Deep Natural Language Processing for User Representation

Clara Gainon de Forsan de Gabriac

SCIENCES SORBONNE UNIVERSITÉ

THÈSE DE DOCTORAT DE SORBONNE UNIVERSITÉ

Spécialité **Informatique**

École Doctorale Informatique, Télécommunications et Électronique (Paris)

Deep Natural Language Processing for User Representation

Traitement du langage naturel profond pour la modélisation d'utilisateurs

Présentée par

Clara Gainon de Forsan de Gabriac

Dirigée par

Vincent Guigue & Patrick Gallinari

Pour obtenir le grade de

DOCTEURE de SORBONNE UNIVERSITÉ

Présentée et soutenue publiquement le 13 Décembre 2021

Devant le jury composé de :

Anne Boyer *Professeur, Université de Lorraine – LORIA*	Rapportrice
Julien Velcin *Professeur, Université Lyon 2 – ERIC Lab*	Rapporteur
Alejandro Bellogín *Associate Professor, Universidad Autónoma de Madrid – IRG@UAM*	Examinateur
Mohamed Chetouani *Professeur, Sorbonne Université – ISIR*	Examinateur
Vincent Guigue *Maître de Conférence, Sorbonne Université – LIP6*	Encadrant de thèse
Patrick Gallinari *Professeur, Sorbonne Université – LIP6*	Directeur de Thèse

ABSTRACT

The last decade has witnessed the impressive expansion of Machine Learning (ML) and in particular Deep Learning (DL) methods, both in academic research and the private sector. This success can be explained by the ability DL to model ever more complex entities. In particular, Representation Learning (RL) methods focus on building latent representations from heterogeneous data that are versatile and re-usable, namely in Natural Language Processing (NLP). In parallel, the ever-growing number of systems relying on user data (social media, recommendation systems ...) brings its own lot of challenges. This work proposes methods to leverage the representation power of NLP in order to learn rich and versatile user representations.

Firstly, we detail the works and domains associated with this thesis. We study Recommendation; a field where User Representations (URs) has long been at the heart of researches. We then go over recent NLP advances and how they can be applied to leverage user-generated texts, before detailing Generative models, that we feel are one of the purest expressions of RL.

Secondly, we present a Recommender System (RS) that is based on the combination of a traditional Matrix Factorization (MF) representation method and a sentiment analysis model. The association of those modules forms a dual model that is trained on user reviews for rating prediction. Experiments show that, on top of improving accuracy performances, the model allows us to better understand what the user is really interested in in a given item, as well as to provide explanations to the suggestions made.

Finally, we introduce a new task centered on UR: Professional Profile Learning. User profiles are often composed of several fields or attributes of different nature, and it is our thesis that user-generated textual data are not only rich enough to uniquely represent a user, but also to predict the other attributes of the profile. We thus propose an NLP-based framework, *Resumé*, to learn and evaluate professional profiles on different tasks, including next job generation.

RÉSUMÉ

La dernière décennie a vu s'imposer le développement exponentiel des méthodes de Machine Learning (ML) et en particulier de Deep Learning (DL), aussi bien dans le monde académique qu'industriel.

Ce succès peut s'expliquer par la capacité du DL à modéliser des entités toujours plus complexes. En particulier, les méthodes de Representation Learning (RL) se concentrent sur l'apprentissage de représentations latentes issues de données hétérogènes, à la fois versatiles et réutilisables, notamment en Natural Language Processing (NLP).

En parallèle, le nombre grandissant de systèmes reposant sur des données utilisateurs (réseaux sociaux, systèmes de recommendation ...) entraînent leur lot de défis.

Cette thèse propose des méthodes tirant partie du pouvoir de représentation du NLP pour apprendre des représentations d'utilisateur riches et versatiles.

Dans un premier temps, nous détaillons les travaux et domaines de recherche en lien avec cette thèse. Nous étudions la Recommandation, qui place de la représentation utilisateur au coeur de ses recherches depuis longtemps. Nous parlons ensuite des récentes avancées du NLP ainsi que des moyens de les appliquer de façon à tirer partie des textes écrits par les utilisateurs, pour enfin détailler les modèles génératifs, qui sont, à notre sens, une des formes les plus pure de RL.

Dans un second temps, nous présentons un Système de Recommandation fondé sur la combinaison, d'une part, d'une méthode de représentation par factorisation matricielle traditionnelle, et d'autre part, d'un modèle d'analyse de sentiments. L'association de ces deux modules forme un modèle double entraîné à prédire des notes à partir des avis textuels laissés par les utilisateurs. Nos expériences montrent que, en plus d'améliorer les performances en *accuracy*, ce modèle nous permet également de comprendre ce qui intéresse l'utilisateur chez un produit, en plus de fournir des explications concernant les suggestions émises par le modèle.

Enfin, nous présentons une nouvelle tâche centrée sur la représentation d'utilisateur : l'apprentissage de profil professionnel. Les profils d'utilisateurs sont souvent composés de plusieurs champs et attributs de natures différentes, et notre thèse est que les textes écrits par les utilisateurs sont non seulement assez riches pour identifier un utilisateur de façon unique au sein d'un système donné, mais également pour prédire les autres attributs du profil. Nous proposons donc *Resumé*, un cadre de travail pour l'apprentissage et l'évaluation des profils professionnels sur différentes tâches, notamment la génération du prochain job.

ACKNOWLEDGMENTS, REMERCIEMENTS

Ce manuscrit ne saurait refléter ma thèse sans mentionner et remercier les personnes qui m'ont permis d'aller au bout.

Merci Thomas, mon amour, mon partenaire, mon plus grand fan et mon modèle. Merci de n'avoir jamais remis en doute les difficultés dont je te faisais part, merci de m'avoir montré que je pouvais être heureuse et merci de m'avoir accompagnée à chacune des étapes qui ont certes mené à l'aboutissement de cette thèse, mais surtout, à mon bonheur. Merci de m'avoir fait rire, rêver et vibrer et de m'avoir rappelé constamment que la vie est pleine de bonnes raisons de se lever. Merci.

Merci ma famille, aussi large que diverse, de m'avoir soutenue moralement durant ce long chemin malgré son côté forcément abstrait. Merci Louise d'être ma meilleure amie. Merci Mathieu et Julie pour votre enthousiasme débordant (et un peu déconcertant) concernant mon travail. Merci Anne, Thierry, Isa et David de vous y mettre à autant pour me fournir un système de soutien solide comme un roc.

Merci Clara et Elsa, d'avoir toujours été à l'écoute de mes moments de désespoirs, coup de gueules et d'euphories quasi hystériques. Merci d'être aussi fières de moi alors que l'informatique vous paraît être de la magie noire. Merci d'être à mes côtés depuis aussi longtemps, quel que soit le pays, quel que soit le diplôme, quels que soient les choix de vie (si vous racontez des dossiers, je nierais tout en bloc, sachez-le).

Merci Pierre et Romain, pour notre dynamique inimitable de la triplette du chaos. Merci d'avoir toujours répondu présent lorsque j'ai proposé un verre, même lorsque ces verres se sont transformés en soft, ou en thé. Merci de m'avoir posé les bonnes questions, quand bien même dures, quand j'ai eu besoin de les entendre. Merci de n'avoir jamais failli à me donner des réponses beaucoup plus douces et justes que les miennes.

