting after layer normalization at Transformer block 11 on DDI.



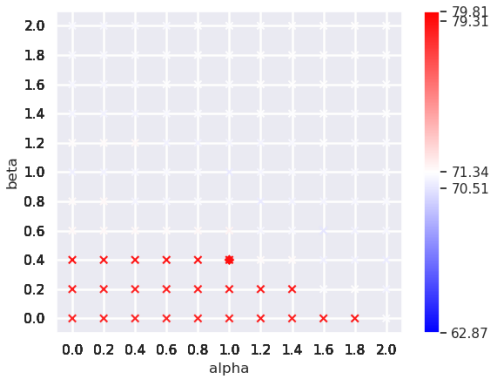
annotated-16



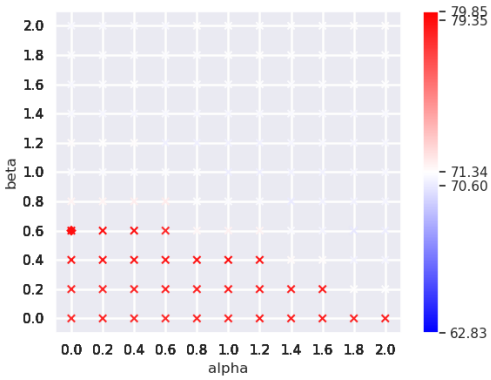
annotated-34



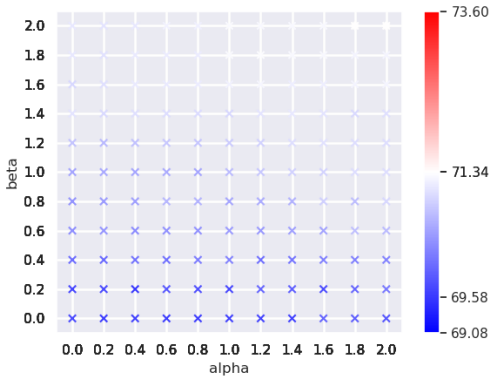
annotated-50



expanded-16

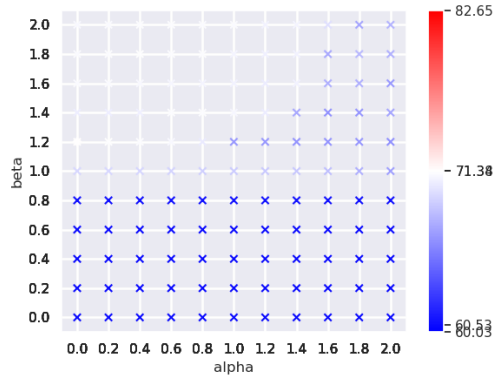


expanded-34

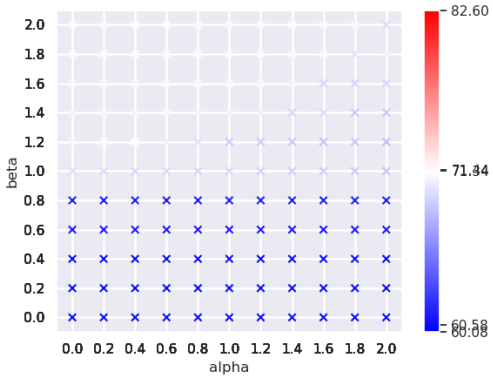


expanded-50

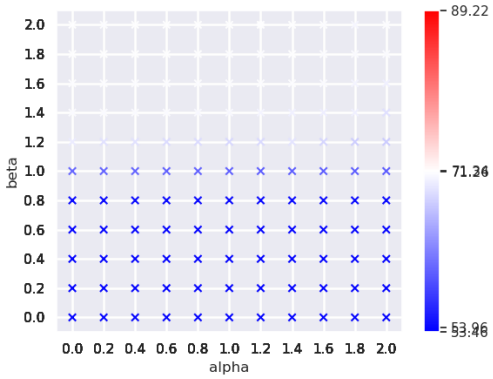
Figure A.7.: Grid search plots of micro F_1-scores for Method A while retrofitting before layer normalization at Transformer block 12 on DDI.



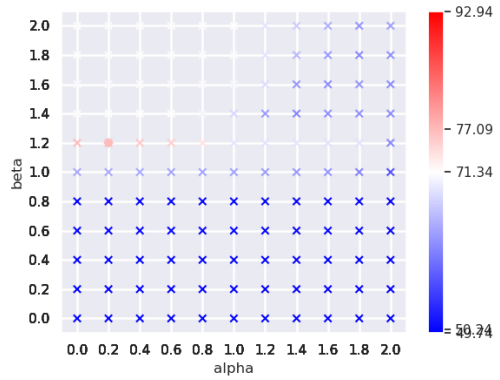
annotated-16



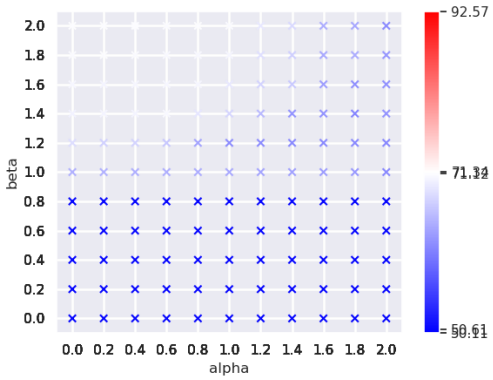
annotated-34



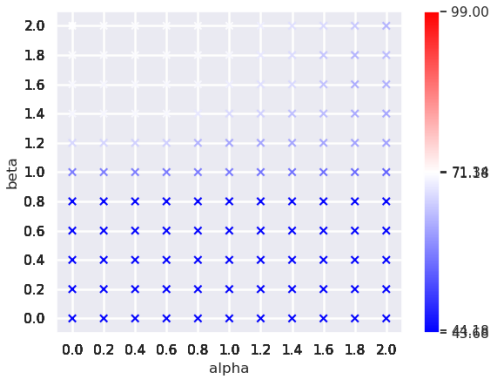
annotated-50



expanded-16

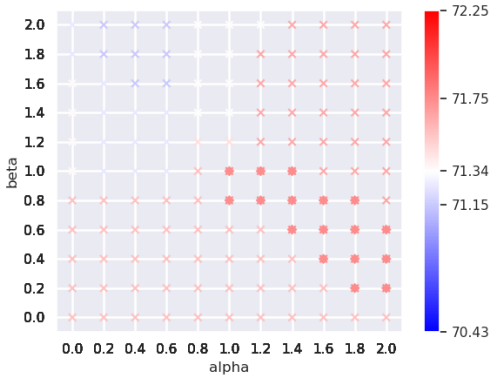


expanded-34

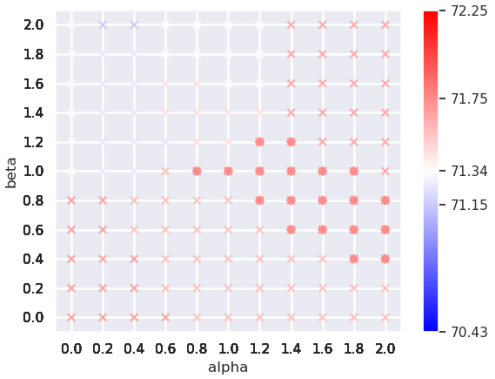


expanded-50

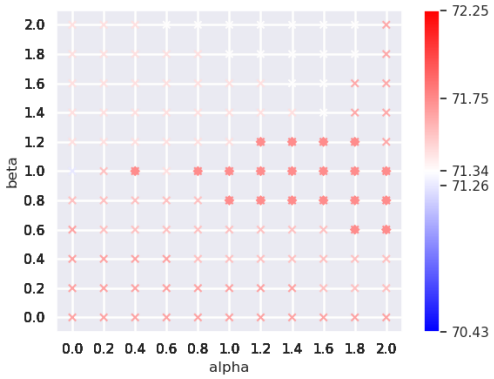
Figure A.8.: Grid search plots of micro F_1-scores for Method A while retrofitting after layer normalization at Transformer block 12 on DDI.



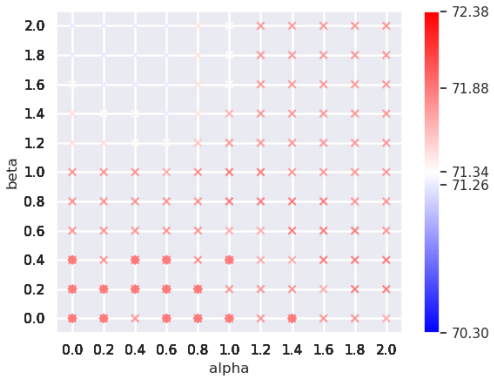
annotated-16



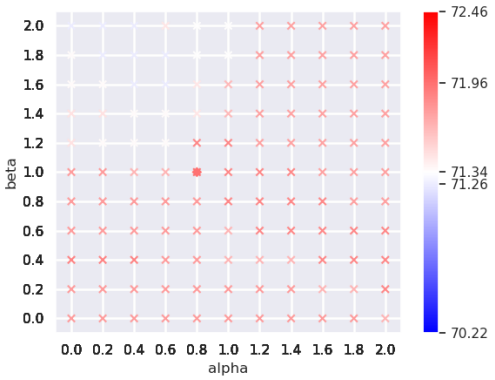
annotated-34



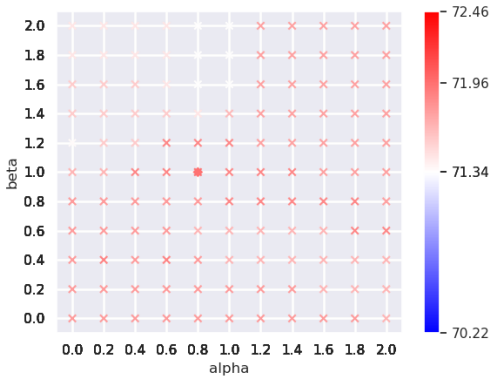
annotated-50



expanded-16

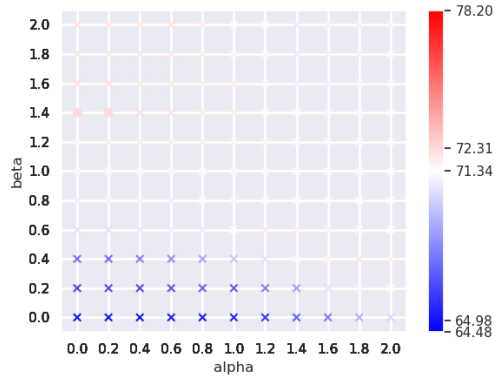


expanded-34

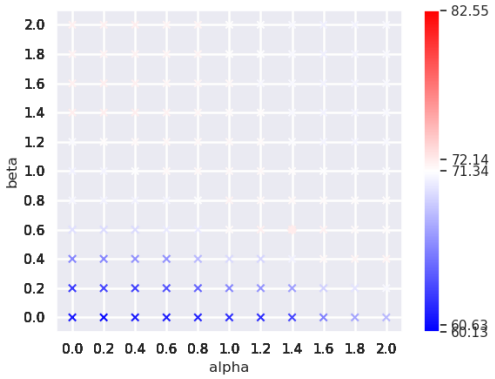


expanded-50

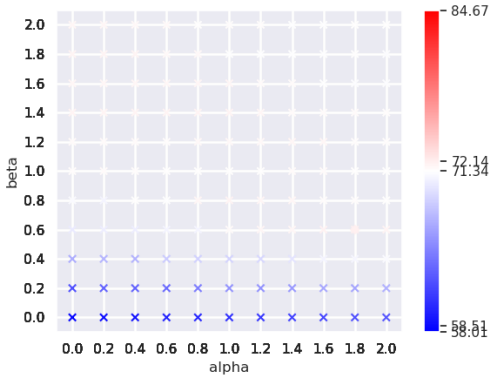
Figure A.9.: Grid search plots of micro F_1-scores for Method A while retrofitting before layer normalization at Transformer block 11 on i2b2-2010.



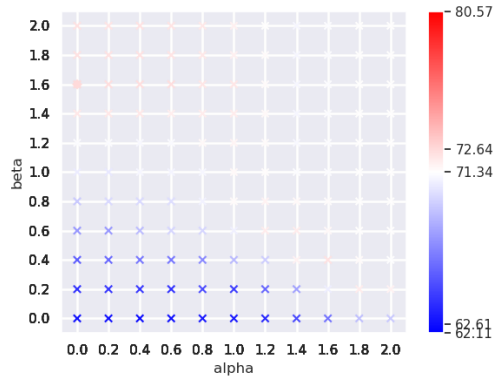
annotated-16



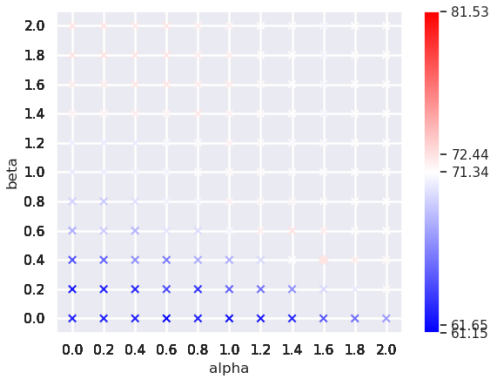
annotated-34



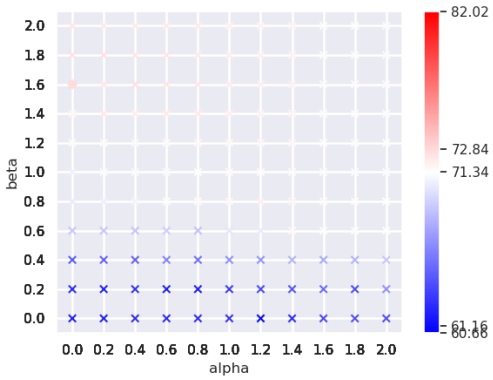
annotated-50



expanded-16

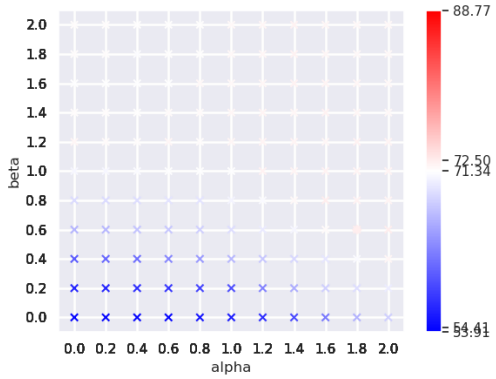


expanded-34

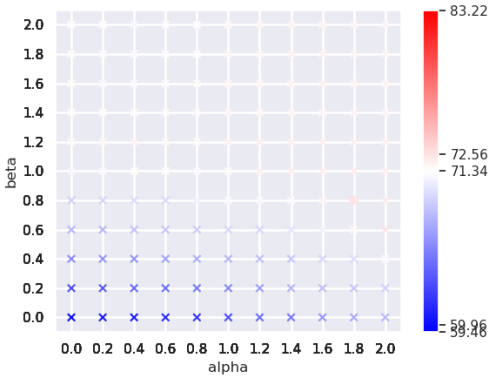


expanded-50

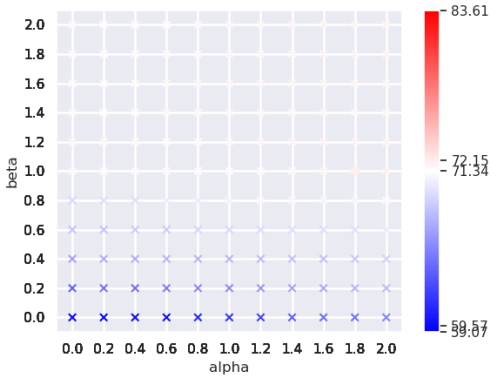
Figure A.10.: Grid search plots of micro F_1-scores for Method A while retrofitting after layer normalization at Transformer block 11 on i2b2-2010.



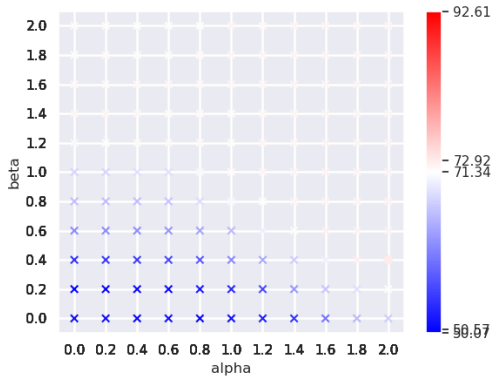
annotated-16



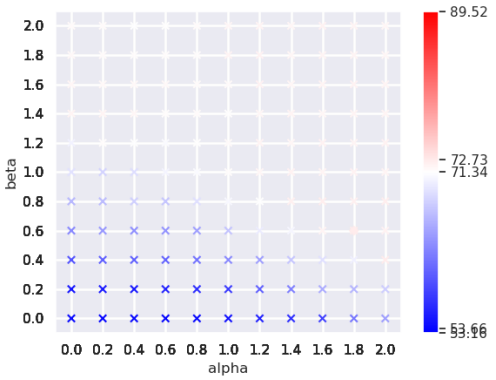
annotated-34



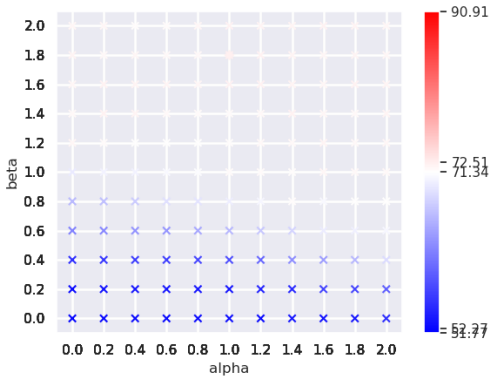
annotated-50



expanded-16

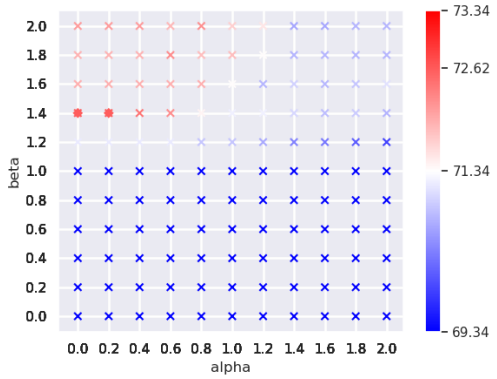


expanded-34

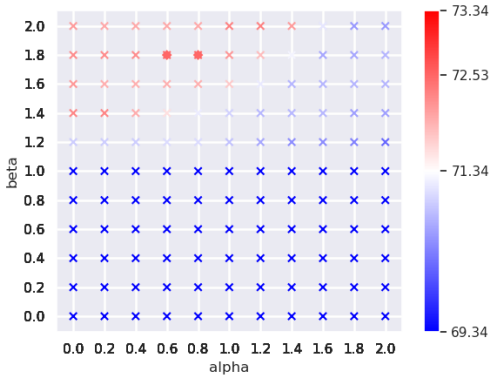


expanded-50

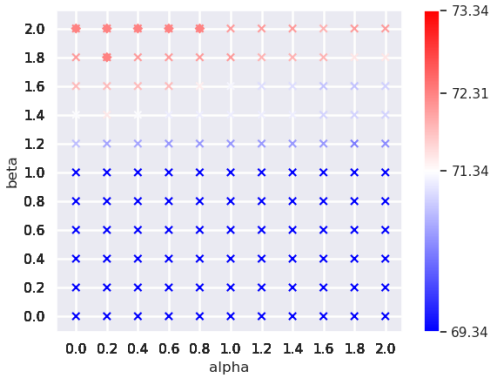
Figure A.11.: Grid search plots of micro F_1-scores for Method A while retrofitting before layer normalization at Transformer block 12 on i2b2-2010.