Merci Perrine d'avoir rendu cette thèse vivable grâce à nos discussions, nos magouilles, nos fous rires, nos sushis quand on ne voulait voir personne d'autre, nos cafés quand on avait besoin d'air. Merci pour la confiance que tu m'as accordée, et de toujours avoir été à la hauteur de la mienne. Merci de rester un point fixe,

même après la labo, parce que la vie est quand même vraiment mieux avec notre très petit gang.

Merci aux autres doctorant•e•s du laboratoire qui ont rendu cette thèse plus facile, que ce soit grâce à nos discussions, nos baby foots, nos cafés ou nos soirées. Un merci tout particulier à Antoine, Yuan et Jean-Yves, dont la bienveillance sans égale a été extrêmement précieuse tout au long de ces années.

Merci Dr Bouteille de m'avoir écoutée parler plus d'une heure lors de notre première rencontre. Merci de m'avoir constamment rappelée que souffrir n'est pas normal et qu'on peut évidemment faire autrement. Merci aussi de m'avoir montré que j'étais assez forte pour prendre mes problèmes à bras le corps quelle qu'en soit la nature.

À tous et toutes, je suis extrêmement fière de vous compter parmi mes amis et ma famille. Merci.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AE	Auto-Encoder
AI	Artificial Intelligence
BART	Bidirectional and Auto-Regressive Transformers
BERT	Bidirectional Encoder Representations from Transformers
BiRNN	Bidirectional Recurrent Neural Network
biLM	Bidirectional Language Model
BLEU	BiLingual Evaluation Understudy
BoW	Bag of Words
BPE	Byte-Pair Encoding
BT	Back Translation
CB	Content-based
CF	Collaborative Filtering
CNN	Convolutional Neural Network
CV	Curriculum Vitae
DAE	Denoising Auto-Encoder
DL	Deep Learning
DNN	Deep Neural Network
ELBO	Evidence Lower BOund
ELMo	Embeddings from Language Models
GAN	Generative Adversarial Network
GPT	Generative Pre-Training
GRU	Gated Recurrent Unit
HR	Human Resources
HRAN	Hierarchical Recurrent Attentive Neural Network for Recommendation
IR	Information Retrieval
IT	Information Technologies)
LM	Language Model
LDA	Latent Dirichlet Allocation

LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MF	Matrix Factorization
ML	Machine Learning
MLM	Masked Language Model
MLP	Multi-Layer Perceptron
MSE	Mean-Squared Error
MTL	Multi-Task Learning
NLP	Natural Language Processing
NLL	Negative Log-Likelihood
NMT	Neural Machine Translation
NN	Neural Network
NSP	Next Sentence Prediction
ReLU	Rectified Linear Unit
RBA	Recurrent Bidirectional Attentive modules
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
RS	Recommender System
RL	Representation Learning
UR	User Representation
VAE	Variational Auto-Encoder
xAI	Explainable Artificial Intelligence

CHAPTER 1

INTRODUCTION

Contents

1.1 Context

Historians describe the period ranging from the middle of the 20th century to today as the Information Age, and with good reason. The development of computers in the mid-to-late 20th century, fueled by the need to solve problems via computations (namely, encryption algorithms during World War 2, but also more prosaic issues like accountability or regression for time series prediction), led to the advent of the Internet as we know it in the end of the last century. In the last decades, the number of web pages have skyrocketed, ranging from online stores to personal blogs to encyclopedic pages. This massive growth of all things on the internet gave birth to a new domain of interest: the data, and in particular, how to select, represent, and access it efficiently. This is the context in which Artificial Intelligence, in particular in the form of Machine Learning bloomed.

Along the way, the progress made in hardware opened the way to representing functions of evermore complex nature, at the expense of the models' number of parameters. This paradigm of heavy, multi-layered models is called Deep Learning. The progresses of Deep Learning are as astounding as they are varied. In the field of Reinforcement Learning, Convolutional Neural Network based models outperformed human performances at the game of Go (Silver et al. 2016) and even Starcraft (Vinyals et al. 2019). In the Natural Language Processing field, in 2019, the Generative Pre-Training model proposed long computer-generated

text of uncanny fluency - so uncanny in fact that their author refused to open source the weights of the model, for fear of malicious usage. Unfortunately, it is hard to prove them wrong. The advances of Deep Learning in Computer Vision and Signal Processing allowed the development of "DeepFakes": audio or video files that replace either the voice or the face of the original speaker by the of... virtually anyone. In short, Deep Learning's recent advances are both rapid and impressive, sometimes tip-toeing the line of surpassing human performances.

Beside the progresses made in hardware and in Deep Learning, the Information Age has seen an explosion in the mass of data being produced and made available, namely in the form of e-business platforms. This amount of items to chose from on the internet created a need for customers (or users) to be helped in their choice, namely by selecting a subgroup of items for them to chose from rather than the whole catalog. This need is addressed by Information Retrieval methods and Recommender System models, that are at the heart of this work. Recommender Systems (RSs) are models that aim at presenting the users of a platform with items that they may like.

1.2 Motivations

Before diving into the specifics of this work, we would like to define three different levels at which research can be done: the application domain (Recommendation, Natural Language Processing), the task formalization (classification, ranking), and the tools (the algorithm used, like Matrix Factorization). This taxonomy aims at easing the reader's understanding of the level of abstraction at stake in the different chapters of this work. For example, a classic movie recommendation task would belong to the Recommendation domain, formalized as a Regression task and solved via a neighborhood and similarity approach, the tool. The work around User Representation can be addressed at any of those levels, as it is a problem that can be found in many use-cases.

In this section, we present the motivations of this work. We divide them into two categories: the technical motivations, that can broadly be seen as emanations of Representation Learning, and applicative motivations.

1.2.1 Representation Learning

All the works presented in this thesis are in line with the philosophy of Representation Learning.

The strength of Representation Learning lies in its capacity to project data living in a discrete space into a continuous one. This implies a capacity to take into account heterogenous and / or structured data and to aggregate them into a single multifaceted representation. This representation can then be fine-tuned for other, more specific down-stream tasks. This versatility of representation directly allows for transfer learning, but also multi-task learning.

Transfer and Multi-task Learning. Transfer Learning is the fact of learning a representation on a source task (for instance, car image recognition) to "jump start" the training of a related task (truck image recognition). Doing so implies that the initially learned representations have enough low-level and structural information to be applied to similar objects. This approach in central in Natural Language Processing where most popular architectures are based on models pre-trained on various unsupervised text generation tasks, before being fine-tune on down stream tasks, often in a Multi-Task learning fashion. Multi-Task learning is a learning paradigm that uses multiple tasks as a training objective in order to achieve better performances in a target task. The idea behind this method is that task-specific representations can be detrimental to the final performances, because then we lose the chance to model and use signals that can help solve related tasks. In other words, we train a model to solve several related tasks in order to get richer representations from the input. It is an approach we describe in Chapter 4. Transfer Learning is another learning paradigm that also leverages the idea of "related tasks".

Both Multi-Task learning and Transfer Learning have been gaining tractions over the past years, deepening the community's interest in Representation Learning as proven by the overwhelming success of the Transformers and their *pre-train-then-fine-tune* approach in Natural Language Processing, as well as Generative Adversarial Networks (GANs) in the domain of Computer Vision.

Our approach to Representation Learning. It is our opinion that a Representation Learning approach to user modeling could vastly improve performances and broaden applications, from classical Recommender System to personalized generative predictions. In this work, we define User Representation as the task of predicting a person's or a group's behavior and tastes by learning their latent representations. Indeed, User Representation can mean more accurate identification of a person's needs or wants, and capacities, and thus a higher quality of service provided to them, from movie recommendation to job matching.

We address this idea by using one of the richest traces left by users, natural language traces, to learn a user representation, and leverage the remainder of the user's information to either refine or evaluate their representation. In Chapter 3,

we combine a user's ratings to their written reviews to accurately recommend them items, as well as providing explanations regarding their tastes. In Chapter 4, we use the user's professional experiences (written in natural language) to build their representation, and use the other available information to evaluate their representation, namely through a text-generation task. Finally, we propose exploratory architectures to learn a user's career dynamic based on their professional experiences in Chapter 5.

1.2.2 Applicative motivations

Improving existent methods of user Representation Learning can be beneficial to any domain that can profit from a personalization mechanism, such as Recommendation, but also Information Retrieval or Professional coaching. If we can build rich, versatile user representations that capture their particularities, tastes, and needs, we can improve the recommendations (or predictions) made to them. We call rich and versatile the user representations that contain discrete information on which we had no robust metric before Representation Learning. This work aims to build such representations by using user-generated textual data.

Textual traces that are generated by a user are a very rich source of information: natural language is such that there are no two people that write the same thing the same way, and the way a user writes can tell a lot about them. Users will mostly write about what is important to them, so identifying the content as well as the polarity of their textual data can tell us about their interest.

Aside from being very dense with information, users textual traces also present the advantage of being quite omnipresent. The most obvious use-case is reviews left by users about a product or a location on a Recommender System, which has been (and still is) extensively studied. But we can also think of another, less studied application: professional resumes, or Curriculum Vitae. Curriculum Vitaes (CVs) are an interesting application because despite their apparent structure and code, they do not follow a standard that would be common to different countries or even different industrial fields. Thus, the tasks around Professional Profile learning require to understand a lot about a user to be carried out, whether one wants to predict a person's next job or extract high-level information from their career. Incidentally, our relationship to work and job-hopping has drastically changed in the past decades: it is more and more rare to spend one's entire career in the same company, whereas it used to be the norm not so long ago. For this reason we think that the work around Professional Profile Learning is increasingly relevant as it addresses contemporary challenges.

Finally, explainability is a central part of our work. We use Natural Language Processing methods such as text-generation and attention mechanism to provide insights regarding our models' behaviours, in line with the movement of Explainable Artificial Intelligence. We detail the stakes of explainability in Section 1.3.

1.3 Ethics and AI

The impressive advances of Artificial Intelligence and the fact that some models can now solve problems that that used to be limited to human beings raised a lot of new ethical questions about Artificial Intelligence. The more we use Artificial Intelligence in our daily life and in a growing number of domains, the more we need to make sure, as a society, that those models do not harm humans. Far from the cliché of a SkyNET's Terminator, the impact of Artificial Intelligence on human society is still very real and serious.

Large scale replication. Contrary to what science fiction has long been predicting, the ethical risks of Artificial Intelligence are not so much that models might make irrational or unpredictable decisions, but rather that they can replicate human-made errors to a much, much larger scale. Since an Artificial Intelligence model is typically trained on a lot of human-annotated data corpora, the biases present in this data will be learned (and thus replicated at inference time) by the model. A rather extreme example of human-made prejudices made a model perpetrate discrimination is the 2016's Microsoft's Artificial Intelligence chatbot. Meant to be trained as a conversational agent from twitter data, was completely hijacked by 4chan [1] users and only replied in racist/anti-semitic/sexist manners after only 24 hours.

While this example is extreme in the way that it was intentionally hijacked by ill-meaning people, it does highlight the fact that an Artificial Intelligence is only ever as neutral as the data it has been trained on. In other word, it is not only never neutral, it is also a vector of society's prejudices; a vector that is much more constant, efficient and quick to spread prejudice than any human.

Privacy. Besides, Artificial Intelligence models can be used to influence people's decisions and behaviors unbeknownst to them. The most famous example of such manipulations is probably the Cambridge Analytica scandal of 2016.

The Cambridge Analytica company took advantage of unlawfully collected data, and used them to build archetypal voters profiles. They have been reported

1. An anonymous online forum, known for its absence of moderation system.

to use those profiles to influence people over political issues such as the Brexit and Donald Trump's presidential campaign. The explosion of user data platforms and data science technologies made possible for this company to 1) illegally collect users' personal Facebook information (and the of their Facebook friends), 2) use those data to influence the course of political events, all without the users' consent or knowledge.

It is interesting to note that the algorithms used to influence people's votes are the same used to influence them to buy such or such television. The ethical problem here lies within the application rather than in the algorithms themselves.

Disinformation. Another change brought on by data-driven platforms is massive disinformation. For instance, the 2020's disinformation campaigns regarding Covid 19 have had undeniably terrible consequences on public health.

The success of those fake news can, in our opinion, be attributed to two things: 1) the exposure received by malicious sponsored contents on social media, and 2) the Echo Chamber phenomenon (i.e., the action for a user to remove divergent opinions from their feed). While the Echo Chamber is a known and studied limitation of RSs, it has found a particular resonance amidst the pandemic's conspiracy theories.

It is certain that conspiracy theories would have never strived if there was not fertile ground among the population to begin with (after all, it is a conscious action for the user to remove disagreeing opinions from their feed), but today's social media platforms and recommendation algorithms allowed ill-intentioned or misinformed people to spread their views to a previously unattainable scale.

AI in Human Resources. Artificial Intelligence also raises ethical concerns that are closer to this work, that is largely based on user data. An obvious one is privacy, and more precisely user consent. More and more online systems leverage the users' interactions with the system (click, scroll, view) but also their profile metadata to profile them. This profiling can be used to provide users with suggestions, or place them in marketing segment. The issue is that the user may not know which of their information is used, or to what end, and they can seldom manage it.

This consent and privacy issue is aggravated by the opaque nature of Machine Learning models, often referred to as the "black-box" problem. There is little to no way of knowing whether the model learned a biased decision function that discriminates people based on their name or gender rather than on relevant information.

Explainability. In thie context of this work, Explainability is thus a crucial challenge. However, when it is not ethically and legally mandatory (such as in examples cited above), Explainable Artificial Intelligence is still relevant in cases where the stakes are to build a sense of trust for the user as they interact with a system.

Although the question of understandability has been around for a while in the form of Expert Systems, statistics (feature selection) Bayesian Networks or decisions trees. The past few years has seen the emergence of post-hoc flexible, readily available frameworks such as LIME (Ribeiro et al. 2016) or SHAP (Lundberg and S.-I. Lee 2017) to further encourage and develop explainability of models. Those frameworks are referred as *post-hoc* because they are used at inference time, after the model converged, namely for failure analysis. Our work is part of a community effort in the form of multi-modal and generative explainability as we leverage attention mechanisms (Chapter 3) and text generation (Chapter 4) to that end.