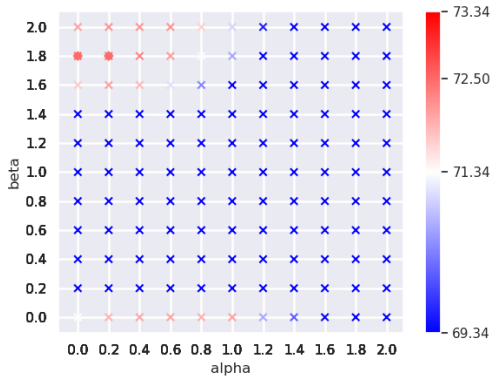
annotated-16



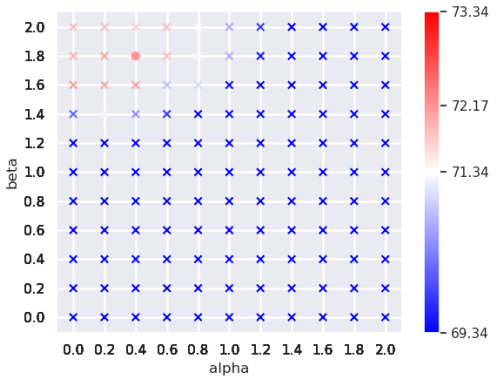
annotated-34



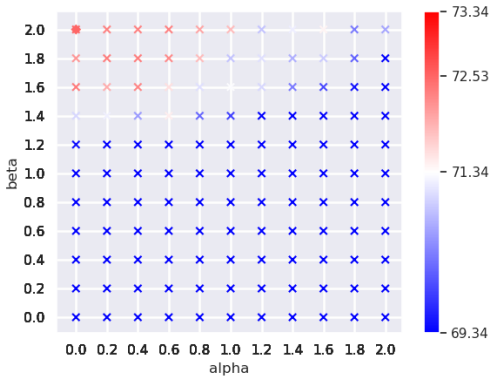
annotated-50



expanded-16

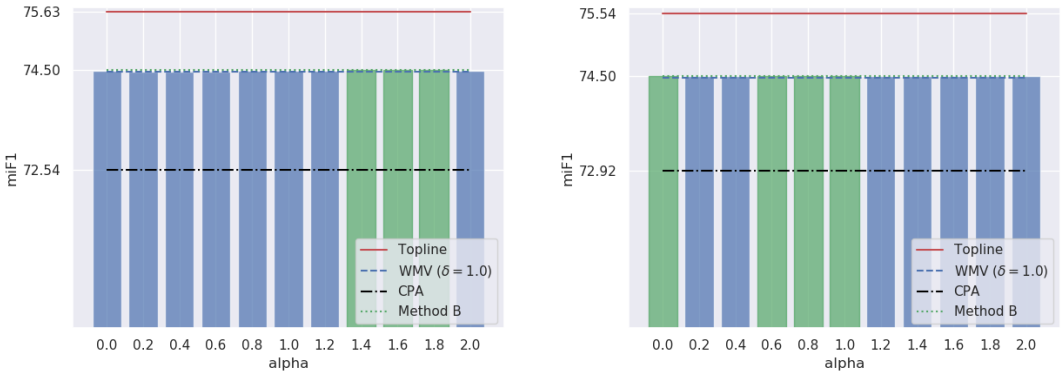


expanded-34



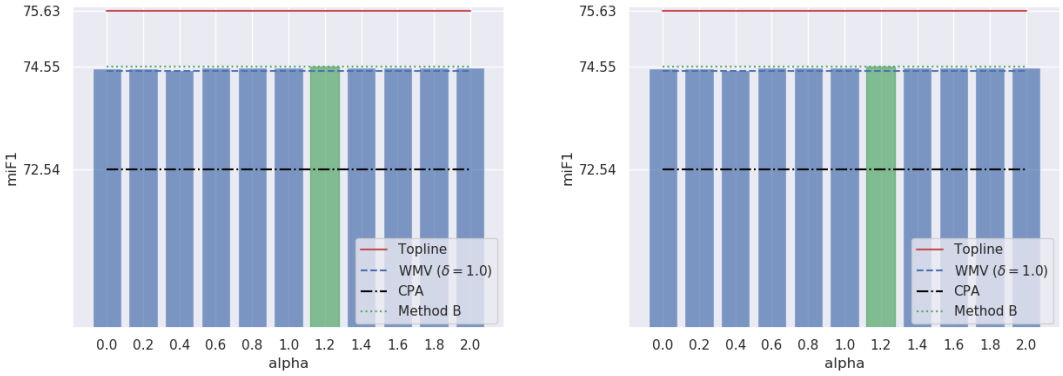
expanded-50

Figure A.12.: Grid search plots of micro F_1-scores for Method A while retrofitting after layer normalization at Transformer block 12 on i2b2-2010.



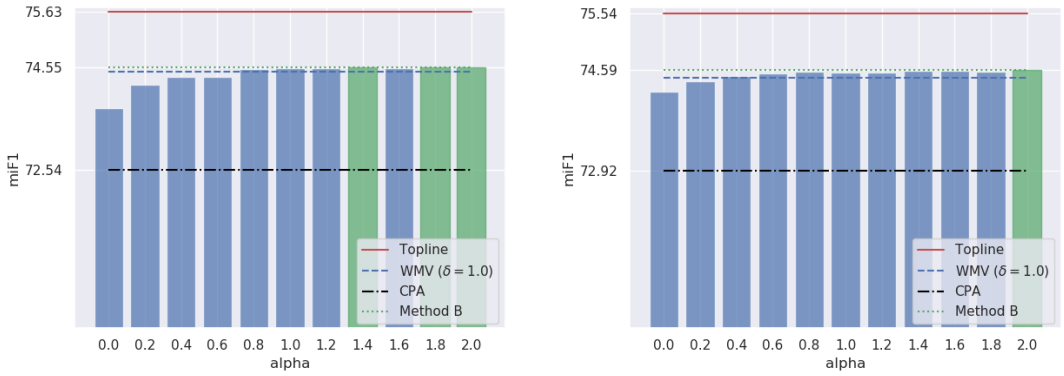
annotated-34 annotated-50

Figure A.13.: Grid search plots of accuracy scores for Method B while retrofitting before layer normalization at Transformer block 11 on ChemProt.



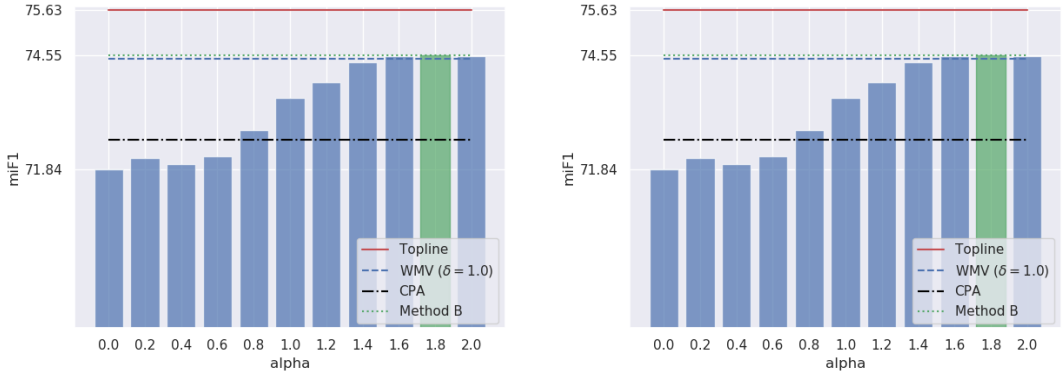
annotated-34 annotated-50

Figure A.14.: Grid search plots of accuracy scores for Method B while retrofitting after layer normalization at Transformer block 11 on ChemProt.



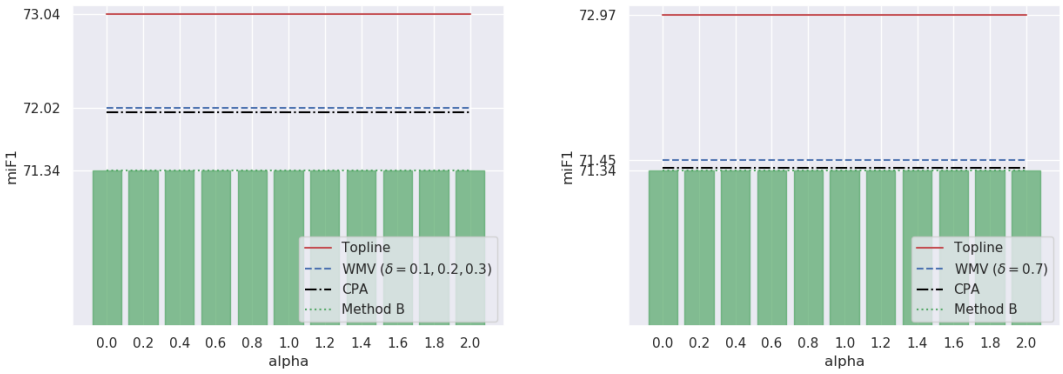
annotated-34 annotated-50

Figure A.15.: Grid search plots of accuracy scores for Method B while retrofitting before layer normalization at Transformer block 12 on ChemProt.



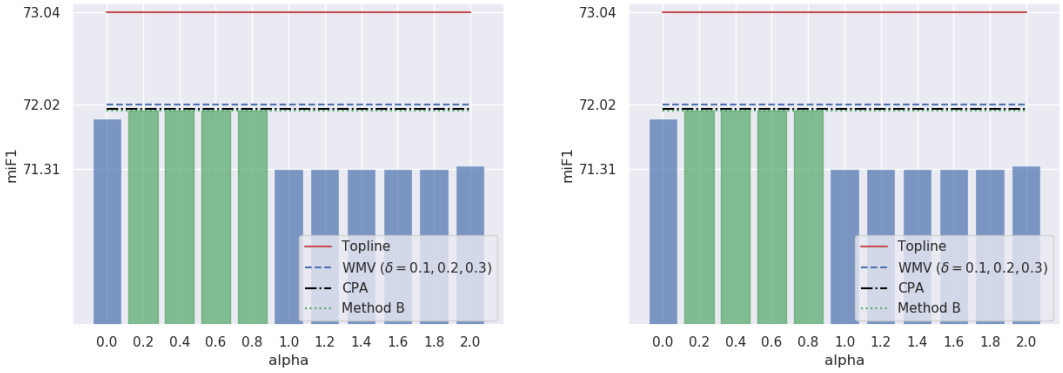
annotated-34 annotated-50

Figure A.16.: Grid search plots of accuracy scores for Method B while retrofitting after layer normalization at Transformer block 12 on ChemProt.



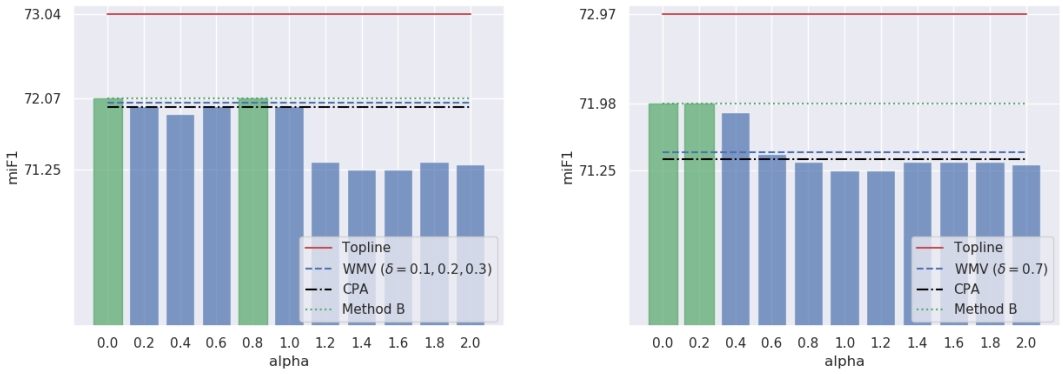
annotated-34 annotated-50

Figure A.17.: Grid search plots of accuracy scores for Method B while retrofitting before layer normalization at Transformer block 11 on DDI.



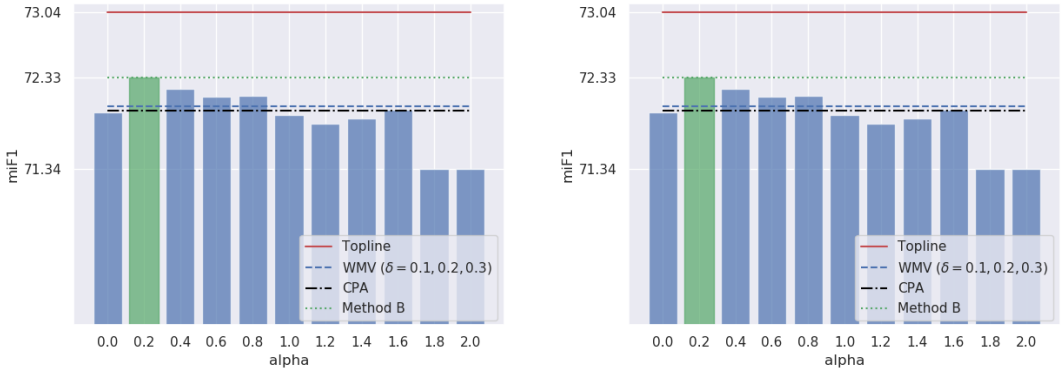
annotated-34 annotated-50

Figure A.18.: Grid search plots of accuracy scores for Method B while retrofitting after layer normalization at Transformer block 11 on DDI.



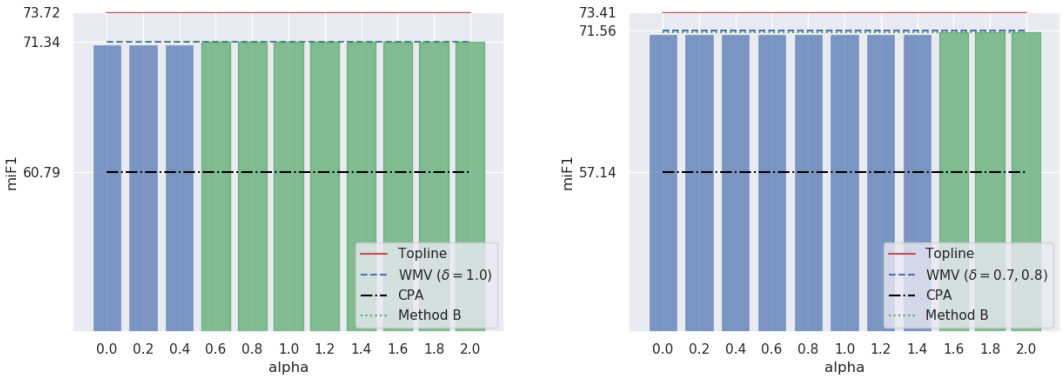
annotated-34 annotated-50

Figure A.19.: Grid search plots of accuracy scores for Method B while retrofitting before layer normalization at Transformer block 12 on DDI.



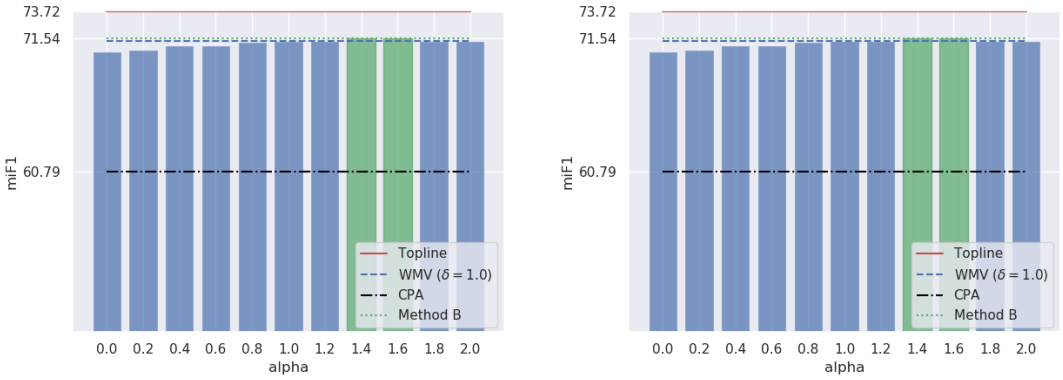
annotated-34 annotated-50

Figure A.20.: Grid search plots of accuracy scores for Method B while retrofitting after layer normalization at Transformer block 12 on DDI.



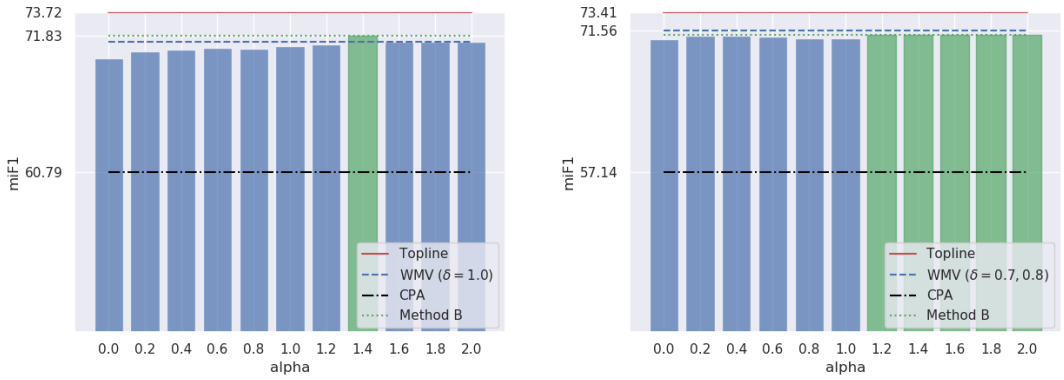
annotated-34 annotated-50

Figure A.21.: Grid search plots of accuracy scores for Method B while retrofitting before layer normalization at Transformer block 11 on i2b2-2010.



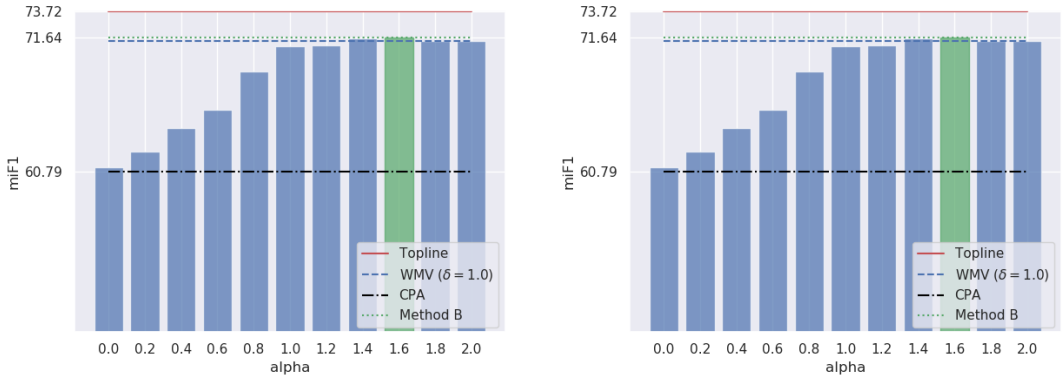
annotated-34 annotated-50

Figure A.22.: Grid search plots of accuracy scores for Method B while retrofitting after layer normalization at Transformer block 11 on i2b2-2010.



annotated-34 annotated-50

Figure A.23.: Grid search plots of accuracy scores for Method B while retrofitting before layer normalization at Transformer block 12 on i2b2-2010.



annotated-34 annotated-50

Figure A.24.: Grid search plots of accuracy scores for Method B while retrofitting after layer normalization at Transformer block 12 on i2b2-2010.