This aspect is especially critical in this work, since we work on professional profiles from LinkedIn. LinkedIn[2] is a professional social network where people's profiles are composed of their education, professional experiences and skills (along with traditional information such as their name, city etc.). Our work with with data is thus a part of the Human Resources application domain. While not exactly new, this domain still has issues growing and remains somewhat overlooked.

Challenges. A possible explanation for that is the strong resistance opposed by the people working in Human Resources, but also by the people subject to recruitment processes. Another possible explanation lies in the ethical questions raised by such work. In the context where recruiting processes would be aided by Deep Learning programs, the question of responsibility would become a delicate one. Imagine for instance that the used model would only pre-select candidates with a certain ethnicity, or systematically reject people coming from a certain high-school or city. Such applicants would be victim of discrimination. But who would be to blame? The program, the person that trained it or the company that uses it? This hypothetical situation also questions the nature of data anonymity and the heated debate of whether or not one should make statistics based on ethnicity, sexual orientation or any other possibly discriminating criteria. It is a strategy used by some Fairness models (i.e., models that explicitly attempt bias correction), as well as some countries like the USA. Although the data we use in this work contain no such sensitive information, it could still be used to highlight discrimination based for instance on the candidate's high-school. The question then asked is: should we explicitly correct this bias by boosting the profiles that

2. https://www.linkedin.com

contain this high-school? Would that ensure equal opportunities, or would it simply lead said high-school to be blacklisted by companies?

We feel those questions need to be at the heart of current and future research, and possibly even state politics if works such as ours are ever to be used in real world Human Resources department. In fact, both the research community and political organizations have already started addressing this issue in the form of Explainable Artificial Intelligence (i.e., the field of Artificial Intelligence which goal is the make "black-box" program's decisions understandable) for the former and the "right to explanation" [3] for the latter.

1.4 Contributions and outline

In this manuscript, we will first go over the numerous works related to ours, detailing the advances of Recommendation, Natural Language Processing and Generative models. In particular, we will show that recent trends Recommendation aim at reconsidering traditional regressive approaches in favor of Representation Learning oriented methods. We will show that Recommendation's focus on rating prediction is shifting towards new qualities, such as understandability, and thus towards new evaluation processes.

In parallel, the recent advances of Natural Language Processing also leaned towards a Representation Learning paradigm. The recent advances of word embeddings and sequence modeling tend to favor a "general pre-training then specific-fine-tuning" approach. Those Natural Language Processing advances allowed us to build user representations by leveraging their textual traces, which are plentiful and user-defining.

The advances of Natural Language Processing also allowed for the expansion of one of the purest form of User Representation Learning: text generation. Text generation is of particular importance to this work because generating relevant text from a latent representation implies that you have achieved a good, abstract representation of the input.

We will then present a model that leverages an attention mechanism on textual review to improve recommendation performances and explain the personalized suggestions made by the system. This dual model is composed of a sentiment analysis part, and a traditional Matrix Factorization part. It aims at combining a representation of the user's tastes, learned via the reviews they wrote, with a Matrix Factorization module in order to better understand the tastes of the

3. As written in the European Union's General Data Protection Regulation bill.

user, but also to understand what characteristics of the product matters most to them. The model's sentiment analysis module then acts as a personalized reading model, providing insights regarding the output recommendations. In this work, our contribution lies in the *Problematic* category: we reformulate a classic Recommendation situation into a sentiment analysis task coupled with a regression.

After that, we present a framework for the learning and evaluation of professional profiles, *Résumé*, thus presenting a new application domain: Human Resources.

The framework relies on a Representation Learning approach in the sense where we aim at comparing the richness of the user representations on three different downstream tasks. We do so by leveraging user-generated texts (their professional experiences) as well as high-level side information of a Curriculum Vitae.

Chapter 5 proposes interesting leads for future work on User Dynamic Modeling. Modeling the dynamic (or evolution) of a user is an original way to improve predictions of their tastes or needs.In this instance, we are interested in modeling the career evolution of users, by using the information of their past professional experiences. We propose two exploratory architectures as well as the challenges posed by User Professional Dynamic Modeling.

CHAPTER 2

RELATED WORK

Contents

Representing and predicting user behavior has long been a topic of interest in Machine Learning. It is a critical issue for many applications, such as Churn Detection (Berger and Kompan 2019; Pudipeddi et al. 2014; Kwon et al. 2019), Advertising (Tu and Lu 2010; Yan et al. 2009), Social Media (Hallac et al. 2019; Wang et al. 2019) and more.

In this work, we tackle User Representation (UR) for Recommendation and Professional Profile Learning from an Natural Language Processing (NLP) point of view. We are also interested in explaining and understanding the obtained user representations, and we propose to do so through text generation, which we consider to be one of the most expressive form of Representation Learning (RL).

One of the domains where UR is most critical is without a doubt Recommendation. We describe the different approaches to UR in Recommender Systems in Section 2.1.

A growing trend in later Recommender Systems (RSs) is to use not only the user's rating of an item, but also their reviews. We propose a user representation based on those texts, as opposed to the Collaborative Filtering (CF) usual focus on user-item interactions. This has been made possible by the rapid evolution of NLP methods, explained in Section 2.2.

An approach that particularly interests us in this work is to use a user representation to generate text. This process relies on the assumption that if we represented a user well enough, we can then generate word sequences that correspond, for instance, to their next job. In this sense, generating text from a learned user representation is a very powerful tool for RL. On top of that, we feel that generating text from a user representation is an original and robust way towards explainability. It is however worth mentioning that evaluation metrics of generated texts are limited, and making those models both tricky to train and to test. We go through generative models in Section 2.3

This chapter presents an overview of the domains related to our work, starting by Recommendation. We will then cover the recent evolutions of NLP before discussing generative models.

2.1 User Representation in Recommendation

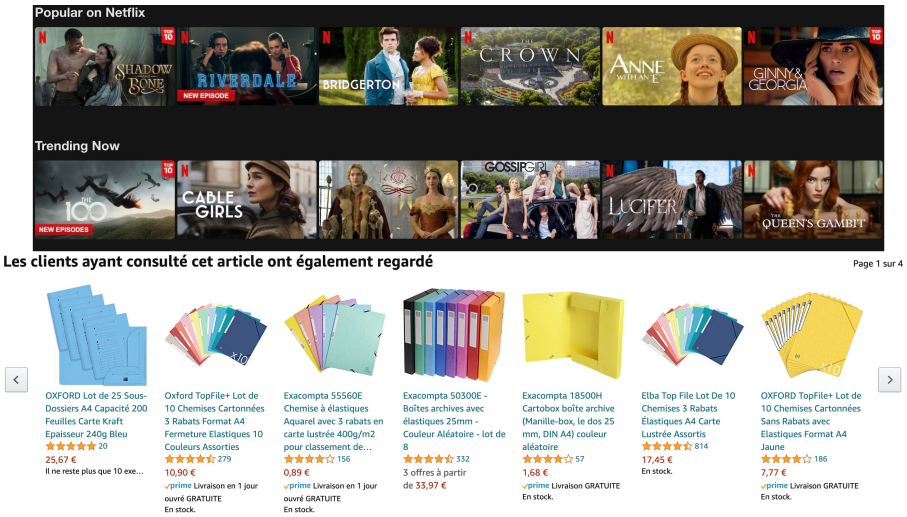

Figure 2.1 – **Examples of Item Recommendation**. on Netflix (top) and Amazon (bottom)

A good RS should present users with relevant new items (items that users may like or find useful), but they should also provide recommendations that are both personalized and understandable. At every step of the process, a good user representation is crucial.