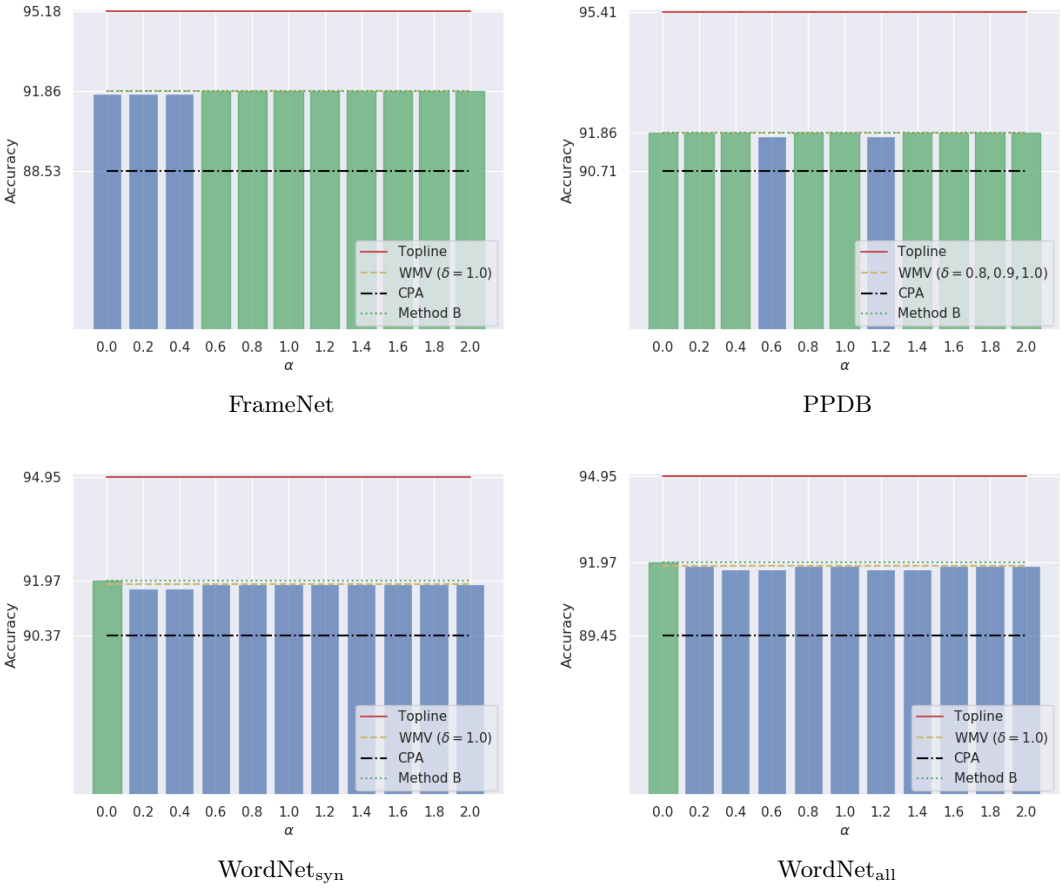


Figure A.25.: Grid search plots of accuracy scores for Method B while retrofitting before layer normalization at Transformer block 11 on SST-2.

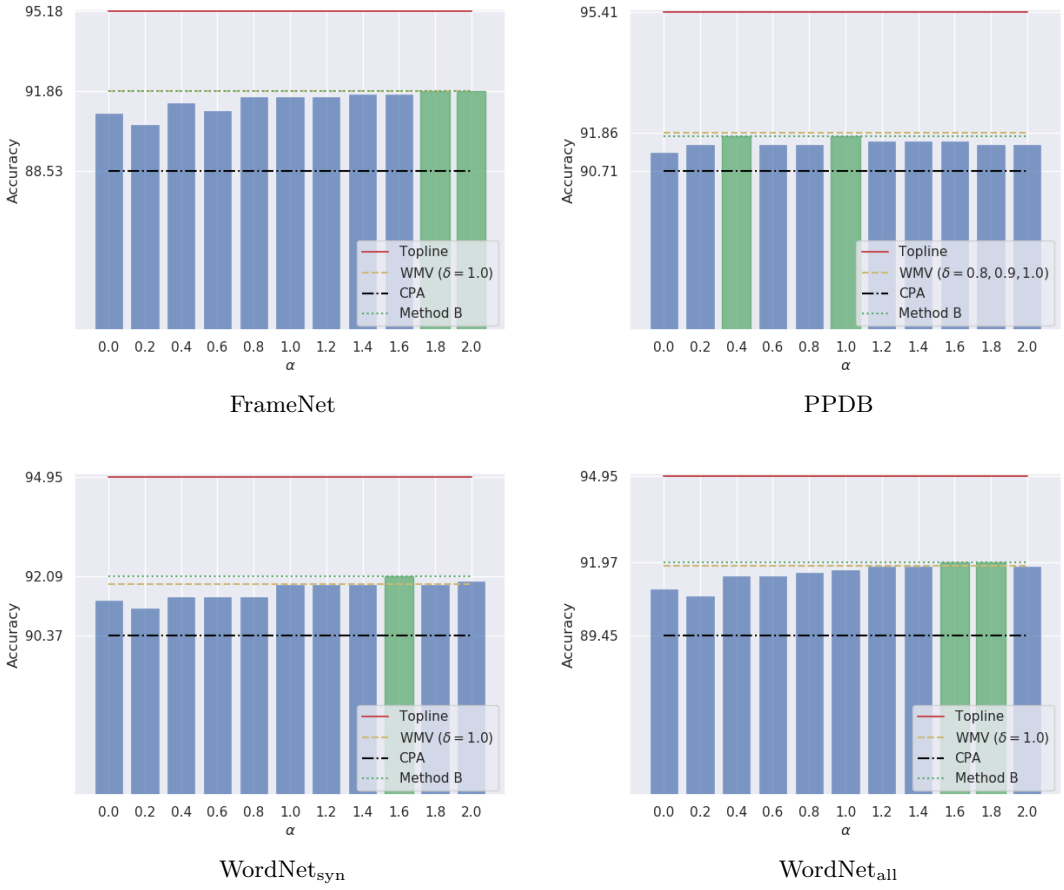


Figure A.26.: Grid search plots of accuracy scores for Method B while retrofitting after layer normalization at Transformer block 11 on SST-2.

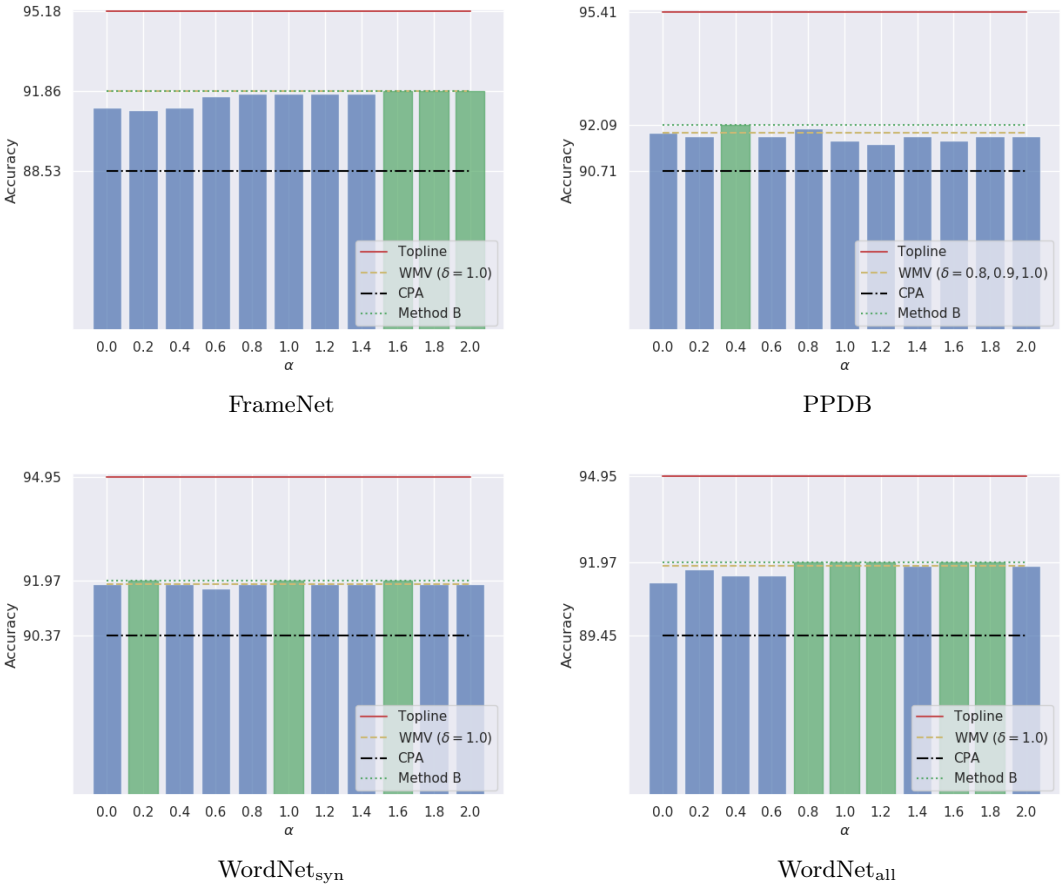


Figure A.27.: Grid search plots of accuracy scores for Method B while retrofitting before layer normalization at Transformer block 12 on SST-2.

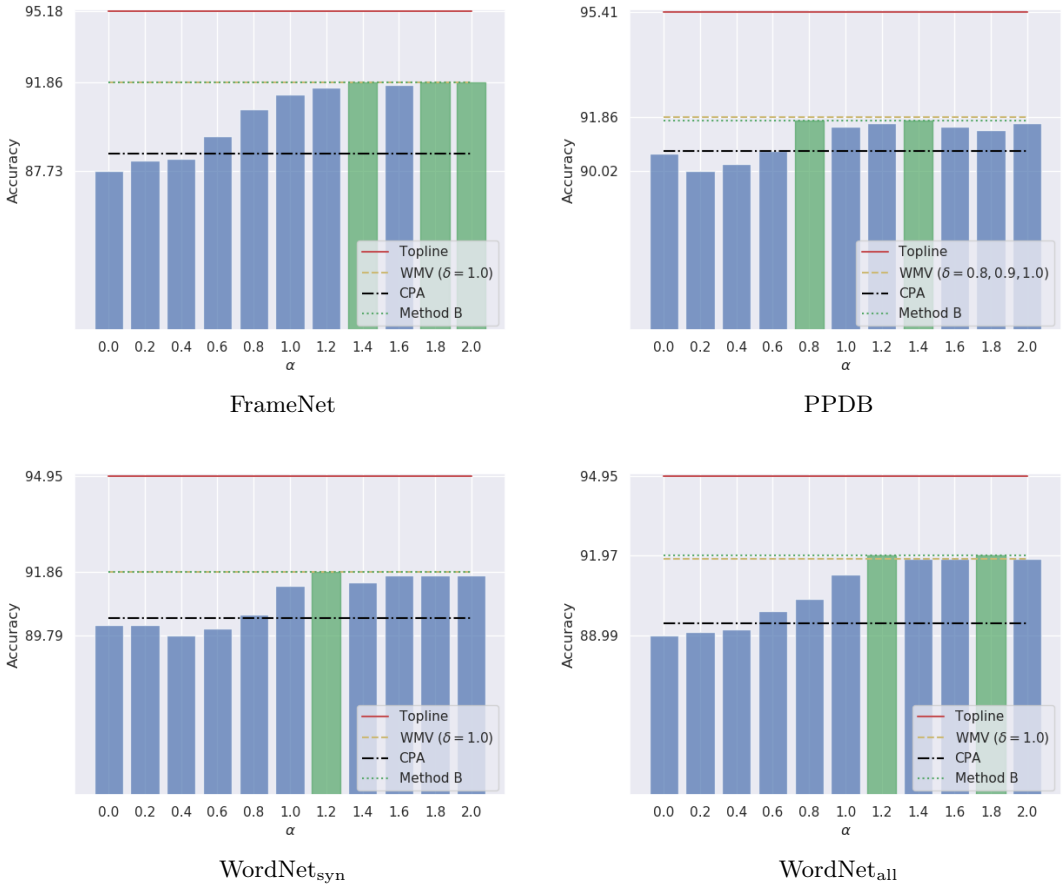


Figure A.28.: Grid search plots of accuracy scores for Method B while retrofitting after layer normalization at Transformer block 12 on SST-2.

B. Supplementary material from Chapter 4

B.1. Input encoding visualizations

Here we provide some visualizations related to the the impact of the input encoding for BERT as discussed in Section 4.4.1. All displayed results correspond to the mean accuracy achieved across 4 runs sorted in ascending order for each input encoding and each analogical relation (Section 4.3.2). In particular, Figure B.1 compares the focus characters with the entity markers and, Figure B.2 illustrates the cases where the embedding of the first entity marker is used to represent the hypernyms.

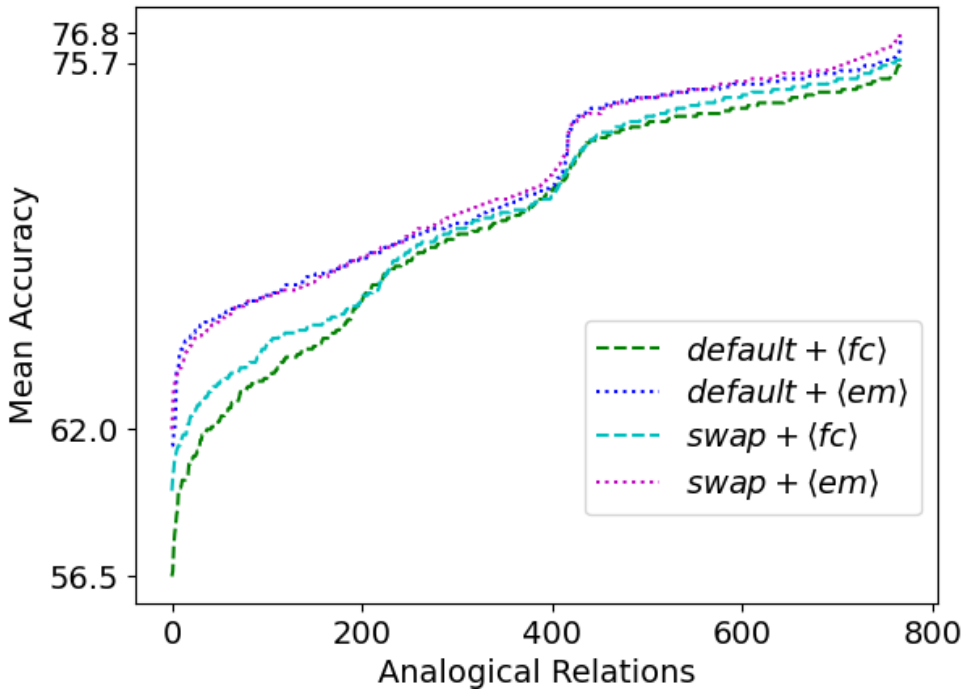


Figure B.1.: Comparison between enclosing the hypernyms with a focus character (⟨**fc**⟩) versus entity markers (⟨**em**⟩). The vertical axis represents the mean accuracy achieved on the development set, and the horizontal axis represents the 768 possible relations $A : B :: C : D$ sorted in order of increasing accuracy.

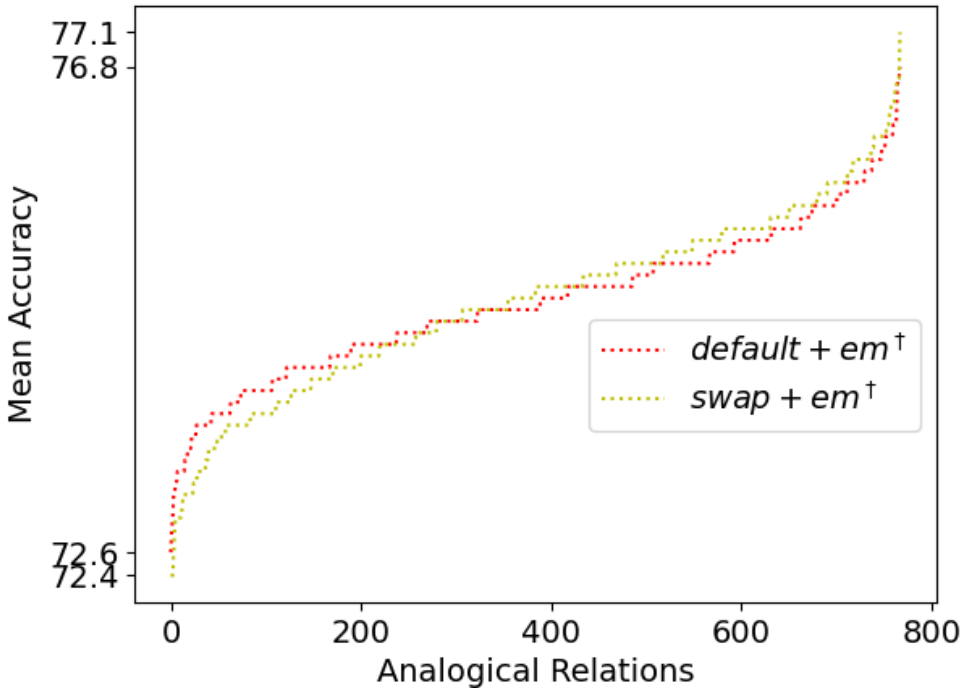


Figure B.2.: Comparison between swapping or not the hypernyms when the embedding of the first entity marker is used to represent them ($\langle$**em**$\rangle$). The vertical axis represents the mean accuracy achieved on the development set, and the horizontal axis represents the 768 possible relations $A : B :: C : D$ sorted in order of increasing accuracy.

B.2. Feature attribution maps

Here we provide some visualizations produced by the explanation methods related to the the importance of each token in the input sequence, and all 12 BERT Transformer blocks as discussed in Section 4.4.4. The visualizations correspond to the examples listed in Table B.1, where our HyperBertCLS baseline model correctly verifies the senses of the target word in the context, and the definition and the hypernyms, while AB4TSV fails, and vice versa.

Table B.1.: Samples from the WiC-TSV development set for which we apply the explanation methods. The target word in the context is highlighted in bold.