Relevance. Determining what items are relevant to recommend to a given user is a vast and open question. The most classical approach is to consider that relevant items are items that the user would rate with a good mark. This approach is in fact a very common learning objective for RS: the model is trained to predict the rating of a user-item pair, and outputs the items with the highest ratings in inference. Those models are said to be optimized on accuracy. There is however a growing trend to move away from this accuracy-based approach and to consider other qualities that should be modeled, either as training objectives or as evaluation metrics (namely, ranking as an objective and Mean Reciprocal Rank as a metric). Aside from the fact that user experience can highly benefit from being recommended unexpected or surprising items, the notion of Diversity [1] has been at the heart of many works in the past years and still remains quite hard to evaluate.

Personalization. Recommending only popular items is far from being an ideal behavior for a RS, as different people have different needs and tastes. Indeed, and while non-personalized RSs could already yield satisfactory results in terms of suggestions and revenue (namely by using only the item bias), we think User experience can greatly benefit from personalization. This is why personalization is a major feature for a RS: taking a user's past preferences into account is a sound way to refine recommendations made to them. We observe two mains strategies to personalize recommendations. On one hand, some Content-based (CB) approaches compute a similarity between a given user and potential items; the recommendation is the item that has the best match with the user. On the other hand, Collaborative Filtering (CF) addresses personalization by finding similar profiles that are similar to the current user, and then recommends items likes by those similar profiles.
Another aspect of personalization worth exploring is the evolution of the user. J. J. McAuley and Leskovec (2013) in particular proposed a framework that refined user recommendations by modeling their level of expertise. We explore this aspect of personalization in Chapter 5.

1. In this work, we will use the terms Diversity, Unexpectedness and Serendipity interchangeably as we do not address those issues separately from one another. We are simply interested in the notion of *surprising* the user.

Finally, a lot of work has been devoted to leveraging user-generated textual reviews to improve the quality and the personalization of the recommendations.

Understandability. A RS recommendations can be as precise and relevant as possible, users will tend to be wary of them if they do not know how they were generated. The opacity of recent RSs is a well-known weakness of theirs. Indeed, the shift from Neighborhood-based to Matrix Factorization (MF) approaches led to an increase in performances, but to a decrease in understandability. The Explainable Artificial Intelligence (xAI) movement emerged as a reaction to the general opacity of Machine Learning (ML) models (i.e., the fact that their decisions are hard to interpret). xAI aims at providing methods and models that can be humanly understandable. In line with these works, we present a way to alleviate the black-box problem by using attention mechanisms on textual reviews in Chapter 3.

Ethics. Several works have been studying the tendency of RS to create Echo Chambers (also called Feedback loops), i.e., a situation in which a user is only exposed to the same kind of content, based on their feedback to the system. These Echo Chambers create a self-reinforcing pattern for users (Jiang et al. 2019) that they can hardly break free from alone (Noordeh et al. 2020). Since those articles, the world has been hit by the Coronavirus pandemic, and many pointed out the role of RS and Social Media in the increase of conspiracy theory bubble filters, further increasing the need for more diverse training objectives and evaluation metrics. Besides, while the ethics of restaurant recommendation might not be immediately critical, there are other applications domains where ethics must absolutely be considered, such as Human Resources (HR), which we cover in Chapter 4. In such applications domains, the issue is less to "surprise" the user rather than not discard potential choices based on a discriminative criterion.

There are two main approaches to RS : Collaborative Filtering (CF) and Content-based (CB). CF is a framework that aims at computing latent representations of users and items, and then makes recommendations based on what users similar to our current users liked. Content-based (CB) on the other hand, suggests a given user items that are similar to those they liked before. Our work takes from both frameworks for user representation. Like CF, we learn latent spaces in which to project users profiles, in a Representation Learning fashion. The obtained representation is abstract, but also versatile. At the same time, we use a lot of the CB techniques in order to represent users from textual traces. Since this thesis is centered around NLP, we will first discuss Section 2.1.1, before going into Section 2.1.2, which is historically more text-oriented. We will also detail the role

played by NLP in recommendation, and how it has a crucial role for this thesis. Whatever the approach chosen, RSs are hard to evaluate, because the very notion of "relevant items" is not trivial to define and formalize. We will explore the traditional training and evaluation metrics, their limitations and their alternative in Section 2.1.5.

2.1.1 Collaborative Filtering

Figure 2.2 – **Schematic illustration of a user-item matrix**.

Collaborative Filtering models rely on the assumption that users that have had similar tastes in the past are likely to have similar tastes in the future. The tastes of the users are expressed by how they rate items that they experienced. These ratings are often called *interactions*. Let us use the illustration in Figure 2.2. Since the green user and the blue one rated the purple and the cyan movies similarly, CF expects the blue user to love the orange movie.

Traditionally, a RS of parameters θ is trained to optimize the cost function between a predicted rating of an item by a user $\widehat{r_{u,i}}$ and the actual rating $r_{u,i}$, that is:

$$\min_{\theta} \frac{1}{R} \sum_{u \in U} \sum_{i \in I_u} (r_{u,i} - f(u, i|\theta))^2 \tag{2.1}$$

$$\widehat{r_{u,i}} = f(u, i|\theta) \tag{2.2}$$

with R the total numbers of ratings, U the set of users, and I_u the set of items rated by user u. Note that while Mean-Squared Error (MSE) is a common learning objective, it is neither the only possible objective nor is it always the chosen

evaluation metric. We further discuss evaluation of recommendation, as well as alternative training objectives in Section 2.1.5.

The prediction of a rating can account for several biases: the global bias of the dataset μ, the current user's personal bias b_u and the current item's bias b_i.

$$\begin{aligned} \widehat{r_{u,i}} &= f(u, i|\theta) \\ &= \mu + b_i + b_u + g(u, i|\theta) \end{aligned} \tag{2.3}$$

The formulation of Equation 2.3 is a commonly used solution to the Cold Start problem.

The **Cold Start Problem** is the state in which a RS cannot make relevant predictions for lack of data. In such a case, a new user would typically be recommended the most popular items of the system.

The nature of the $g(u, i|\theta)$ function depends on the type of RS that is used.

We differentiate between two approaches to CF: the Neighborhood-based approach and the Matrix Factorization approach.