Context	**Definition**	**Hypernyms**	**Label**
A **fight** broke out at the hockey game.	The act of fighting any contest or struggle.	conflict, struggle, battle	True
The floats and the horses in the **parade** were impressive, but the marching bands were really amazing.	A visible display.	display, exhibit, showing	False
She felt a **tremor** in her stomach before going on stage.	A small earthquake.	earthquake, quake, temblor, seism	False
He had the **gem** set in a ring for his wife.	A person who is as brilliant and precious as a piece of jewelry.	person, individual, somebody, mortal, soul	False

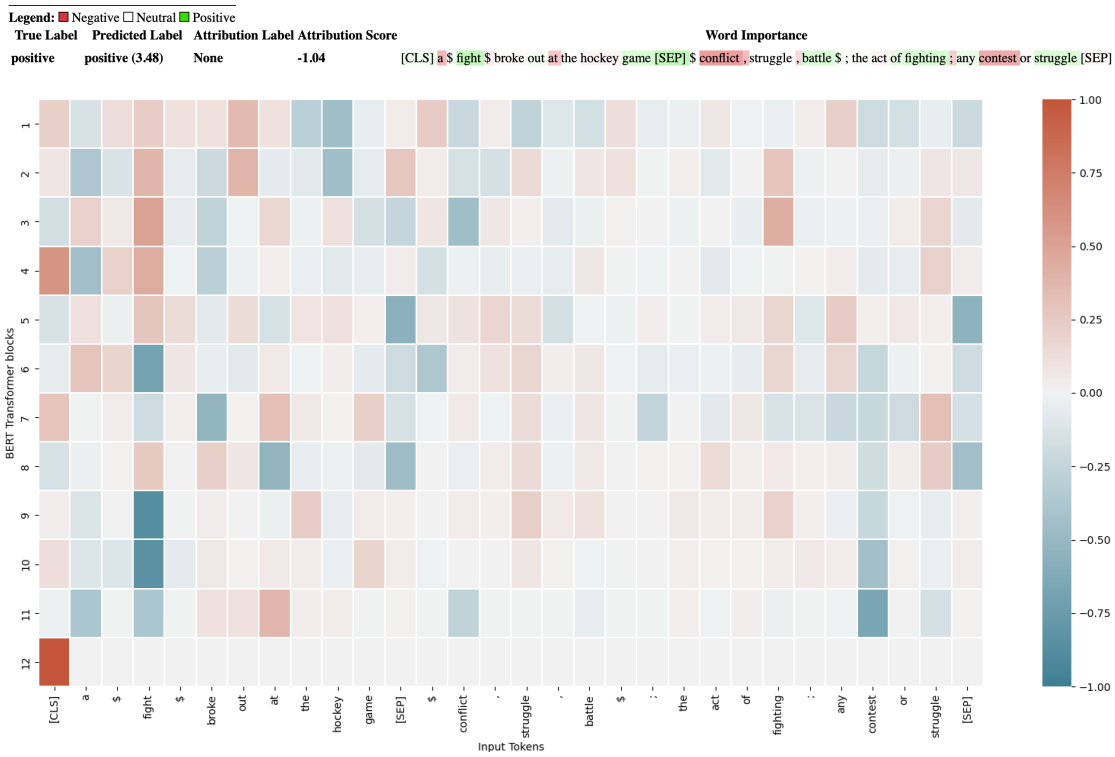


Figure B.3.: Saliency map of input token attributions (top) and heat map of attributions across all 12 Transformer blocks (bottom) of HyperBertCLS on a falsely predicted example.

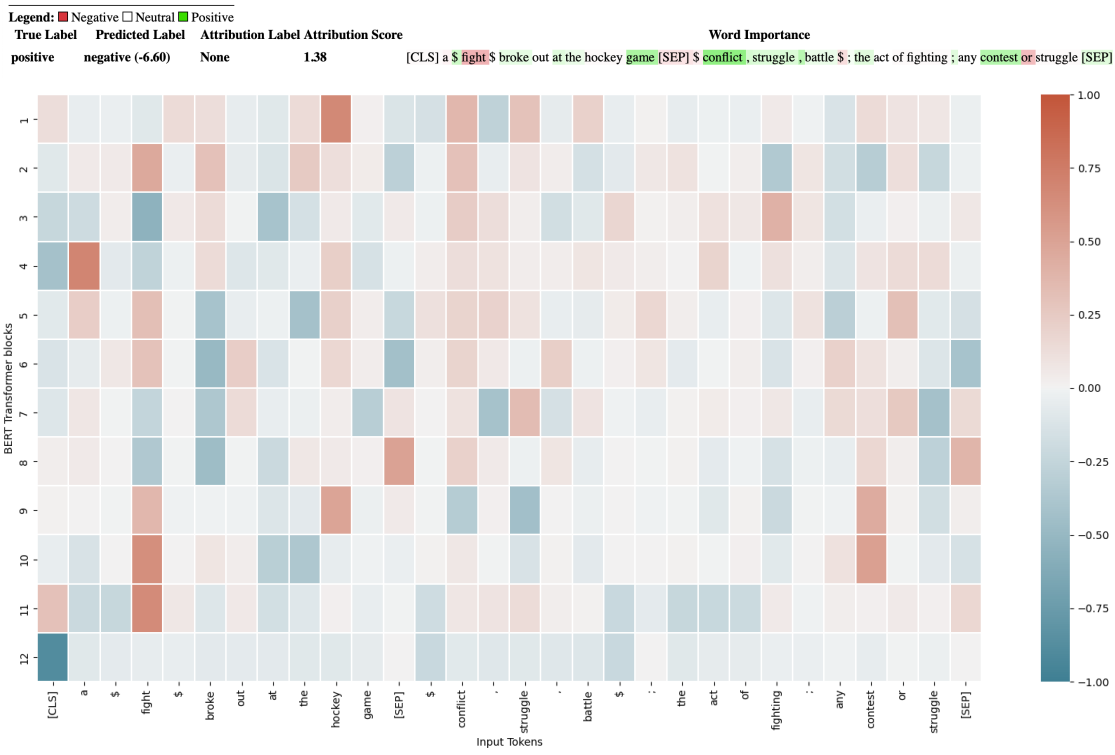


Figure B.4.: Saliency map of input token attributions (top) and heat map of attributions across all 12 Transformer blocks (bottom) of AB4TSV on a correctly predicted example.

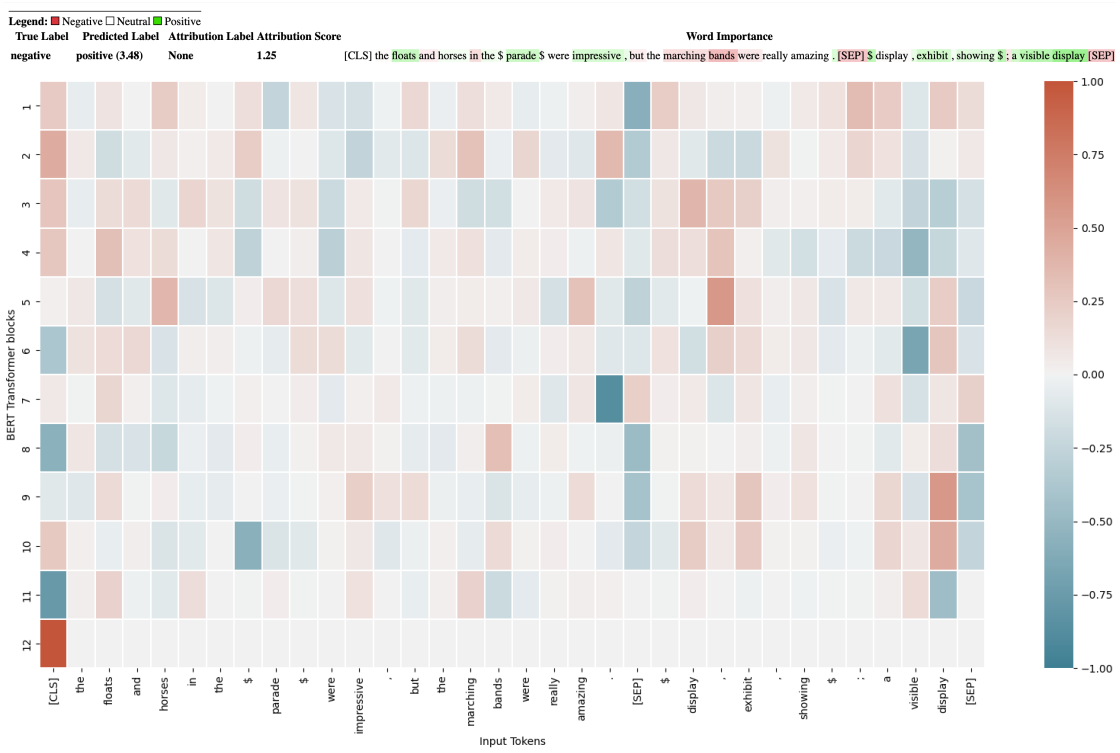


Figure B.5.: Saliency map of input token attributions (top) and heat map of attributions across all 12 Transformer blocks (bottom) of HyperBertCLS on a falsely predicted example.

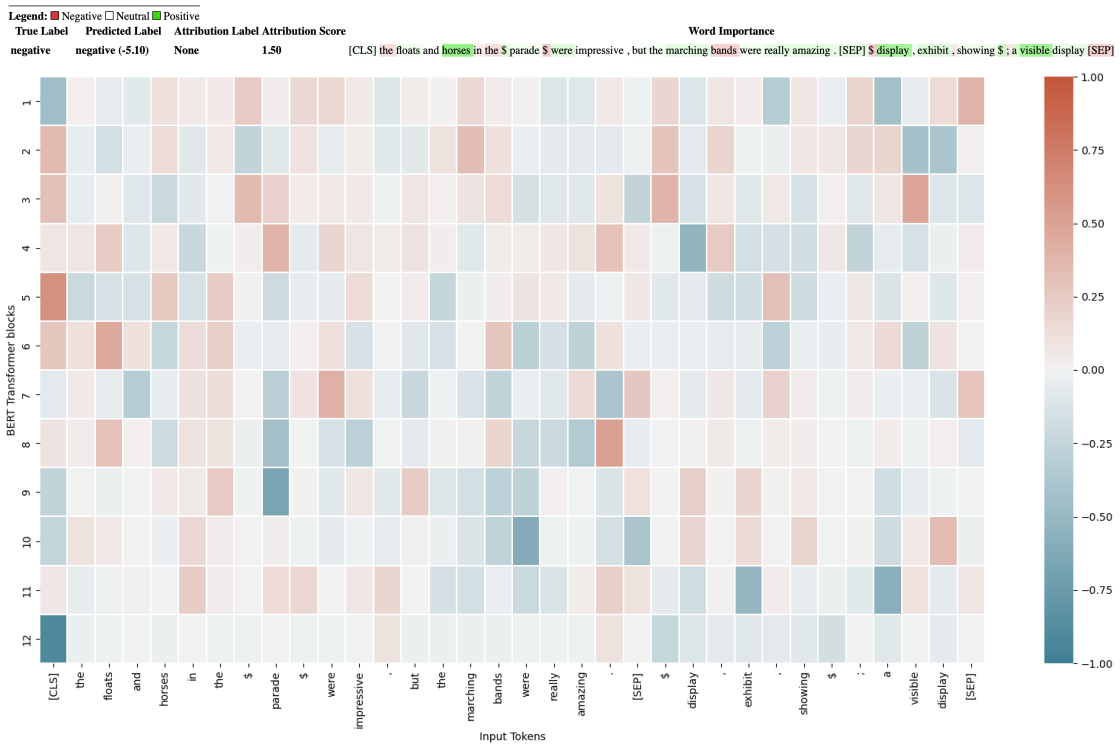


Figure B.6.: Saliency map of input token attributions (top) and heat map of attributions across all 12 Transformer blocks (bottom) of AB4TSV on a correctly predicted example.

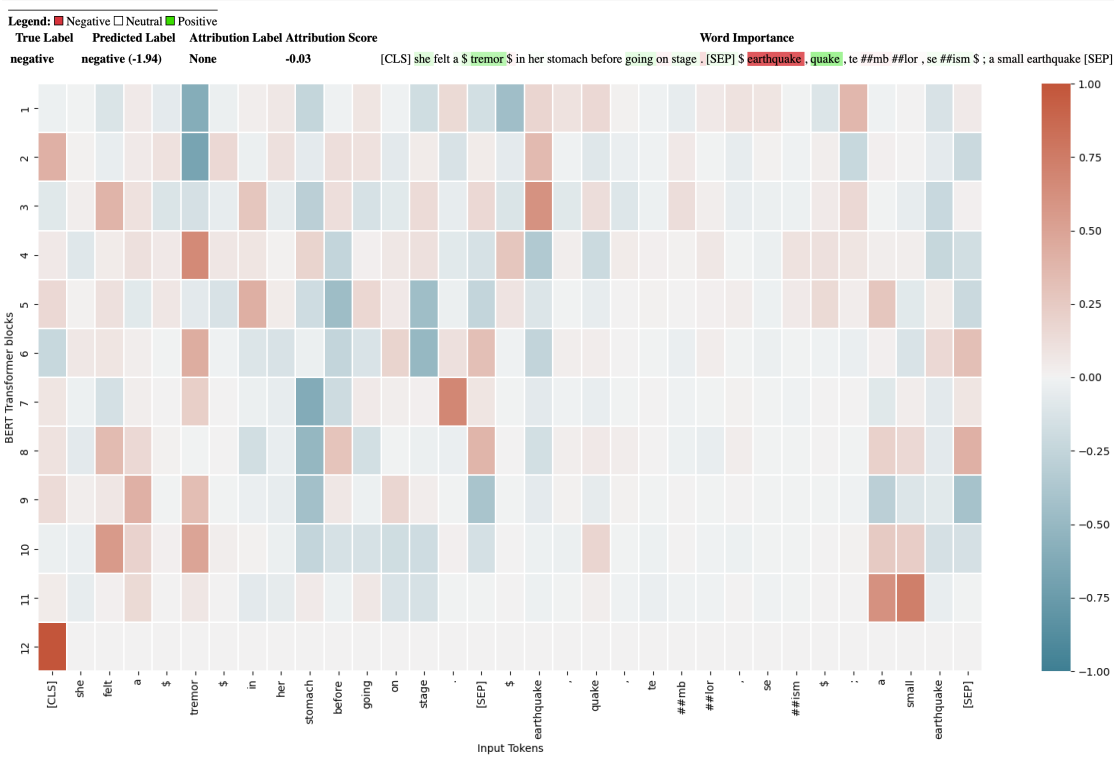


Figure B.7.: Saliency map of input token attributions (top) and heat map of attributions across all 12 Transformer blocks (bottom) of HyperBertCLS on a correctly predicted example.

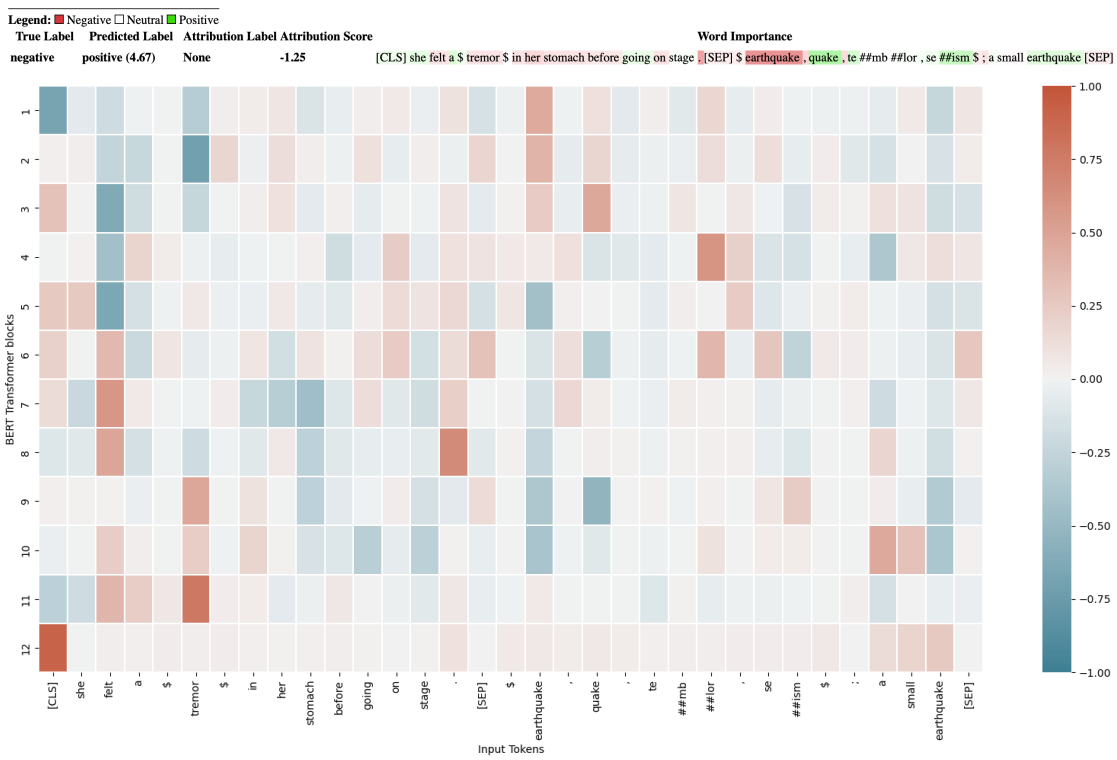


Figure B.8.: Saliency map of input token attributions (top) and heat map of attributions across all 12 Transformer blocks (bottom) of AB4TSV on a falsely predicted example.

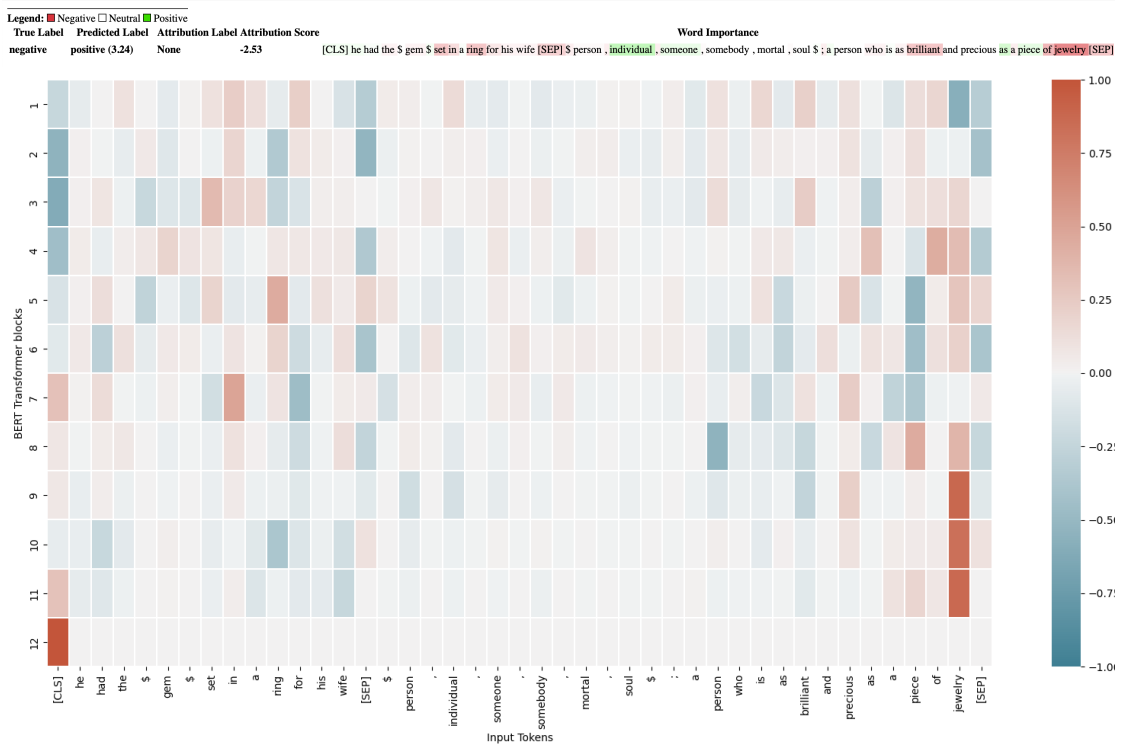


Figure B.9.: Saliency map of input token attributions (top) and heat map of attributions across all 12 Transformer blocks (bottom) of HyperBertCLS on a falsely predicted example.

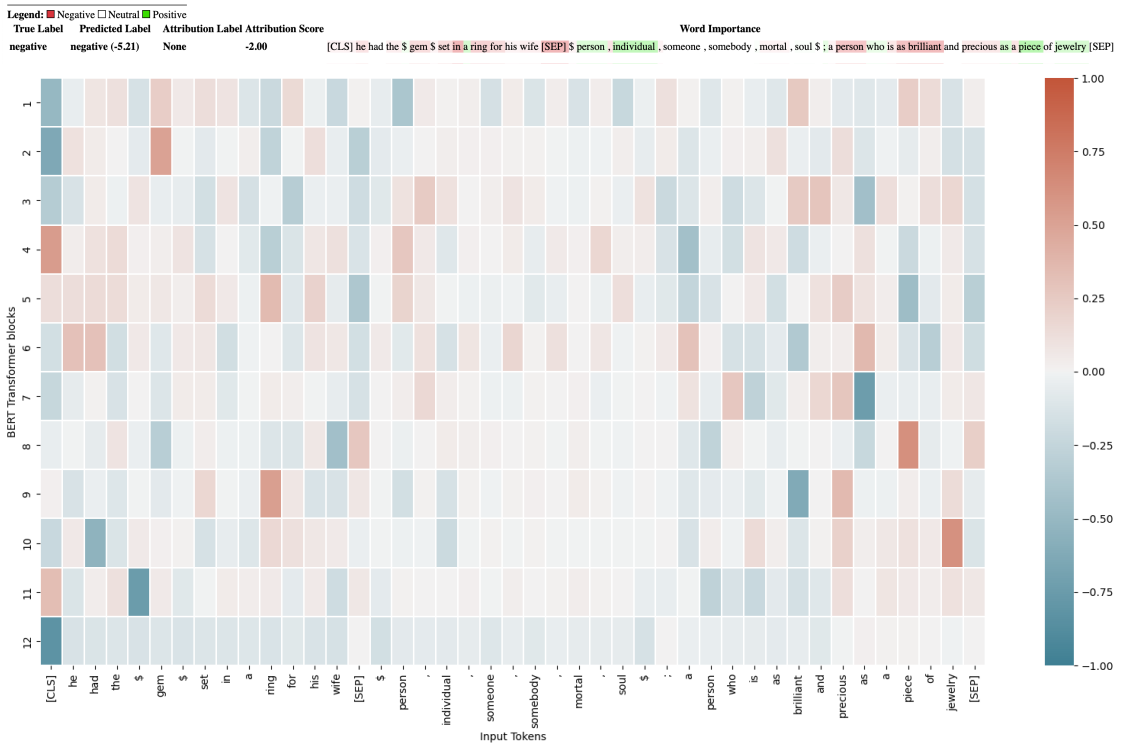


Figure B.10.: Saliency map of input token attributions (top) and heat map of attributions across all 12 Transformer blocks (bottom) of AB4TSV on a correctly predicted example.

C. Résumé étendu

C.1. Introduction

Le principal défi du traitement automatique des langues (TAL) est de créer des modèles capables de traiter et de représenter avec précision le langage naturel. Les progrès récents dans ce domaine ont permis de résoudre des problèmes du monde réel en automatisant des processus intermédiaires qui nécessitaient auparavant une expertise du domaine, du travail humain et beaucoup de temps. Par exemple, l'embauche et le recrutement de personnel pour une entreprise avec des milliers de candidats pour un seul poste, la traduction d'un texte d'une langue à une autre, la reconnaissance et la conversion de la demande de l'utilisateur en texte, la réponse à une question posée par un utilisateur, ne sont que quelques applications de TAL qui bénéficient de cet essor.

L'apprentissage profond permet d'aborder plus facilement les tâches de TAL par rapport aux approches traditionnelles qui nécessitent beaucoup d'ingénierie des caractéristiques. En principe, les ingrédients les plus importants sont l'acquisition de données et la sélection d'un algorithme d'apprentissage (modèle) approprié pour la tâche à accomplir. En général, le processus d'apprentissage est un problème d'optimisation dans lequel le but est d'optimiser une fonction objective. Pendant l'apprentissage, les paramètres du réseau sont ajustés tandis que la sortie du modèle est comparée à une valeur de référence. Une fois l'apprentissage terminé, les paramètres du modèle peuvent être directement appliqués à de nouvelles données. Dans l'exemple précédent du processus d'embauche et de recrutement, nous pouvons imaginer que les données d'apprentissage sont des milliers de formulaires de candidature pour différents postes, accompagnés d'annotations de caractéristiques importantes telles que le lieu, l'entreprise, etc. Un réseau de neurones de bout-en-bout pour la reconnaissance d'entités nommées est ensuite appris afin d'obtenir des représentations vectorielles des mots, également appelées *plongements*, et de prédire la classe d'entité correcte pour chaque mot parmi un ensemble de classes fixé.

Cependant, malgré leur succès, ces approches présentent de sérieuses limitations. Tout d'abord, elles fonctionnent en boîte noire : on ne sait pas comment le système arrive à une certaine conclusion pour la tâche à accomplir, ni pourquoi il suit un modèle spécifique de prise de décision. De plus, la capacité d'explication et d'interprétation est cruciale pour établir la confiance entre les humains et les systèmes, car dans de nombreux secteurs d'application tels que la santé, la justice ou les voitures autonomes, le risque est élevé. Il existe plusieurs méthodes qui améliorent l'*explicabilité* des modèles profonds en fournissant des preuves qui justifient leur comportement (Simonyan et al., 2014; Ribeiro et al., 2016; Lundberg and Lee, 2017; Shrikumar et al., 2017). Cette notion est différente de celle d'*interprétabilité* qui met en avant les propriétés intrinsèques du

modèle en mesurant à quel point la prédiction est facilement comprise par les humains (Li et al., 2022). Une autre question encore est liée à l'équité des réseaux de neurones profonds, comme la discrimination des femmes lors du processus d'embauche pour un poste de travail, ou l'association des personnes noires à un taux de criminalité futur plus élevé que celui des personnes blanches. Ce problème est lié à des erreurs dans le processus de collecte des données (Pessach and Shmueli, 2022). En outre, une attention particulière est accordée à la manière dont les modèles d'apprentissage profond traitent les questions de sécurité et de confidentialité. Il a été démontré que ces modèles peuvent être violés par diverses attaques visant à voler les paramètres du réseau, à déduire les données privées des utilisateurs, à compromettre les performances ou à manipuler les décisions du modèle (Liu et al., 2020b). Par exemple, les exemples adverses sont une situation typique où de petites perturbations dans les données peuvent produire des prédictions erronées avec un degré de confiance élevé. Ce phénomène est également lié à la *robustesse* du modèle, qui est mesurée comme la performance attendue sur des données inédites qui sont généralement générées en perturbant synthétiquement l'entrée, ou tirées d'une distribution différente (Wang et al., 2022). En plus, l'évolution de l'apprentissage profond nécessite de plus en plus de ressources informatiques, ce qui rend peu pratique le développement de leurs propres modèles par les praticiens. Enfin, leurs performances sont faibles en l'absence de données d'apprentissage suffisantes et souvent sur-spécialisées à des groupes cibles particuliers présents dans les données. C'est un problème si l'on considère que la collecte de données de bonne qualité est à la fois coûteuse et longue, et qu'elle contraste avec la façon dont les êtres humains raisonnent et apprennent — en traitant les concepts en relation les uns avec les autres plutôt que comme des unités séparées — ce qui permet d'appréhender de nouvelles connaissances avec seulement quelques exemples et peu de supervision.

Une étape vers la résolution des limitations susmentionnées consiste à exploiter les connaissances existantes en les intégrant dans le cadre de l'apprentissage profond. Selon la catégorisation de Deng et al. (2020), les connaissances sont divisées en deux grandes classes : les *connaissances générales* et les *connaissances du domaine*. Pour le reste de la thèse, nous nous concentrons sur les connaissances symboliques du domaine, car elles sont plus faciles à intégrer avec des modèles d'apprentissage profond que les connaissances générales, en raison de leur représentation formelle (symbolique). Quelques exemples sont les règles logiques, la logique du premier ordre, les expressions mathématiques telles que les équations et les inégalités, les règles probabilistes, etc. La structure la plus populaire pour la connaissance du domaine dans l'apprentissage profond est représentée sous la forme d'un graphe dont les nœuds codent les concepts d'intérêt et dont les arêtes désignent leurs attributs et leurs relations. Dans sa forme la plus générale, un graphe de connaissances constitue un ensemble de triplets qui suivent un modèle sujet-relation-objet.

Nous pensons que l'intégration de la connaissance du domaine et du raisonnement dans le cadre de l'apprentissage profond est un bon pas vers la résolution des diverses limitations que présentent les systèmes d'intelligence artificielle (IA) modernes. Par exemple, dans le contexte de la reconnaissance d'images, le fait que « les chats ont des

moustaches et de la fourrure » et que « les otaries ont des moustaches mais pas de fourrure » devrait aider le système à reconnaître les moustaches, même si les images d'apprentissage ne sont étiquetées qu'en termes de « chat » et d'« otarie ». Ensuite, l'apprentissage d'un classifieur « moustaches » sur des images d'animaux divers devrait permettre d'augmenter les performances de reconnaissance de ces deux animaux, mais aussi de découvrir d'autres animaux qui ne sont ni des chats ni des otaries. À cette fin, ce projet de thèse vise à développer et à évaluer des moyens d'intégrer différents types de connaissances et de raisonnements symboliques dans des systèmes de classification basés sur le modèle de langue pré-entraîné BERT (Devlin et al., 2019), dans le but de répondre aux questions suivantes :

- Comment reformater les connaissances du domaine pour les rendre compatibles avec les modèles de langue pré-entraînés basés sur BERT ?
- Comment incorporer efficacement les connaissances du domaine et le raisonnement dans les modèles de langue pré-entraînés basés sur BERT ?
- L'utilisation des connaissances du domaine et du raisonnement est-elle bénéfique en termes de performance pour une tâche particulière ?
- Les connaissances du domaine et le raisonnement pourraient-ils aider à augmenter la robustesse de modèles de langue pré-entraînés basés sur BERT ?

La suite de ce résumé est organisée comme suit. Les Parties C.2 et C.3 décrivent respectivement les Chapitres 3 et 4, et la Partie C.4 conclut ce résumé.

C.2. Adaptation des plongements contextuels de BERT grâce à des lexiques sémantiques

Les plongements lexicaux jouent un rôle fondamental dans de nombreuses applications du TAL. Il a été prouvé que l'exploitation de connaissances symboliques améliore la qualité des plongements et la performance de nombreuses tâches qui en découlent (Mrkšić et al., 2016, 2017; Ferret, 2017; Vulić et al., 2018; Lengerich et al., 2018).

Le *retrofitting* (Faruqui et al., 2015) est une technique pour adapter les plongements lexicaux distributionnels grâce aux relations issues d'un lexique sémantique. Étant donné un premier ensemble de plongements lexicaux distributionnels $\hat{Q} = (\hat{q}_1, \ldots, \hat{q}_n)$, l'objectif est d'obtenir de nouveaux plongements lexicaux $Q = (q_1, \ldots, q_n)$ pour les mêmes mots qui minimisent la fonction suivante :

$$\mathcal{L}(Q) = \sum_{i=1}^{n} \left[a_i ||q_i - \hat{q}_i||^2 + \sum_{(i,j)\in\mathcal{E}} b_{ij} ||q_i - q_j||^2 \right] \tag{C.1}$$

où $\mathcal{E}$ est l'ensemble des paires (i, j) de mots associés dans le graphe, et a_i et b_{ij} contrôlent la force des termes correspondants dans $\mathcal{L}$.

Dans le cadre de ce travail, nous visons à étendre le *retrofitting* aux plongements lexicaux contextuels.

C.2.1. Méthodes proposées d'adaptation des plongements lexicaux contextuels

Supposons un vocabulaire de mots $\mathcal{V} = \{w_1, \ldots, w_n\}$ et une ontologie Ω de relations sémantiques entre les mots de $\mathcal{V}$. On peut alors représenter Ω sous la forme d'un graphe non orienté $(\mathcal{V}, \mathcal{E})$, où les nœuds correspondent aux mots de $\mathcal{V}$ et les arêtes $(w_i, w_j) \in \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ aux relations sémantiques entre les nœuds. Supposons de plus que nous disposons d'un modèle de représentation contextuelle des mots $\mathcal{M}$, ainsi que d'un corpus d'apprentissage $\mathcal{D}_{\text{train}}$ sur lequel il est adapté et d'un corpus de test $\mathcal{D}_{\text{test}}$ sur lequel il est évalué pour une tâche particulière.

C.2.1.1. Méthode A

La méthode A combine le plongement contextuel d'un mot donné dans $\mathcal{D}_{\text{test}}$ avec les plongements contextuels de toutes les occurrences de tous les mots similaires dans $\mathcal{D}_{\text{train}}$.

Soit $\bar{q}_i \in \mathbb{R}^d$ le plongement contextuel du mot $w_i \in \mathcal{V}$ provenant de $\mathcal{M}$ pour une instance de test donnée[1]. On désigne par $\mathcal{J}_i$ l'ensemble des mots w_j qui sont adjacents à w_i selon Ω, et par $\mathcal{K}_j$ l'ensemble des instances d'apprentissage où w_j apparaît. Nous définissons alors $\hat{q}_{jk} \in \mathbb{R}^d$ comme le plongement contextuel calculé pour toutes les occurrences de w_j dans $\mathcal{D}_{\text{train}}$, avec l'indice $k \in \mathcal{K}_j$. Les ensembles d'indices $\mathcal{J}_i$ et $\mathcal{K}_j$ varient dynamiquement pour chaque mot.

Le but est d'apprendre un nouveau plongement q_i qui est proche de $\bar{q}_i$ et des nœuds adjacents dans Ω au sens de la norme $\mathcal{L}_2$ en minimisant

$$\mathcal{L}(q_i) = ||q_i - \bar{q}_i||^2 + \sum_{j \in \mathcal{J}_i} \sum_{k \in \mathcal{K}_j} b_{ijk} ||q_i - \hat{q}_{jk}||^2. \tag{C.2}$$

Les coefficients b_{ijk} doivent naturellement dépendre du nombre de voisins $|\mathcal{J}_i|$ de w_i, et du nombre d'occurrences $|\mathcal{K}_j|$ de chaque voisin w_j dans $\mathcal{D}_{\text{train}}$. À cet effet, nous les définissons comme $b_{ijk} = c_{ij} \times d_{jk} = \frac{1}{|\mathcal{J}_i|^\alpha} \cdot \frac{1}{|\mathcal{K}_j|^\beta}$, $\alpha, \beta \in [0, \infty)$ où c_{ij} contrôle la contribution de chaque voisin et d_{jk} contrôle la contribution de chacune de ses occurrences.

En mettant à zéro la dérivée de $\mathcal{L}$ par rapport à q_i et en exprimant la somme $\sum_k b_{ijk} \hat{q}_{jk}$ en fonction de la moyenne $\mu_{\hat{q}_j}$ de tous les $\hat{q}_{jk}$, on obtient la règle de mise à jour suivante :

$$q_i = \frac{\bar{q}_i + \sum_j \sum_k b_{ijk} \hat{q}_{jk}}{1 + \sum_j \sum_k b_{ijk}} = \frac{\bar{q}_i + |\mathcal{J}_i|^{-\alpha} \sum_j |\mathcal{K}_j|^{1-\beta} \mu_{\hat{q}_j}}{1 + |\mathcal{J}_i|^{-\alpha} \sum_j \mathcal{K}_j^{1-\beta}}. \tag{C.3}$$

C.2.1.2. Méthode B

La méthode B n'utilise pas du tout $\mathcal{D}_{\text{train}}$. Au lieu de cela, tout se passe au moment du test.

Là encore, nous utilisons $\mathcal{M}$ pour obtenir le plongement $\bar{q}_i$ du mot w_i pour une phrase spécifique dans $\mathcal{D}_{\text{test}}$. En outre, nous dérivons un plongement $\hat{q}_j$ pour chaque mot w_j qui

[1] Pour simplifier, $\bar{q}_i$ n'a pas d'exposant pour la phrase de test car nous ne traitons qu'une phrase de test à la fois.

est adjacent à w_i selon Ω. Pour ce faire, nous créons une nouvelle phrase en remplaçant w_i par w_j dans la phrase test, et nous répétons l'opération pour chaque nœud adjacent de w_i dans Ω.

L'objectif est d'apprendre un nouveau vecteur q_i qui est proche à la fois de $\bar{q}_i$ et de tous les $\hat{q}_j$ au sens de la norme $\mathcal{L}_2$ en minimisant

$$\mathcal{L}(q_i) = ||q_i - \bar{q}_i||^2 + \sum_{j \in \mathcal{J}_i} b_{ij} ||q_i - \hat{q}_j||^2. \tag{C.4}$$

De la même manière que ci-dessus, nous définissons les coefficients comme $b_{ij} = \frac{1}{|\mathcal{J}_i|^\alpha}$, $\alpha \in [0, \infty)$.

En mettant à zéro la dérivée de $\mathcal{L}$ par rapport à q_i et en exprimant la somme $\sum_j b_{ij}\hat{q}_j$ en fonction de la moyenne $\mu_{\hat{q}_j}$ de tous les $\hat{q}_j$, on obtient la règle de mise à jour suivante :

$$q_i = \frac{\bar{q}_i + \sum_j b_{ij}\hat{q}_j}{1 + \sum_j b_{ij}} = \frac{\bar{q}_i + |\mathcal{J}_i|^{1-\alpha}\mu_{\hat{q}_j}}{1 + |\mathcal{J}_i|^{1-\alpha}}. \tag{C.5}$$

C.2.2. Protocole expérimental

Tout d'abord, nous évaluons nos méthodes avec BlueBERT (Peng et al., 2019) pour l'extraction de relations biomédicales sur trois jeux de données biomédicales (ChemProt, DDI, i2b2 2010) issus du Biomedical Language Understanding Evaluation benchmark (Peng et al., 2019), à l'aide des deux lexiques sémantiques de verbes (*annotated* et *expanded clusters*) de Chiu et al. (2019a). Ensuite, nous les évaluons pour l'analyse du sentiment de critiques de films avec BERT (Devlin et al., 2019) sur le jeu de données SST-2 (Socher et al., 2013), en utilisant les mêmes lexiques semantiques (FrameNet, PPDB, WordNet, $\text{WordNet}_{\text{all}}$) que Faruqui et al. (2015), et le lexique de sentiments de Bing Liu (Hu and Liu, 2004) afin de réduire la portée des mots non pertinents.

Nous rapportons les performances pour chaque jeu de données en termes de F_1-score micro pour l'extraction de relations et de précision pour l'analyse des sentiments, comme Peng et al. (2019) et Wang et al. (2018), respectivement.

C.2.2.1. *Retrofitting* et architecture de BERT

BERT se compose de 12 blocs Transformer suivis d'une couche totalement connectée avec *dropout* et une fonction d'activation tanh. Chaque bloc contient une séquence de transformations divisée en couches. La couche de sortie de chaque bloc est constituée d'une transformation linéaire, suivie d'un *dropout* et d'un *layer normalization*. Nous supposons que l'impact de tout changement dans les plongements de BERT augmente à mesure que nous nous rapprochons de la sortie. Par conséquent, pour les deux approches, nous expérimentons quatre choix différents :

1. *retrofitting* avant *layer normalization* au bloc Transformer 11,
2. *retrofitting* après *layer normalization* au bloc Transformer 11,
3. *retrofitting* avant *layer normalization* au bloc Transformer 12,
4. *retrofitting* après *layer normalization* au bloc Transformer 12.