2.1.1.1 Neighborhood-based Approach

An intuitive approach to CF is to compare how similar a current user is to other users of the dataset, and to recommend the current user items that similar people have liked. There are several ways to compute a similarity $\text{sim}(u_a, u_b)$ between two users. Some are count-based, such as the Jaccard Similarity, and thus particularly well-suited to situations where the user feedback is binary and/or implicit (click, page visit etc.). Those methods however fail to account for the value of the user interaction with the item (namely, the rating).
The cosine similarity, or any derivative from the inner product, does take the value of the rating into account. Such methods are invariant to the length of vectors u_a and u_b, meaning that they yield a metric that is not biased by the popularity of items or the prolixity of certain users.
When data is highly sparse, Minkowski distances have been shown to produce relevant similarity (G. Jain et al. 2020).

However intuitive, this kind of method has been gradually abandoned in favor of the model-based approaches, yielding faster results at inference time (since the k-nearest-neighbor search can be very time consuming for large user databases).

2.1.1.2 Matrix Factorization

The most common implementation of CF is Matrix Factorization (MF), and especially Non-Negative Matrix Factorization since its success in the Netflix challenge

(Bennett, Lanning, et al. 2007). The idea of MF is to extract continuous latent profiles of dimension Z for both the users and the items from the rating matrix R as shown in Equation 2.4 and Equation 2.5.

$$P = \{\mathbf{p}_u\} \quad \text{s.t.} \quad u = 1, \dots, N_u \in \mathbb{R}^{N_u \times Z} \tag{2.4}$$

$$Q = \{\mathbf{q}_i\} \quad \text{s.t.} \quad i = 1, \dots, N_i \in \mathbb{R}^{N_i \times Z} \quad \text{with} \quad P \times Q^\top \approx R. \tag{2.5}$$

With this formalism, the prediction of the rating of item i by user u is simply the scalar product between $\mathbf{p}_u$ and $\mathbf{q}_i$. Plugging this new formulation in Equation 2.3 yields:

$$\widehat{r_{u,i}} = \mu + b_i + b_u + {\mathbf{p}_u}^\top \mathbf{q}_i. \tag{2.6}$$

The dimension Z of the matrices P and Q is often much smaller than the dimension of the rating matrix R, thus representing the latent profiles in a more compressed latent space.

However, the extracted latent profiles are often qualified as "black boxes" and are hard to interpret. One major challenge for recommendation is to provide explainable recommendations. Besides, MF methods suffer from the Cold Start problem, i.e., they cannot produce relevant personalized results until they have a sufficient amount of interactions. A classic technique to still produce suggestions is to use only the overall mean μ and the biases b_i and b_u of Equation 2.6. On top of that, MF methods have a tendency to overfit, and thus require regularization. A traditional way to regularize MF methods is to add a a regularizing term to the objective function, constraining the norm of the model parameters. Another option is to use latent profiles trained on another task to predict the rating. We present such methods in Section 2.1.4.

2.1.2 Content Based Recommender Systems

Content-based RSs rely on the assumption that a user may be interested in items similar to the ones they liked before. In the case of movie recommendation, one can imagine that if a user rated the Star Wars movies highly, a content-based RS would propose this user with movies that have a similar cast or the same director for instance.

Content Based Recommender Systems work by computing representations of the system's items, computing a similarity measure between items and recommending to a user items that are similar to the ones they liked before.

We distinguish between two types of items: the structured (or tabular) ones and the textual ones. The latter are a crucial inspiration to this work and are thus detailed in Section 2.1.4.

In the case of structured items, such as movies, their representation of an item is based on its attributes. A Content Based Recommender System's item matrix would be of size $I * A$, with I the number of items in the database and A the number of possible attributes. Of course, adding new features via features engineering to the matrix is always possible.

When the items of the system are textual, they can be represented in the system by a variety of methods from Information Retrieval, namely TF-IDF [2] vectorization (Sparck Jones 1988) and Topic Modeling by Latent Dirichlet Allocation (LDA) (Blei et al. 2003).

However, Content-based RSs require a heavy amount of data pre-processing and/or feature engineering in order to work, namely to fill the attributes of each items (Lops et al. 2011).

2.1.3 Hybrid Models

Both the CF and the CB approaches have their pitfalls and their strength. The gain in performances allowed by CF is counterbalanced by their vulnerability to the Cold Start Problem (defined at the beginning of Section 2.1.1) and their opacity. The understandability and intuitiveness of CB models are offset by the cost of the neighborhood similarity computations. For those reasons, recent works have considered hybrid approaches in the hope of combining the qualities of both approaches while nullifying their weaknesses (Thorat et al. 2015).
Most hybrid models are developed with a specific application in mind. This is in part due to the fact that content-based RSs are domain-dependent: one could not build a set of explicit features that fit both movies and restaurants. Netflix is a famous example of hybrid RS (Gomez-Uribe and Hunt 2016). M. Li et al. (2020) combine CF and CB with a complementarity-based method on "Question & Answer" (Q&A) documents to help the users in the process of troubleshooting.

2.1.4 Leveraging Textual information for Recommendation

Leveraging textual information to improve the performances of a recommender system has been a growing topic over the past decade. While the trend aims at using user-generated text, and in particular, the reviews, textual item descriptions

2. Term Frequency / Inverse Document Frequency , detailed in Section 2.1.4

can also be used to compute a representation, namely in a content-based RS context. Using textual information has proven to help to regularize the latent representations and / or the model, as well as providing explainability insights regarding the predictions.

2.1.4.1 Textual Content-Based Recommendation

Content-based RSs have long been interested in deriving a representation from the textual description of an item. It is of particular importance when the items you want to recommend are textual documents, such as web pages for instance. As such, the evolution of textual Content-based RSs has closely followed the of the Information Retrieval and NLP domains. Although textual Content-based RSs are mainly item-centered, the techniques they use to learn item representations can be symmetrically applied to users of a RS, provided the users leave textual traces. For this reason, we detail the most popular document representations of textual Content-based RSs in the following paragraphs.

Bag of Words. The Bag of Words (BoW) representation is one of the earliest and most intuitive ways to vectorize a document. The idea of BoW is that any document d in a set of documents $\mathcal{D}$ with a common vocabulary of size V can be represented by the count of the words it contains.

TF-IDF. TF-IDF stands for Term-Frequency - Inverse Document Frequency. Like in the BoW formulation, a document $d \in \mathcal{D}$ is represented by a vector of size V. However, the TF-IDF vectorization aims at representing documents through the words that differentiates d from the rest of the corpus. To do so, TF-IDF represents a word by the its frequency in in document d, weighted by its frequency across the whole corpus.

This ponderation allows for the emphasis of rare (and thus more discriminative) words in a document, with respect to the whole corpus. Conversely, it is an elegant way to overlook too frequent words.

Pre-processing. It is worth noting that all of the methods presented above are highly sensitive to noise. Thus, the input data should undergo a pre-processing routine involving stop-words removal, case normalization etc.

Note: Lots of works also make use word embeddings models, which are detailed in Section 2.2.

2.1.4.2 Using user reviews in Collaborative Filtering

The use of user textual reviews is also very developed in CF, for several reasons.

Firstly, text is a great way to regularize prone-to-overfitting MF-based models, as well a way to overcome the cold start problem. There is a tendency for CF models to turn to text-based hybrid solutions in order to alleviate the two aforementioned issues.