C.2.2.2. Optimisation de la recherche sur la grille

Afin de trouver les bonnes valeurs des hyperparamètres du *retrofitting* α, β, nous effectuons une recherche sur une grille de valeurs en utilisant les ensembles de développement. Pour l'extraction de relations biomédicales nous utilisons les *annotated* et *expanded clusters*, et nous recherchons α et β dans $[0, 2]$ avec un pas de 0,2 pour la méthode A, et α dans $[0, 2]$ avec un pas de 0,2 pour la méthode B. Pour l'analyse des sentiments des critiques de films, nous utilisons les quatre lexiques sémantiques en conjonction avec le lexique de sentiments de Bing Liu, et nous recherchons α dans $[0, 2]$ avec un pas de 0,2 pour la méthode B.

C.2.2.3. Stratégies de classification alternatives

Afin d'évaluer la capacité de notre méthode à exploiter les informations contenues dans les lexiques sémantiques, nous réalisons l'expérience suivante. Tout d'abord, nous augmentons chaque ensemble de données en ajoutant toutes les phrases modifiées qui sont générées en remplaçant le mot sous-jacent de la phrase originale par un mot voisin du lexique sémantique. Il en résulte un nombre de versions augmentées des ensembles de données originaux égal au nombre de lexiques sémantiques par tâche. Ensuite, nous utilisons BERT affiné sur la tâche en aval — BlueBERT pour l'extraction de relations biomédicales ou BERT pour l'analyse du sentiment de critiques de films — et évaluons les phrases originales et modifiées dans l'ensemble de données augmenté. Une fois que nous avons obtenu un score de probabilité pour chaque échantillon de l'ensemble de données augmenté, nous le comparons aux stratégies alternatives suivantes :

- **Topline** : Toujours sélectionner la vraie classe d'une phrase test comme prédiction finale, si elle a été prédite pour au moins une des phrases originales ou modifiées.
- **Weighted majority vote (WMV)** : Sélectionner la classe prédite ayant le plus d'occurrences comme prédiction finale parmi les phrases de test originales et modifiées. Ici, nous attribuons un poids de 1 à la phrase originale et un poids de $\frac{1}{|S|^\delta}, \delta \in [0, 1]$ à chaque phrase modifiée, où $|S|$ est le nombre total de phrases pour l'entrée de test actuelle.
- **Class posterior averaging (CPA)** : Calculer la moyenne des probabilités des classes prédites pour les phrases de test originales et modifiées, et prendre la classe avec la probabilité maximale comme prédiction finale.

C.2.3. Résultats et étude qualitative

Après avoir trouvé l'ensemble d'hyperparamètres le plus performant sur l'ensemble de développement, nous évaluons le modèle correspondant sur les données de test. Les résultats des deux méthodes sont affichés dans le Tableau C.1.

Les méthodes proposées n'ont pas d'impact substantiel sur les performances de ces tâches, nous effectuons donc une analyse qualitative pour mieux comprendre ce résultat négatif. Les résultats de la recherche sur grille indiquent que la performance est plus sensible au choix des valeurs des hyperparamètres lorsque nous rétrofittons plus près de

Table C.1.: Performances obtenues par les méthodes A et B, les stratégies de classification alternatives, et d'autres approches de *retrofitting*. La méthode Baseline correspond à BERT affiné sur chaque ensemble de données pour la tâche visée.

Corpus	**Modèle**	**Lexique sémantique**	**Développement** miF_1/Acc (%)	**Test** miF_1/Acc (%)
ChemProt	Baseline	—	74,47	72,61
	Méthode A	expanded-16	74,86	72,56
	Méthode B	annotated-50	74,59	72,63
	Topline	annotated-50	75,54	73,67
	CPA	annotated-50	72,92	72,07
	WMV ($\delta = 1, 0$)	annotated-50	74,47	72,61
	Chiu et al. (2019a)	expanded-34	—	71,00
DDI	Baseline	—	71,34	80,11
	Méthode A	expanded-34	79,35	78,78
	Méthode B	annotated-34	72,33	79,43
	Topline	annotated-34	73,04	80,97
	CPA	annotated-34	71,97	79,40
	WMV ($\delta = 0, 1$)	annotated-34	72,02	79,60
i2b2-2010	Baseline	—	71,34	72,69
	Méthode A	expanded-16	72,92	72,52
	Méthode B	annotated-34	71,83	72,63
	Topline	annotated-34	73,71	74,18
	CPA	annotated-34	60,79	58,50
	WMV ($\delta = 1, 0$)	annotated-34	71,34	72,69
SST-2	Baseline	—	91,86	92,00
	Méthode B	$\text{WordNet}_{\text{syn}}$	92,09	92,11
	Topline	$\text{WordNet}_{\text{syn}}$	94,95	94,55
	CPA	$\text{WordNet}_{\text{syn}}$	90,37	90,11
	WMV ($\delta = 1, 0$)	$\text{WordNet}_{\text{syn}}$	91,86	92,00
	Faruqui et al. (2015)	$\text{WordNet}_{\text{syn}}$	—	82,40

la couche de sortie de BERT. Après avoir mis en œuvre une heuristique de choix des des valeurs des hyperparamètres basée sur la performance au voisinage des valeurs choisies, nous observons que les performances sont contrastées entre les ensembles de données, ce qui suggère que le choix optimal de ces paramètres dépend de l'ensemble de données. En plus, les stratégies de classification alternatives proposées montrent que l'utilisation des lexiques peut améliorer de manière significative les performances pour toutes les tâches et tous les ensembles de données. En outre, le classement des distances euclidiennes entre un sous-ensemble de vecteurs avant et après l'opération de *retrofitting* révèle qu'une grande partie de ces vecteurs ne se déplacent pas de manière substantielle ou se déplacent trop

loin par rapport à leur position initiale dans l'espace latent. Par conséquent, il y a beaucoup de variation entre les plongements des mots voisins et donc tous les mots du lexique ne sont pas pertinents pour les tâches à accomplir. Pour résoudre ce problème, nous nous appuyons sur le test statistique de McNemar et réduisons la taille des lexiques en sélectionnant uniquement les voisins pertinents pour un mot donné, avec des niveaux de confiance variables. Cette expérience montre que le *retrofitting* peut être significatif, mais que la taille des lexiques réduits qui en résultent n'est pas suffisamment grande pour pouvoir être généralisée au moment du test. Enfin, nous démontrons que notre méthode préserve le vote majoritaire, ce qui implique que le *retrofitting* a du potentiel à condition que le lexique puisse aider.

C.3. Vérification du sens d'un mot par une approche basée sur l'analogie

Malgré leur grand succès pour une pléthore de tâches, la capacité des modèles de langue pré-entraînés à effectuer des raisonnements est limitée et peu étudiée (Talmor et al., 2020; Rogers et al., 2020). Le raisonnement analogique est l'une des approches d'inférence les plus utilisées dans la vie quotidienne et on le retrouve dans de nombreuses applications du TAL telles que la traduction automatique (Langlais et al., 2009), la réponse à des questions visuelles (Peyre et al., 2019), la résolution de problèmes sémantiques (Lim et al., 2019) et morphologiques (Alsaidi et al., 2021a).

Pour résoudre les problèmes basés sur l'analogie, le système doit apprendre à raisonner sur des relations de la forme $A : B :: C : D$, ce qui se traduit par *« A est à B ce que C est à D »*. En suivant l'axiomatisation de Lepage (2004), une relation quaternaire constitue une proportion analogique si et seulement si les propriétés suivantes sont vraies : $\forall A,\ B,\ C,\ D$,

1. $A : B :: C : D \Rightarrow C : D :: A : B$ (symétrie)
2. $A : B :: C : D \Rightarrow A : C :: B : D$ (permutation centrale).

C.3.1. Formulation du problème

La tâche de vérification du sens d'un mot (*target sense verification* ou TSV) (Breit et al., 2021) est une tâche de désambiguïsation dans laquelle le système dispose d'un mot cible en contexte d'une part et d'une définition et d'un ensemble d'hypernymes de ce mot d'autre part, et doit décider si leurs sens correspondent ou non. Nous proposons une manière de formuler la tâche de TSV comme une tâche de détection d'analogie, en transformant les données d'entrée en quadruplets de la forme $A : B :: C : D$, comme le montre la Figure C.1.

C.3.2. Architecture AB4TSV

Suivant Lim et al. (2019) et Alsaidi et al. (2021a), nous utilisons un classifieur par réseau de neurones convolutif (CNN) qui modélise explicitement les relations *« est à »*

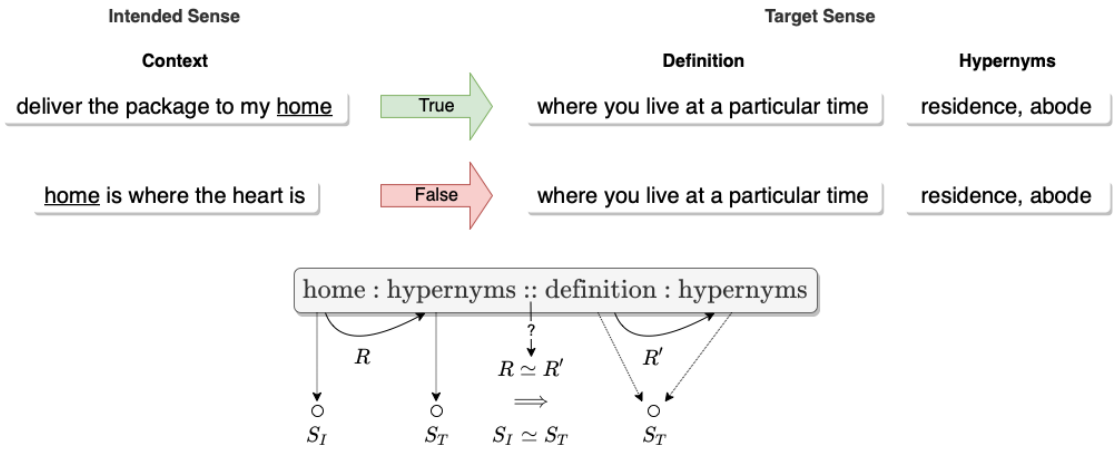


Figure C.1.: Illustration de la traduction de TSV en détection d'analogie.

et *« ce que »* dans l'analogie. L'architecture du modèle est composée de deux parties principales. Tout d'abord, nous encodons les mots ou les syntagmes à comparer (mot cible, contexte, définition, hypernymes, etc.) en 4 plongements $\mathbf{A}$, $\mathbf{B}, \mathbf{C}, \mathbf{D}$ en utilisant la couche d'encodage finale de BERT. Ensuite, nous les concaténons en une matrice de taille $n \times 4$, où n est la taille de plongement. Cette matrice sert d'entrée à un classifieur CNN qui se compose des couches suivantes :

- Une couche convolutive avec 128 filtres de taille 1×2 avec *stride* $(1, 2)$ et activation ReLU. La sortie de cette opération est une matrice de taille $128 \times n \times 2$. Intuitivement, cette couche modélise le *« est à »* dans *« A est à B »* et *« C est à D »*.
- Une couche convolutive avec 64 filtres de taille 2×2 avec *stride* $(2, 2)$ et activation ReLU. La sortie de cette opération est aplatie en un vecteur de longueur $64 \times (n-1)$. Intuitivement, cette couche modélise la relation *« ce que »* dans *« A est à B ce que C est à D »*.
- Une couche totalement connectée avec une activation sigmoïde, résultant en une sortie scalaire.

L'architecture *Analogy and BERT for TSV* (AB4TSV) proposée est représentée sur la Figure C.2.

C.3.3. Choix de la relation analogique et du codage d'entrée

Pour résoudre la tâche de TSV en se basant sur le raisonnement analogique, il faut sélectionner A, B, C, $D \in S$ de sorte que la relation $A : B :: C : D$ donne de bonnes performances de classification. On définit $S = \{cls, tgt, ctx, def, hyps, descr\}$ l'ensemble des tokens[2] qui peuvent être obtenus à partir de BERT comme suit : cls correspond au token `[CLS]`, tgt désigne le mot cible dans le contexte, ctx inclut tous les tokens dans le

[2]Nous indiquons en gras les plongements correspondants.

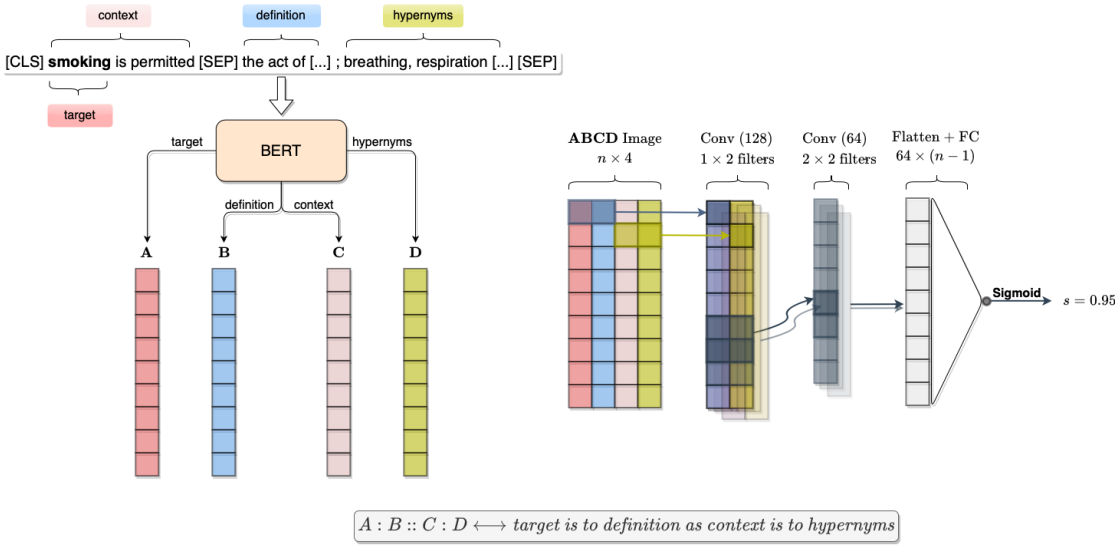


Figure C.2.: Vue d'ensemble de AB4TSV.

contexte, *def* représente tous les tokens dans la définition, *hyps* contient tous les tokens dans les hypernymes, et *descr* est la concaténation de tous les tokens dans la définition et les hypernymes.

Le format d'encodage d'entrée par défaut (Breit et al., 2021) structure la séquence d'entrée de BERT en une paire de phrases, de sorte que le contexte vient en premier, et la concaténation de la définition et des hypernymes dans la deuxième phrase, comme illustré en haut à gauche de la Figure C.2. En outre, nous présentons des méthodes alternatives d'encodage de la définition et des hypernymes dans des plongements basés sur les opérations suivantes :

- **swap** : échange de la position de la définition et des hypernymes dans l'encodage d'entrée par défaut ;
- **fc** : en entourant les hypernyms de caractères de focus, par exemple, *résidence, domicile* devient *$ résidence, domicile $* ;
- **em** : en entourant les hypernymes de marqueurs d'entité, par exemple, *résidence, domicile* devient *[H] résidence, domicile [/H]*.

Sur la base de ces trois opérations, nous testons les 6 encodages d'entrée suivants :

1. **default** :
   `[CLS] contexte [SEP] définition ; hypernymes [SEP].`
2. **default+fc** :
   `[CLS] contexte [SEP] définition ; $ hypernymes $ [SEP]`
3. **default+em** :
   `[CLS] contexte [SEP] définition ; [H] hypernymes [/H] [SEP].`
4. **swap** :
   `[CLS] contexte [SEP] hypernymes ; définition [SEP]`
5. **swap+fc** :

`[CLS] contexte [SEP] $ hypernymes $ ; définition [SEP]`

6. **swap+em** :
   `[CLS] contexte [SEP] [H] hypernymes [/H] ; définition [SEP]`

C.3.4. Configuration expérimentale

Nous évaluons AB4TSV sur le jeu de données WiC-TSV (Word in Context - Target Sense Verification), qui a été conçu spécifiquement pour la tâche de TSV (Breit et al., 2021).

Il y a $|S|^4 = 6^4 = 1\,296$ combinaisons possibles de $A : B :: C : D$ au total. Pour s'assurer que nous testons si le sens du mot cible dans le contexte correspond au sens cible dans la définition et les hypernymes, nous distinguons d'abord deux ensembles : $S_1 = \{cls, tgt, ctx\}$ et $S_2 = \{cls, def, hyps, descr\}$ où $S_1 \cup S_2 = S$. Le premier ensemble comprend les plongements qui contiennent principalement des informations provenant du contexte, tandis que le second contient les plongements qui reflètent les informations trouvées dans la définition et les hypernymes. cls appartient aux deux puisqu'il représente la totalité de l'information d'entrée. Ensuite, nous définissons les règles suivantes : pour tous les $A,\ B,\ C,\ D$ dans S,

1. $A \neq B \vee C \neq D$
2. $\neg\Big[\big[(A \in S_1 \setminus S_2) \wedge (B, C, D \in S_1)\big] \vee \big[(A \in S_2 \setminus S_1) \wedge (B, C, D \in S_2)\big]\Big]$

La première règle garantit que nous évitons les relations où les plongements de chaque côté sont identiques, et la deuxième règle garantit que $A,\ B,\ C,\ D$ ne peuvent pas être obtenus à partir de S_1 ou S_2 exclusivement. Ces règles réduisent le nombre de relations à tester à 768. Pour chaque choix d'encodage d'entrée et de relation, nous entraînons notre système 4 fois en utilisant différentes graines aléatoires, ce qui donne $6 \times 768 \times 4 = 18\,432$ exécutions. Une seule exécution prend environ 35 minutes sur une Nvidia GTX 1080 Ti 11GB.

C.3.5. Évaluer et promouvoir l'invariance de permutation des proportions analogiques

Une partie essentielle de nos expériences consiste à utiliser les propriétés d'invariance de permutation des proportions analogiques pour évaluer (i) si le raisonnement analogique est bénéfique en termes de performance dans la tâche, et (ii) si le modèle apprend naturellement à être invariant à ces permutations ou si elles doivent être explicitement appliquées au moment de l'apprentissage. Pour (ii), nous calculons les plongements $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ de la relation donnée à tester, et comparons simplement les performances obtenues en alimentant le classifieur avec la relation initiale par rapport aux relations obtenues en permutant les proportions analogiques. Pour (i), nous incluons à la fois les relations initiales et les relations permutées dans chaque *minibatch*. Nous distinguons les apprentissages sans et avec invariance de permutation en ajoutant l'indice « pi », c'est-à-dire AB4TSV et AB4TSV$_{\text{pi}}$, respectivement.

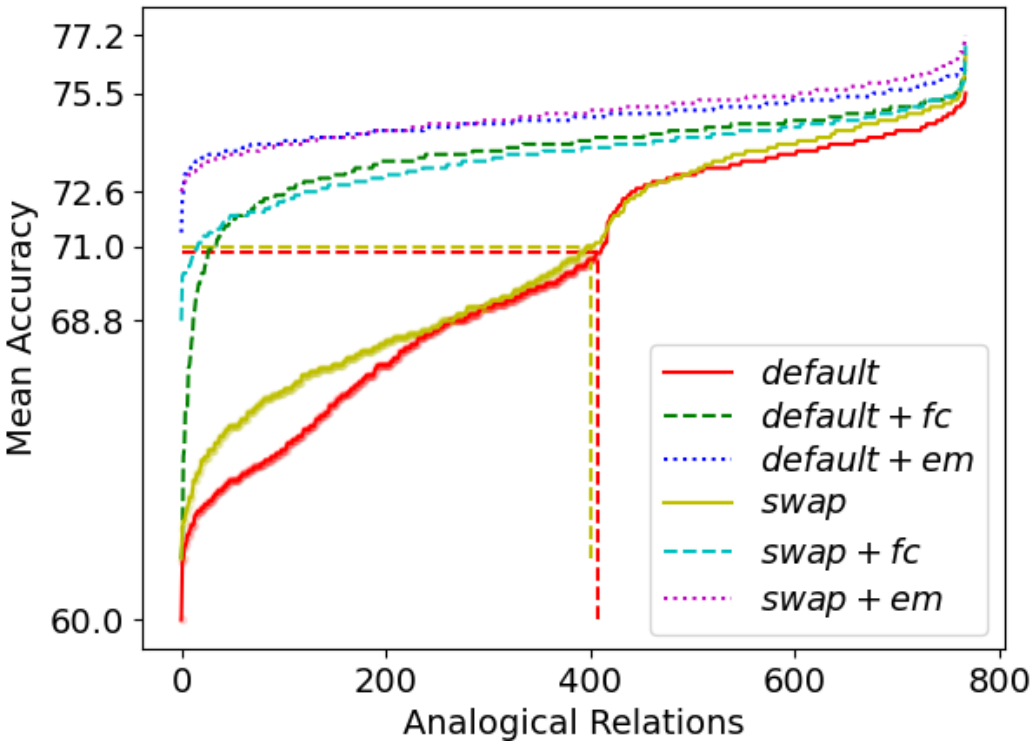


Figure C.3.: Précision moyenne obtenue sur l'ensemble de développement. Chaque courbe représente un codage d'entrée possible, et l'axe horizontal représente les 768 relations possibles $A : B :: C : D$ triées par ordre de précision croissante. La zone en surbrillance montre la partie des relations analogiques qui donnent de mauvaises performances pour les codages d'entrée **default** et **swap**.

C.3.6. Résultats

Les résultats de la Figure C.3 montrent que pour certains encodages d'entrée, la représentation des hypernymes peut dégrader la performance finale. Cependant, ce problème n'est pas présent lorsque nous entourons les hypernymes avec des caractères de focus ou des marqueurs d'entité. La répétition de la même expérience d'optimisation avec des encodages d'entrée supplémentaires, montre qu'il n'est pas suffisant d'inclure les hypernymes avec des caractères de focus ou des marqueurs d'entité au niveau de l'entrée. Ce qui fait vraiment la différence, c'est l'inclusion du plongement de ces caractères spéciaux dans le calcul du plongement des hypernymes.

Après avoir incorporé la symétrie et la permutation centrale avec AB4TSV, les résultats montrent une légère diminution des performances par rapport au modèle entraîné sans elles, de l'ordre de 1% dans l'absolu. Cependant, ils sont cohérents pour tous les encodages d'entrée. Par conséquent, AB4TSV gagne en robustesse puisqu'il ne dépend

plus du choix de l'encodage d'entrée. De plus, après avoir donné en entrée les quadruplets de symétrie et de permutation centrale au classifieur d'analogie, nous remarquons que AB4TSV n'est pas naturellement invariant par rapport à ces propriétés, mais qu'elles doivent être explicitement appliquées pendant l'apprentissage.

Ensuite, nous testons notre système sur le benchmark d'évaluation WiC-TSV, et nous montrons qu'il peut surpasser les approches existantes en termes de précision et de score F1, sans tenir compte de l'application ou non des propriétés axiomatiques des analogies soient appliquées ou non au moment de l'apprentissage. Les résultats sont affichés dans le Tableau C.2.

Table C.2.: Résultats des tests de nos systèmes les plus performants entraînés avec et sans les permutations des proportions analogiques, comparés aux résultats précédemment rapportés. Tous les résultats sont calculés par les auteurs du benchmark WiC-TSV (Breit et al., 2021).

Approach	**Test** *Acc*	**Test** *F*1
Supervised		
CTLR (Moreno et al., 2021)	78,3	78,5
Vandenbussche et al. (2021)	71,9	76,2
BERT-B (Breit et al., 2021)	76,6	78,2
BERT-L (Breit et al., 2021)	76,3	77,8
FastText (Breit et al., 2021)	53,4	63,4
AB4TSV+swap+em	75,7	77,5
AB4TSV+swap+fc	**78,6**	**79,8**
AB4TSV$_{pi}$+default+em	**78,6**	79,4
Unsupervised		
U-dBERT (Breit et al., 2021)	61,2	51,3
U-BERT (Breit et al., 2021)	60,5	51,9
MIRRORWIC (Liu et al., 2021)	73,7	—

Enfin, les méthodes d'attribution des caractéristiques révèlent que le plongement `[CLS]` reçoit toujours une attribution extrêmement positive ou négative, ce qui signifie qu'il s'agit d'un facteur décisif pour la prédiction finale. Bien que le reste des tokens obtiennent une attribution quasi-nulle pour notre modèle de base BERT, AB4TSV attribue des attributions non triviales aux composants définis par le quadruplet d'entrée.

C.4. Conclusion

En résumé, dans cette thèse, nous avons tenté d'incorporer des connaissances et des raisonnements symboliques aux modèles de type BERT afin d'améliorer leurs performances sur des tâches en aval et d'augmenter leur robustesse. Pour le premier objectif,

nous avons conçu deux méthodes de *retrofitting* qui injectent des informations provenant de lexiques sémantiques dans les plongements de BERT, et nous les avons évaluées sur l'extraction de relations biomédicales et l'analyse de sentiments de critiques de films. Les résultats de l'analyse qualitative montrent que les opérations de *retrofitting* ont le potentiel d'améliorer les résultats, cependant des mécanismes plus sophistiqués de sélection des voisins pertinents pour un mot donné dans les lexiques sémantiques sont nécessaires, afin d'éviter d'intégrer des informations bruitées dans les plongements. Pour ce dernier objectif, nous avons formulé la tâche de TSV comme un problème de détection d'analogie, et nous avons proposé AB4TSV, une architecture hybride qui a obtenu des résultats compétitifs sur le benchmark d'évaluation WiC-TSV. Après avoir inclus les propriétés axiomatiques des analogies dans la phase d'apprentissage du modèle, nous avons montré qu'il conserve une bonne performance, et gagne néanmoins en robustesse, car il devient moins dépendant de la sélection de l'encodage d'entrée. Nous croyons fermement que la combinaison de la connaissance et du raisonnement symboliques avec les modèles de langue pré-entraînés est la voie à suivre pour répondre aux diverses limitations que présentent les systèmes d'IA modernes, et nous espérons que cette thèse fournira du matériel utile et encouragera la poursuite des recherches sur ce sujet.

Bibliography

Afantenos, S., Kunze, T., Lim, S., Prade, H., and Richard, G. (2021). Analogies between sentences: theoretical aspects - preliminary experiments. In *Proceedings of the 16th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, pages 3–18.

Aghajanyan, A., Gupta, A., Shrivastava, A., Chen, X., Zettlemoyer, L., and Gupta, S. (2021). Muppet: Massive multi-task representations with pre-finetuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5799–5811.

Ahmed, A., Shervashidze, N., Narayanamurthy, S., Josifovski, V., and Smola, A. J. (2013). Distributed large-scale natural graph factorization. In *Proceedings of the 22nd International Conference on the World Wide Web (WWW)*, pages 37–48.

Alimova, I., Tutubalina, E., and Nikolenko, S. I. (2021). Cross-domain limitations of neural models on biomedical relation classification. *IEEE Access*, 10:1432–1439.

Alsaidi, S., Decker, A., Lay, P., Marquer, E., Murena, P.-A., and Couceiro, M. (2021a). A neural approach for detecting morphological analogies. In *Proceedings of the 2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10.

Alsaidi, S., Decker, A., Lay, P., Marquer, E., Murena, P.-A., and Couceiro, M. (2021b). On the transferability of neural models of morphological analogies. In *Proceedings of ECML PKDD Workshop on Advances in Interpretable Machine Learning and Artificial Intelligence (AIMLAI)*, pages 76–89.

Ammar, W., Groeneveld, D., Bhagavatula, C., Beltagy, I., Crawford, M., Downey, D., Dunkelberger, J., Elgohary, A., Feldman, S., Ha, V., Kinney, R., Kohlmeier, S., Lo, K., Murray, T., Ooi, H.-H., Peters, M., Power, J., Skjonsberg, S., Wang, L. L., Wilhelm, C., Yuan, Z., van Zuylen, M., and Etzioni, O. (2018). Construction of the literature graph in semantic scholar. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 84–91.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. In *Proceedings of NIPS 2016 Deep Learning Symposium*.

Bach, S. H., Broecheler, M., Huang, B., and Getoor, L. (2017). Hinge-loss Markov random fields and probabilistic soft logic. *Journal of Machine Learning Research (JMLR)*, 18(109):1–67.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Baker, C. F., Fillmore, C. J., and Lowe, J. B. (1998). The Berkeley FrameNet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics (ACL-COLING)*, pages 86–90.

Baldini Soares, L., FitzGerald, N., Ling, J., and Kwiatkowski, T. (2019). Matching the blanks: Distributional similarity for relation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2895–2905.

Bannard, C. and Callison-Burch, C. (2005). Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 597–604.

Baroni, M., Dinu, G., and Kruszewski, G. (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 (ACL)*, pages 238–247.

Belkin, M. and Niyogi, P. (2001). Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Proceedings of the 14th International Conference on Neural Information Processing Systems (NIPS)*, pages 585–591.

Beltagy, I., Lo, K., and Cohan, A. (2019). SciBERT: A pretrained language model for scientific text. 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 3615–3620.

Bengio, Y., Ducharme, R., and Vincent, P. (2003). A neural probabilistic language model. *Journal of Machine Learning Research (JMLR)*, 13:1137–1155.

Bhagat, S., Cormode, G., and Muthukrishnan, S. (2011). Node classification in social networks. In *Social Network Data Analytics*, pages 115–148. Springer US.

Bihani, G. and Rayz, J. (2021). Low anisotropy sense retrofitting (LASeR) : Towards isotropic and sense enriched representations. In *Proceedings of Deep Learning Inside Out (DeeLIO): The 2nd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 81–95.

Boag, W. and Kané, H. (2017). AWE-CM vectors: Augmenting word embeddings with a clinical metathesaurus. In *Proceedings of NIPS 2017 Workshop on Machine Learning for Health (ML4H)*.

Bodenreider, O. (2004). The unified medical language system (UMLS): integrating biomedical terminology. *Nucleic Acids Research*, 32(suppl_1):D267–D270.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics (TACL)*, 5:135–146.

Bosselut, A., Bras, R. L., , and Choi, Y. (2021). Dynamic neuro-symbolic knowledge graph construction for zero-shot commonsense question answering. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*, pages 4923–4931.

Bosselut, A., Rashkin, H., Sap, M., Malaviya, C., Celikyilmaz, A., and Choi, Y. (2019). COMET: Commonsense Transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4762–4779.

Bouraoui, Z., Camacho-Collados, J., and Schockaert, S. (2020). Inducing relational knowledge from BERT. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, pages 7456–7463.

Breit, A., Revenko, A., Rezaee, K., Pilehvar, M. T., and Camacho-Collados, J. (2021). WiC-TSV: An evaluation benchmark for target sense verification of words in context. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 1635–1645.

Camacho-Collados, J. and Pilehvar, M. T. (2018). From word to sense embeddings: A survey on vector representations of meaning. *Journal of Artificial Intelligence Research (JAIR)*, 63(1):743–788.

Cao, S., Lu, W., and Xu, Q. (2015). GraRep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM)*, pages 891–900.

Cao, S., Lu, W., and Xu, Q. (2016). Deep neural networks for learning graph representations. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1145–1152.

Chen, W., Wang, X., and Wang, W. Y. (2021). A dataset for answering time-sensitive questions. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1 (NeurIPS Datasets and Benchmarks)*.

Chiu, B., Baker, S., Palmer, M., and Korhonen, A. (2019a). Enhancing biomedical word embeddings by retrofitting to verb clusters. In *Proceedings of the 18th BioNLP Workshop and Shared Task (BioNLP)*, pages 125–134.

Chiu, B., Majewska, O., Pyysalo, S., Wey, L., Stenius, U., Korhonen, A., and Palmer, M. (2019b). A neural classification method for supporting the creation of BioVerbNet. *Journal of Biomedical Semantics*, 10(1):1–12.

Cohen, W., Yang, F., and Mazaitis, K. R. (2020). Tensorlog: A probabilistic database implemented using deep-learning infrastructure. *Journal of Artificial Intelligence Research (JAIR)*, 67:285–325.

Colon-Hernandez, P., Havasi, C., Alonso, J., Huggins, M., and Breazeal, C. (2021). Combining pre-trained language models and structured knowledge. *arXiv preprint arXiv:2101.12294*.

Conneau, A., Rinott, R., Lample, G., Williams, A., Bowman, S., Schwenk, H., and Stoyanov, V. (2018). XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2475–2485.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Dai, D., Dong, L., Hao, Y., Sui, Z., Chang, B., and Wei, F. (2022). Knowledge neurons in pretrained Transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 8493–8502.

Dai, W.-Z., Xu, Q., Yu, Y., and Zhou, Z.-H. (2019). Bridging machine learning and logical reasoning by abductive learning. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NeurIPS)*, pages 2815–2826.

Dai, Y., Wang, S., Xiong, N. N., and Guo, W. (2020). A survey on knowledge graph embedding: Approaches, aplications and benchmarks. *Electronics*, 9(5):750–778.

De Cao, N., Aziz, W., and Titov, I. (2021). Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6491–6506.

De Raedt, L., Kimmig, A., and Toivonen, H. (2007). ProbLog: A probabilistic Prolog and its application in link discovery. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence (IJCAI)*, pages 2468–2473.

Demeester, T., Rocktäschel, T., and Riedel, S. (2016). Lifted rule injection for relation embeddings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1389–1399.

Deng, C., Ji, X., Rainey, C., Zhang, J., and Lu, W. (2020). Integrating machine learning with human knowledge. *Iscience*, 23(11):101656–101682.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). 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 (NAACL)*, pages 4171–4186.

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7):1895–1923.

Diligenti, M., Gori, M., and Saccà, C. (2017). Semantic-based regularization for learning and inference. *Artificial Intelligence*, 244:143–165.

Donadello, I. (2018). *Semantic Image Interpretation-Integration of Numerical Data and Logical Knowledge for Cognitive Vision*. PhD thesis, University of Trento.

Dong, H., Mao, J., Lin, T., Wang, C., Li, L., and Zhou, D. (2019). Neural logic machines. In *Proceedings of the 7th International Conference on Learning Representations (ICRL)*.

Fam, R. and Lepage, Y. (2018). Tools for the production of analogical grids and a resource of n-gram analogical grids in 11 languages. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC)*, pages 1060–1066.

Faruqui, M., Dodge, J., Jauhar, S. K., Dyer, C., Hovy, E., and Smith, N. A. (2015). Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 1606–1615.

Ferret, O. (2017). Turning distributional thesauri into word vectors for synonym extraction and expansion. In *Proceedings of the 8th International Joint Conference on Natural Language Processing (IJCNLP)*, pages 273–283.

Fillmore, C. J. and Baker, C. F. (2001). Frame semantics for text understanding. In *Proceedings of NAACL 2001 WordNet and Other Lexical Resources Workshop*.

Fiorini, N., Leaman, R., Lipman, D. J., and Lu, Z. (2018). How user intelligence is improving pubmed. *Nature biotechnology*, 36:937–945.

Ganitkevitch, J., Van Durme, B., and Callison-Burch, C. (2013). PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 758–764.

Gao, J., He, D., Tan, X., Qin, T., Wang, L., and Liu, T.-Y. (2019). Representation degeneration problem in training natural language generation models. In *Proceedings of the 7th International Conference on Learning Representations (ICRL)*.

Gao, T., Fisch, A., and Chen, D. (2021). Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 3816–3830.

Garneau, N., Hartmann, M., Sandholm, A., Ruder, S., Vulić, I., and Søgaard, A. (2021). Analogy training multilingual encoders. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, pages 12884–12892.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1263–1272.

Goodfellow, I. J., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press, Cambridge, MA, USA.

Grbovic, M. and Cheng, H. (2018). Real-time personalization using embeddings for search ranking at Airbnb. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 311—-320.

Grover, A. and Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 855–864.

Gu, J., Lu, Z., Li, H., and Li, V. O. (2016). Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1631–1640.

Hamilton, W. L., Ying, R., and Leskovec, J. (2017). Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin*, 40(3):52–74.

Haviv, A., Berant, J., and Globerson, A. (2021). BERTese: Learning to speak to BERT. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 3618–3623.

He, B., Zhou, D., Xiao, J., Jiang, X., Liu, Q., Yuan, N. J., and Xu, T. (2020). BERT-MK: Integrating graph contextualized knowledge into pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2281–2290.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Hendrycks, D. and Gimpel, K. (2016). Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*.

Hernandez, E. and Andreas, J. (2021). The low-dimensional linear geometry of contextualized word representations. In *Proceedings of the 25th Conference on Computational Natural Language Learning (CoNLL)*, pages 82–93.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 328–339.

Hu, M. and Liu, B. (2004). Mining and summarizing customer reviews. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 168–177.

Hu, Z., Ma, X., Liu, Z., Hovy, E., and Xing, E. (2016). Harnessing deep neural networks with logic rules. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2410–2420.

Huang, L., Sun, C., Qiu, X., and Huang, X. (2019). GlossBERT: BERT for word sense disambiguation with gloss knowledge. 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 3509–3514.

Hüllermeier, E. (2020). Towards analogy-based explanations in machine learning. In *Proceedings of the 17th International Conference on Modeling Decisions for Artificial Intelligence (MDAI)*, pages 205–217.

Hwang, J. D., Bhagavatula, C., Bras, R. L., Da, J., Sakaguchi, K., Bosselut, A., and Choi, Y. (2021). COMET-ATOMIC 2020: On symbolic and neural commonsense knowledge graphs. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*, pages 6384–6392.

Ji, S., Pan, S., Cambria, E., Marttinen, P., and Philip, S. Y. (2021). A Survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514.

Jiang, H., He, P., Chen, W., Liu, X., Gao, J., and Zhao, T. (2020a). SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2177–2190.

Jiang, Z., Gu, Q., Yin, Y., and Chen, D. (2018). Enriching word embeddings with domain knowledge for readability assessment. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, pages 366–378.

Jiang, Z., Xu, F. F., Araki, J., and Neubig, G. (2020b). How can we know what language models know? *Transactions of the Association for Computational Linguistics (TACL)*, 8:423–438.

Johnson, A. E., Pollard, T. J., Shen, L., Lehman, L.-w. H., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Anthony Celi, L., and Mark, R. G. (2016). MIMIC-III, a freely accessible critical care database. *Scientific Data*, 3(1):1–9.

Kapočiūtė-Dzikienė, J., Salimbajevs, A., and Skadiņš, R. (2021). Monolingual and cross-lingual intent detection without training data in target languages. *Electronics*, page 1412.

Keane, M. T. and Smyth, B. (2020). Good counterfactuals and where to find them: A case-based technique for generating counterfactuals for Explainable AI (XAI). In *Proceedings of the 28th International Conference on Case-Based Reasoning Research and Development (ICCBR)*, pages 163–178.

Kiefer, J. and Wolfowitz, J. (1952). Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466.

Kipf, T. N. and Welling, M. (2016). Variational graph auto-encoders. In *Proceedings of NeurIPS 2016 Workshop on Bayesian Deep Learning*.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.

Kipper, K., Korhonen, A., Ryant, N., and Palmer, M. (2008). A large-scale classification of English verbs. *Language Resources and Evaluation*, 42(1):21–40.