Secondly, MF-based models often suffer from their opacity, i.e., the fact that their predictions are neither understandable nor explainable. Thirdly, more often than not, the ratings alone are insufficient to accurately represent a user's experience. Considering the interactions makes it hard to capture the complexity of the users' wants and likes.

The Cold Start problem. Hu and Dai (2017) and Dias et al. (2017b) both propose ways to use user reviews to alleviate the Cold Start problem in CF approaches. Hu and Dai (2017) are using text reviews to address the cold start problem by embedding item reviews and building matrix factorization-based Bayesian personalized ranking models on top of them. Dias et al. (2017b) show that using only two reviews on an item yields better performances than predicting a user mean for the item cold start problem.

Opacity and Complexity. We mentioned earlier that CF models that only use interactions can fail to understand the users tastes in depth. Let us illustrate the problem with an example. If we take the case a restaurant, user A may have given a restaurant 4 stars because user A was very appreciative of the food, and totally indifferent to the price or the service. If a user B on the other hand, is on a budget and expects great service, then the rating of user A is of no interest to them. This why users often go further than the average rating of a place and look for the previous reviews. Ganu et al. (2009) and Ganu et al. (2013) address this issue by proposing to predict the sentiment associated with the topic of a sentence (food, service, ...) within a review. Such models are often referred to as *multi-aspect recommender systems*. The review sentiment prediction allowed for a gain in performance on the rating prediction task.

J. McAuley and Leskovec (2013) go even one step further by using a topic modeling approach to textual reviews in order to better understand and explain the users' tastes as well as the model's suggestions. Topic Modeling is a technique that allows for the discovery of subjects (or topics) in a document. J. McAuley and Leskovec (2013) use this technique user reviews to better understand the users' *rating dimensions* (that is to say, what aspects of an item was liked by the user), as well as a way to regularize their model and tackle the Cold Start Problem. They

define a document d_u as the set of reviews written by a user u and use LDA (Blei et al. 2003) to compute the topic distribution of each document which encodes the fraction of words in d_u that discuss each of the K topics. Their objective function is thus composed of the prediction rating error (similar to Equation 2.6) and the likelihood of the corpus, with respect to the set of topics and the rating and topic parameters. Almahairi et al. (2015) compare LDA-based models such as the one of J. McAuley and Leskovec (2013) to a model combining distributed bag-of-words together with matrix factorization for predicting ratings, improving the MSE on the Amazon Reviews dataset (J. McAuley et al. 2015b).

J. McAuley and Leskovec (2013) and Almahairi et al. (2015)'s works can be seen as pioneering for this thesis, as they address the problem of user understanding by leveraging their textual traces, as well as propose *justifications* for their model's suggestions.

Catherine and Cohen (2017) also propose to enrich user-item interactions with user reviews. They do so by predicting the rating of item B i_B by user A u_A, r_{AB}, by transforming textual representations of user u_A and item i_B into a latent representation of rev_{AB}, the textual review of i_B by u_A. It is a very interesting approach of MF in the way that Catherine and Cohen (2017) computes latent representations of both i_B and u_A and then mix those to represent the interaction between i_B and u_A. The latent representation of user u_A (respectively, item i_B) is built by processing all reviews written by u_A (respectively about i_B) without rev_{AB} through a Convolutional Neural Network (CNN) text processor. The CNN text processor is aligned at training time with a more classical rating-prediction model. Their approach is an original way to combine Deep Learning architectures to MF as well as an elegant way to face the Cold Start Problem, since they can infer the representation of rev_{AB} even if it does not exist.

Note: Lots of works also make use word embeddings models, which are detailed in Section 2.2.

2.1.5 Training Objectives and Evaluation Criteria in Recommendation

If the methods to increase performances and leverage textual reviews are evolving, the same can be said about evaluation methods in recommendation (Bellogín and Said 2018). There is an ongoing discussion about separating the training objective(s) of an RS (and thus, its modeling of user and/or items) from the evaluation metrics. These research directions are motivated by two factors:

- Training a model to predict ratings alone is not enough to properly represent the users of a system,
- There are many other qualities one can expect from an RS besides correctly predicting a rating.

The following section covers the traditional metrics used for the evaluation of RSs, as well as emerging metrics.

2.1.5.1 Traditional Metrics and their Limitations

Mean Square Error and derivatives. The Mean-Squared Error (MSE) metric has long been the preferred method for evaluating RSs. It consists in computing the average squared error between a predicted rating $\widehat{r_i}$ and a target rating r_i. Both the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) are also widely used for evaluation purposes.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(r_i - \widehat{r_i})^2$$

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|r_i - \widehat{r_i}|$$

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(r_i - \widehat{r_i})^2}{N}}$$

The MSE and associated metrics are often the training objective of a model, as well as its evaluation metric.

They are however insufficient both as a training objective and as an evaluation metric in the way that they are unstable (the sum over the embeddings' dimensions fails to account for fine-grain feature alignment) and are simply not a good estimation of the RS performances, especially in the case of implicit feedback RSs.

We will refer to this situation as the Regressive Formulation.

Recall, Precision, F1-score. Accuracy metrics are not sufficient to evaluate the quality of a RS.

A classic alternative to the Regressive Formulation is Classification. In this setting, we aim at finding what rating corresponds to the current (user, item) couple by classifying this pair into the class that corresponds to the actual rating. We thus use classification-related metrics: Recall, Precision, F1-score.

The Recall metric informs us regarding how many of the items highly rated (or consumed) by a user C_u were actually recommended to that user (that is to say, how many of those items where in the recommendation list R_u). Conversely, the Precision metric measures how many items in R_u were indeed consumed by a user. Finally, the F1-score is a metric that links both the Recall and the Precision metrics.

$$\text{Recall} = \frac{|C_u \cap R_u|}{C_u}$$
$$\text{Precision} = \frac{|C_u \cap R_u|}{R_u}$$
$$\text{F1-score} = 2 \times \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}$$

Limitations. The aforementioned metrics present two main limitations:

1. They are relevant assuming that the data are missing completely at random (MCAR assumption),
2. They merely account for one quality of a RS, which one could call *Accuracy*.

Regarding point 1, several works showed that the MCAR assumption rarely stands (Steck 2010). Indeed, one can imagine that if a user disliked the first Harry Potter movie, then it is quite unlikely they will bother with the seven others. Symmetrically, the observed ratings are biased both by the user's initial tastes (a user that liked the series of *The Lord of the Rings* is likely to try out the *The Hobbit* series) and the previous RS's suggestions. It becomes then apparent that there is implicit feedback from the user in the ratings, and the regressive formulation shuns it entirely.

Furthermore, aside from bypassing important information, the regressive formulation tends to over-specialize the recommendations, and thus create the Echo Chambers mentioned in Section 2.1's introduction.

2.1.5.2 Alternative Training & Evaluation Methods

Overcoming the traditional MSE objective for training is a necessary step in order to capture complex interactions of the user with the system. The more complex those representations get, the bigger the need for more complete evaluation techniques grows.