Kokhlikyan, N., Miglani, V., Martin, M., Wang, E., Reynolds, J., Melnikov, A., Lunova, N., and Reblitz-Richardson, O. (2019). Pytorch captum. In *Proceedings of ICRL Workshop on Responsible AI (RAI)*.

Korhonen, A., Krymolowski, Y., and Collier, N. (2006). Automatic classification of verbs in biomedical texts. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, pages 345–352.

Langlais, P., Yvon, F., and Zweigenbaum, P. (2009). Improvements in analogical learning: Application to translating multi-terms of the medical domain. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 487–495.

Lauscher, A., Majewska, O., Ribeiro, L. F. R., Gurevych, I., Rozanov, N., and Glavaš, G. (2020a). Common sense or world knowledge? Investigating Adapter-based knowledge injection into pretrained Transformers. In *Proceedings of Deep Learning Inside Out (DeeLIO): The 1st Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 43–49.

Lauscher, A., Vulić, I., Ponti, E. M., Korhonen, A., and Glavaš, G. (2020b). Specializing unsupervised pretraining models for word-level semantic similarity. In *Proceedings of the 28th International Conference on Computational Linguistics (COLING)*, pages 1371–1383.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of IEEE*, 86(11):2278–2324.

Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., and Kang, J. (2020). BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Lengerich, B., Maas, A., and Potts, C. (2018). Retrofitting distributional embeddings to knowledge graphs with functional relations. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, pages 2423–2436.

Lepage, Y. (2004). Analogy and formal languages. In *Proceedings of the Joint Meeting of the 6th Conference on Formal Grammar and the 7th Conference on Mathematics of Language (FG-MOL)*, pages 180–191.

Levy, O. and Goldberg, Y. (2014). Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 302–308.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (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 (ACL)*, pages 7871–7880.

Li, X., Xiong, H., Li, X., Wu, X., Zhang, X., Liu, J., Bian, J., and Dou, D. (2022). Interpretable deep learning: interpretation, interpretability, trustworthiness, and beyond. *Knowledge and Information Systems*, 64(12):3197–3234.

Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. S. (2016). Gated graph sequence neural networks. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*.

Lim, S., Prade, H., and Richard, G. (2019). Solving word analogies: A machine learning perspective. In *Proceedings of the 15th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, pages 238–250.

Liu, N. F., Gardner, M., Belinkov, Y., Peters, M. E., and Smith, N. A. (2019a). Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 1073–1094.

Liu, Q., Huang, H., Zhang, G., Gao, Y., Xuan, J., and Lu, J. (2018). Semantic structure-based word embedding by incorporating concept convergence and word divergence. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence and 30th Innovative Applications of Artificial Intelligence Conference and 8th AAAI Symposium on Educational Advances in Artificial Intelligence (AAAI-IAAI-EAAI)*, pages 5261–5268.

Liu, Q., Jiang, H., Wei, S., Ling, Z.-H., and Hu, Y. (2015). Learning semantic word embeddings based on ordinal knowledge constraints. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 1501–1511.

Liu, Q., Liu, F., Collier, N., Korhonen, A., and Vulić, I. (2021). MirrorWiC: On eliciting word-in-context representations from pretrained language models. In *Proceedings of the 25th Conference on Computational Natural Language Learning (CoNNL)*, pages 562–574.

Liu, W., Zhou, P., Zhao, Z., Wang, Z., Ju, Q., Deng, H., and Wang, P. (2020a). K-BERT: Enabling language representation with knowledge graph. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, pages 2901–2908.

Liu, X., Ji, K., Fu, Y., Tam, W., Du, Z., Yang, Z., and Tang, J. (2022). P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 61–68.

Liu, X., Xie, L., Wang, Y., Zou, J., Xiong, J., Ying, Z., and Vasilakos, A. V. (2020b). Privacy and security issues in deep learning: A survey. *IEEE Access*, 9:4566–4593.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019b). RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Logan, R., Liu, N. F., Peters, M. E., Gardner, M., and Singh, S. (2019). Barack's wife Hillary: Using knowledge graphs for fact-aware language modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 5962–5971.

Lu, H., Liu, Q., Ichien, N., Yuille, A. L., and Holyoak, K. J. (2019). Seeing the meaning: Vision meets semantics in solving pictorial analogy problems. In *Proceedings of the 41st Annual Conference of the Cognitive Science Society (CogSci)*, pages 2201–2207.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*, pages 4768–4777.

Majewska, O., Collins, C., Baker, S., Björne, J., Brown, S. W., Korhonen, A., and Palmer, M. (2021). Bioverbnet: a large semantic-syntactic classification of verbs in biomedicine. *Journal of Biomedical Semantics*, 12(1):1–13.

Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T., and De Raedt, L. (2018). DeepProbLog: Neural probabilistic logic programming. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS)*, pages 3753–3763.

Marra, G., Diligenti, M., Giannini, F., Gori, M., and Maggini, M. (2020). Relational neural machines. In *Proceedings of the 24th European Conference on Artificial Intelligence and the 10th Conference on Prestigious Applications of Artificial Intelligence (ECAI-PAIS)*, pages 1340–1347.

Marra, G., Giannini, F., Diligenti, M., and Gori, M. (2019a). Constraint-based visual generation. In *Proceedings of the 28th International Conference on Artificial Neural Networks (ICANN)*, pages 565–577.

Marra, G., Giannini, F., Diligenti, M., and Gori, M. (2019b). Integrating learning and reasoning with deep logic models. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, pages 517–532.

Marra, G., Giannini, F., Diligenti, M., and Gori, M. (2019c). LYRICS: A general interface layer to integrate logic inference and deep learning. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, pages 283–298.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133.

Mel'cuk, I. (1996). Lexical functions: A tool for the description of lexical relations in the lexicon. In *Lexical Functions in Lexicography and Natural Language Processing*, pages 37–102. John Benjamins Publishing Company.

Meng, K., Bau, D., Andonian, A., and Belinkov, Y. (2022). Locating and editing factual associations in GPT. In *Proceedings of the 36th Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 746–751.

Miller, G. A. (1995). WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

Mitchell, E., Lin, C., Bosselut, A., Finn, C., and Manning, C. D. (2022). Fast model editing at scale. In *Proceedings of the 10th International Conference on Learning Representations (ICRL)*.

Mitchell, M. (2021). Abstraction and analogy-making in artificial intelligence. *Annals of the New York Academy of Sciences*, 1505(1):79–101.

Mitchell, T., Cohen, W., Hruschka, E., Talukdar, P., Yang, B., Betteridge, J., Carlson, A., Dalvi, B., Gardner, M., Kisiel, B., et al. (2018). Never-ending learning. *Communications of the ACM*, 61(5):103–115.

Moreno, J. G., Pontes, E. L., and Dias, G. (2021). CTLR@WiC-TSV: Target sense verification using marked inputs and pre-trained models. In *Proceedings of the 6th Workshop on Semantic Deep Learning (SemDeep-6)*, pages 1–6.

Mrkšić, N., Ó Séaghdha, D., Thomson, B., Gašić, M., Rojas-Barahona, L. M., Su, P.-H., Vandyke, D., Wen, T.-H., and Young, S. (2016). Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 142–148.

Mrkšić, N., Vulić, I., Ó Séaghdha, D., Leviant, I., Reichart, R., Gašić, M., Korhonen, A., and Young, S. (2017). Semantic specialization of distributional word vector spaces using monolingual and cross-lingual constraints. *Transactions of the Association for Computational Linguistics (TACL)*, 5:309–324.

Mu, J., Bhat, S., and Viswanath, P. (2018). All-but-the-top: Simple and effective post-processing for word representations. In *Proceedings of the 6th International Conference on Learning Representations (ICRL)*.

Murena, P.-A., Al-Ghossein, M., Dessalles, J.-L., and Cornuéjols, A. (2020). Solving analogies on words based on minimal complexity transformation. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1848–1854.

Navigli, R. (2016). Ontologies. In *The Oxford Handbook of Computational Linguistics*, pages 518–546. Oxford University Press, 2nd edition.

Navigli, R. and Ponzetto, S. P. (2012). BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Nickel, M., Murphy, K., Tresp, V., and Gabrilovich, E. (2015). A review of relational machine learning for knowledge graphs. *Proceedings of IEEE*, 104(1):11–33.

Ou, M., Cui, P., Pei, J., Zhang, Z., and Zhu, W. (2016). Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1105–1114.

Peng, Y., Yan, S., and Lu, Z. (2019). Transfer learning in biomedical natural language processing: An evaluation of BERT and ELMo on ten benchmarking datasets. In *Proceedings of the 18th BioNLP Workshop and Shared Task (BioNLP)*, pages 58–65.

Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 701–710.

Pessach, D. and Shmueli, E. (2022). A review on fairness in machine learning. *ACM Computing Surveys (CSUR)*, 55(3):1–44.

Peters, M. E., Ammar, W., Bhagavatula, C., and Power, R. (2017). Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1756–1765.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). 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)*, pages 2227–2237.

Peters, M. E., Neumann, M., Logan, R., Schwartz, R., Joshi, V., Singh, S., and Smith, N. A. (2019). Knowledge enhanced contextual word representations. 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 43–54.

Petroni, F., Rocktäschel, T., Riedel, S., Lewis, P., Bakhtin, A., Wu, Y., and Miller, A. (2019). Language models as knowledge bases? 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 2463–2473.

Peyre, J., Sivic, J., Laptev, I., and Schmid, C. (2019). Detecting unseen visual relations using analogies. In *Proceedings of the 2019 International Conference on Computer Vision (ICCV)*, pages 1981–1990.

Poerner, N., Waltinger, U., and Schütze, H. (2020). E-BERT: Efficient-yet-effective entity embeddings for BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 803–818.

Polguère, A. (2009). Lexical systems: graph models of natural language lexicons. *Language resources and evaluation*, pages 41–55.

Polguère, A. (2014). From writing dictionaries to weaving lexical networks. *International Journal of Lexicography*, 27(4):396–418.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9–32.

Raedt, L. d., Dumančić, S., Manhaeve, R., and Marra, G. (2020). From statistical relational to neuro-symbolic artificial intelligence. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4943–4950.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text Transformer. *Journal of Machine Learning Research (JMLR)*, 21(140):1–67.

Ras, G., Xie, N., van Gerven, M., and Doran, D. (2022). Explainable deep learning: A field guide for the uninitiated. *Journal of Artificial Intelligence Research*, 73:329–397.

Reed, S., Zhang, Y., Zhang, Y., and Lee, H. (2015). Deep visual analogy-making. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS)*, page 1252–1260.

Reif, E., Yuan, A., Wattenberg, M., Viégas, F. B., Coenen, A., Pearce, A., and Kim, B. (2019). Visualizing and measuring the geometry of BERT. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NIPS)*, pages 8592–8600.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. 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 3982–3992.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "Why should I trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1135–1144.

Richardson, M. and Domingos, P. (2006). Markov logic networks. *Machine Learning*, 62(1):107–136.

Rocktäschel, T. and Riedel, S. (2017). End-to-end differentiable proving. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*, pages 3791–3803.

Rocktäschel, T., Singh, S., and Riedel, S. (2015). Injecting logical background knowledge into embeddings for relation extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 1119–1129.

Rogers, A., Kovaleva, O., and Rumshisky, A. (2020). A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408.

Roy, A. and Pan, S. (2021). Incorporating extra knowledge to enhance word embedding. In *Proceedings of the 29th International Conference on International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 4929–4935.

Roychowdhury, S., Diligenti, M., and Gori, M. (2021). Regularizing deep networks with prior knowledge: A constraint-based approach. *Knowledge-Based Systems*, 222:106989–106998.

Sadeghi, F., Zitnick, C. L., and Farhadi, A. (2015). Visalogy: Answering visual analogy questions. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS)*, pages 1882–1890.

Sap, M., Le Bras, R., Allaway, E., Bhagavatula, C., Lourie, N., Rashkin, H., Roof, B., Smith, N. A., and Choi, Y. (2019). ATOMIC: An atlas of machine commonsense for if-then reasoning. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*, pages 3027–3035.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2008). The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Schlichtkrull, M., Kipf, T. N., Bloem, P., van den Berg, R., Titov, I., and Welling, M. (2018). Modeling relational data with graph convolutional networks. In *Proceedings of the 15th European Semantic Web Conference (ESWC)*, pages 593–607.

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, pages 2673–2681.

Serafini, L. and Garcez, A. (2016). Logic tensor networks: Deep learning and logical reasoning from data and knowledge. In *Proceedings of the 11th International Workshop on Neural-Symbolic Learning and Reasoning co-located with the Joint Multi-Conference on Human-Level Artificial Intelligence (NeSy-HLAI)*, pages 23–34.

Shen, T., Mao, Y., He, P., Long, G., Trischler, A., and Chen, W. (2020). Exploiting structured knowledge in text via graph-guided representation learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8980–8994.

Shi, W., Chen, M., Zhou, P., and Chang, K.-W. (2019). Retrofitting contextualized word embeddings with paraphrases. 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 1198–1203.

Shin, T., Razeghi, Y., Logan IV, R. L., Wallace, E., and Singh, S. (2020). AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235.

Shrikumar, A., Greenside, P., and Kundaje, A. (2017). Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 3145–3153.

Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Proceedings of the International Conference on Learning Representations Workshop (ICLR)*.

Singh, P., Lin, T., Mueller, E. T., Lim, G., Perkins, T., and Zhu, W. L. (2002). Open mind common sense: Knowledge acquisition from the general public. In *Confederated International Conferences CoopIS, DOA, and ODBASE Proceedings*, pages 1223–1237.

Sinitsin, A., Plokhotnyuk, V., Pyrkin, D., Popov, S., and Babenko, A. (2020). Editable neural networks. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1631–1642.

Speer, R. and Havasi, C. (2012). Representing general relational knowledge in ConceptNet 5. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC)*, pages 3679–3686.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research (JMLR)*, 15(1):1929–1958.

Su, C., Chen, M., and Xie, X. (2021a). Graph convolutional matrix completion via relation reconstruction. In *Proceedings of the 10th International Conference on Software and Computer Applications (ICSCA)*, pages 51–56.

Su, Y., Han, X., Zhang, Z., Lin, Y., Li, P., Liu, Z., Zhou, J., and Sun, M. (2021b). CokeBERT: Contextual knowledge selection and embedding towards enhanced pre-trained language models. *AI Open*, 2:127–134.

Talmor, A., Elazar, Y., Goldberg, Y., and Berant, J. (2020). oLMpics-on what language model pre-training captures. *Transactions of the Association for Computational Linguistics (TACL)*, 8:743–758.

Tenney, I., Xia, P., Chen, B., Wang, A., Poliak, A., McCoy, R. T., Kim, N., Durme, B. V., Bowman, S. R., Das, D., and Pavlick, E. (2019). What do you learn from context? probing for sentence structure in contextualized word representations. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*.

Tissier, J., Gravier, C., and Habrard, A. (2017). Dict2vec : Learning word embeddings using lexical dictionaries. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 254–263.

Tshitoyan, V., Dagdelen, J., Weston, L., Dunn, A., Rong, Z., Kononova, O., Persson, K. A., Ceder, G., and Jain, A. (2019). Unsupervised word embeddings capture latent knowledge from materials science literature. *Nature*, 571(7763):95–98.

Ushio, A., Camacho-Collados, J., and Schockaert, S. (2021a). Distilling relation embeddings from pretrained language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9044–9062.

Ushio, A., Espinosa Anke, L., Schockaert, S., and Camacho-Collados, J. (2021b). BERT is to NLP what AlexNet is to CV: Can pre-trained language models identify analogies? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 3609–3624.

van der Maaten, L. and Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research (JMLR)*, 9(86):2579–2605.

Vandenbussche, P.-Y., Scerri, T., and Daniel Jr, R. (2021). Word sense disambiguation with Transformer models. In *Proceedings of the 6th Workshop on Semantic Deep Learning (SemDeep-6)*, pages 7–12.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*, pages 5998–6008.

Vig, J. (2019). A multiscale visualization of attention in the Transformer model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 37–42.

Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R., and Borgwardt, K. M. (2010). Graph kernels. *Journal of Machine Learning Research (JMLR)*, 11:1201–1242.

Vulić, I., Glavaš, G., Mrkšić, N., and Korhonen, A. (2018). Post-specialisation: Retrofitting vectors of words unseen in lexical resources. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 516–527.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. (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 (EMNLP)*, pages 353–355.

Wang, D., Cui, P., and Zhu, W. (2016). Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1225–1234.

Wang, R., Tang, D., Duan, N., Wei, Z., Huang, X., Ji, J., Cao, G., Jiang, D., and Zhou, M. (2021a). K-Adapter: Infusing knowledge into pre-trained models with Adapters. In *Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 1405–1418.

Wang, X., Gao, T., Zhu, Z., Zhang, Z., Liu, Z., Li, J., and Tang, J. (2021b). KEPLER: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics (TACL)*, 9:176–194.

Wang, X., Wang, H., and Yang, D. (2022). Measure and improve robustness in NLP models: A survey. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 4569–4586.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 38–45.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J. R., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G. S., Hughes, M., and Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Xu, J., Zhang, Z., Friedman, T., Liang, Y., and Van den Broeck, G. (2018). A semantic loss function for deep learning with symbolic knowledge. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 5502–5511.

Yamada, I., Shindo, H., Takeda, H., and Takefuji, Y. (2016). Joint learning of the embedding of words and entities for named entity disambiguation. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*, pages 250–259.

Yan, J., Wang, C., Cheng, W., Gao, M., and Zhou, A. (2018). A retrospective of knowledge graphs. *Frontiers of Computer Science*, 12(1):55–74.

Yang, B. and Mitchell, T. (2017). Leveraging knowledge bases in LSTMs for improving machine reading. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1436–1446.

Yang, F., Yang, Z., and Cohen, W. W. (2017). Differentiable learning of logical rules for knowledge base reasoning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*, pages 2316–2325.

Yang, J., Xiao, G., Shen, Y., Jiang, W., Hu, X., Zhang, Y., and Peng, J. (2021). A survey of knowledge enhanced pre-trained models. *arXiv preprint arxiv:2110.00269*.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., and Le, Q. V. (2019). XLNet: Generalized autoregressive pretraining for language understanding. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NIPS)*, pages 5753–5763.

Ye, Z.-X., Chen, Q., Wang, W., and Ling, Z.-H. (2019). Align, mask and select: A simple method for incorporating commonsense knowledge into language representation models. *arXiv preprint arXiv:1908.06725*.

Zaheer, M., Guruganesh, G., Dubey, A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., and Ahmed, A. (2020). Big Bird: Transformers for longer sequences. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS)*, page 17283–17297.

Zhang, H. (2004). The optimality of naive Bayes. In *Proceedings of the 17th International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, pages 562–567.

Zhang, Z., Han, X., Liu, Z., Jiang, X., Sun, M., and Liu, Q. (2019). ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1441–1451.

Zhang, Z., Wu, Y., Zhao, H., Li, Z., Zhang, S., Zhou, X., and Zhou, X. (2020). Semantics-aware BERT for language understanding. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, pages 9628–9635.

Zhou, M., Duan, N., Liu, S., and Shum, H.-Y. (2020). Progress in neural NLP: Modeling, learning, and reasoning. *Engineering*, 1505(1):275–290.

Zhou, W. and Chen, M. (2022). An improved baseline for sentence-level relation extraction. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (AACL-IJCNLP)*, pages 161–168.

Zhu, C., Rawat, A. S., Zaheer, M., Bhojanapalli, S., Li, D., Yu, F., and Kumar, S. (2020). Modifying memories in Transformer models. *arXiv preprint arXiv:2012.00363*.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 19–27.