Ranking. Ranking is not exactly a new approach to RSs evaluation per se, but it is worth mentioning in this section because it does provide a different approach to RS evaluation. It is an interesting way to quantify how far the preferred item is in the list of recommendations. Ranking metrics, such as the Mean Reciprocal

Rank (or MRR) focus on the precision of a RS rather than its accuracy (Valcarce et al. 2020). MRR can be understood as the average number of iteration in the recommendation list it takes to obtain a relevant item. The Mean Average Precision (or mAP) proposes a metric that weighs the head of the suggestion list more than its tail, thus putting an emphasis on the need to get relevant items at the top of the list. In the case where we have a relevance score for each item on the suggested list, the Discount Cumulative Gain (DCG) gives us a score that is weighed by the relevance score and the position of the item in the list (the relevance score is reduced logarithmically with respect to the position of the item on the list). Its normalized version (nDCG) divides the previous score by the "ideal discounted cumulative gain", which is the discounted cumulative gain across all the relevant documents of the corpus.

Such evaluation is crucial in cases of suggested translations or browser queries for instance.

Ranking can also be used as a way to train RS. The Information Retrieval literature beams with examples of such tasks and training objective (T.-Y. Liu 2009).

Training a Recommender System to Generate Text. Section 2.1.4 highlighted the increased usage of textual reviews as the inputs of RSs, but there is also a trend to train RS to generate text (and in particular, reviews).

Dong et al. (2017) propose a Long Short-Term Memory (LSTM)-based model that learns to generate reviews of an item based on its attributes. Ni et al. (2017) generate user reviews with a character-level Recurrent Neural Network (RNN) model. Radford et al. (2017) set the state-of-the-art on the sentiment classification with a byte-level RNN model.

Serendipity. Our growing understanding of RS leads us to look for even better qualities in an RS than simple accuracy. Silveira et al. (2019) describe up to 6 categories of qualities to evaluate in an RS, including Serendipity, Novelty and Diversity. A serendipitous recommendation in a RS can be described as "a happy surprise". This term both englobes the concept of Utility ("happy") and Unexpectedness ("surprise"). Serendipity has been and remains at the heart of RS evaluation discussions (Ziarani and Ravanmehr 2021; Ge et al. 2010)

Many other aspects of Recommendation Evaluation are being studied to improve the quality of RS, such as fairness (Anelli et al. 2021), anti-relevance (Sánchez and Bellogín 2018a), Novelty and Diversity (Vargas and Castells 2011; Castells et al. 2015) and so on.

2.1.6 Conclusion

UR is at the heart of research in Recommendation. The intuitive neighborhood-based approaches of CB tend to give way to the more compact yet abstract representations of CF. Both approaches can be combined into Hybrid models, that are mostly application-driven and applied in an industrial context, as a way to address the cold start problem while still achieving good performances.

However, the very notion of good performances have been the topic of many discussion and research in Recommendation. It is understood that accuracy on score prediction alone cannot be seen as a sufficient evaluation metric, even less when the number of RS in our daily lives is growing.

Using textual reviews in RS represents a major gain for performances, model regularization and even understandability of predictions. The interest of the Recommendation research community regarding textual data keeps on growing along with the advances of NLP.

The contribution we propose in Chapter 3, Hierarchical Recurrent Attentive Neural Network for Recommendation (HRAN), relies on both CF methods and the leveraging of textual reviews. In this model, we jointly train a Multi-Layer Perceptron (MLP) and a Bidirectional Recurrent Neural Network (BiRNN) [3] on both the ratings and the reviews to learn what item a user will like, and *why* - thus proposing an explanation for each personalized suggestion. We propose a way to overcome the limitations of the regression training by associating the MSE metric (for the MLP) to a Negative Log-Likelihood (NLL) loss (for the BiRNN), and jointly optimizing them. Once trained, the BiRNN is used a personalized language model for each user. The architectures and models that made it possible are detailed in the following section. In Chapter 4 we present the work at the crossroads of CB and CF. In CB, the models work with textual descriptions of items and are devoid of personalization. Conversely, CF models rely on interactions between users and items to propose personalized recommendations. Chapter 4 proposes a way to leverage interactions between the users and the items that are textual (professional experiences), and not numerical (ratings).

3. Both those architectures are duly detailed in Section 3.1

2.2 NLP and Leveraging User-Generated text for User Representation

NLP methods did not only revolve around leveraging text reviews for Recommendation. Instead, they gained more and more importance as the number of tasks related to natural language grew.

The evolutions of NLP, detailed in Section 2.2, are of particular interest to this work because we aim at representing users through the textual traces they leave. It is our thesis that those traces are semantically and syntactically unique to and descriptive of a user. In the case of RS, the textual traces are often reviews on a item. The topics mentioned in such reviews are characteristic of the user's interests and preferences.

We also think that, beyond the semantics, the syntax and structure of the text is also unique to a user. Symmetrically to the way some users tend to rate items with high marks, some users tend to speak abundantly and in high praise, while others are more neutral. We present a personalized reading model based on those assumptions in Chapter 3.

NLP is a vast domain that aims at leveraging human-generated text (or Natural Language) to solve a given task. Applications and examples of NLP tasks are numerous: chatbots, question-answering, named-entity-recognition, sentiment analysis, text generation *etc*. Note that the latter, being a crucial focus for this thesis, will be covered in detail in Section 2.3.1.

Semantic over syntax. The past decades have witnessed an evolution from rule-based models to Machine Learning models, then Deep models, forsaking syntax one step at a time along the way. This change of paradigm naturally triggered a discussion about evaluating those new methods, leaning towards the evaluation of the representation's richness and versatility.

Multi-scale: from words to sequences. Natural Language Processing is an inherently multi-scale domain, and this aspect poses a real aggregation problem. The gaps aren't completely filled yet, even though recent architectures tend to bridge character/subword-level to sentence-level end-to-end. In our work, we have an additional level of granularity, since we aim at aggregating sentences into a user.

Representation Learning in NLP. The rise of CF has led to a new paradigm of user representation: RL. The success of MF, powered by Neural Network (NN), over Neighborhood-based approaches led to more compressed but less understandable

representations of items and users. The growing popularity of NNs allowed RS to shift from human-friendly representations to continuous, abstract and more meaningful ones. In parallel, the same drift operated in NLP, with word and sequence representations becoming continuous and existing in abstract latent spaces. Nowadays, RL is at the heart of NLP, from Language Modeling to Question Answering. It took a particular resonance with the advent of generation-pre-trained models such as Bidirectional Encoder Representations from Transformers (BERT). Indeed, the success of the Transformer brought generative models at the forefront of the NLP architectures, allowing a powerful pre-training of models in an unsupervised way.

The following section will cover technical aspects of NLP methods evolution. We will start by reviewing the evolution of word embeddings. Then, we will go over the different models and methods for sequence modeling. Finally, we will detail the evolution of Attentions mechanisms, and their contributions to the field.

2.2.1 Language Models and Word Embedding Models

Text representation has long been performed at the document level in a bag-of-words setting. However, Mikolov et al. (2013b) introduce distributed word representations with the Word2Vec model, enabling us to predict words in a local context and thus opening the way for meaningful continuous word embeddings and text generation applications. It is a Language Model that relies on two key principles: the Skip-Gram model and the Negative Sampling procedure.

2.2.1.1 Language Model

The task of Language Modeling can be defined as the task of predicting the next word given a context (Mikolov et al. 2010). In other words, it is the task of learning the joint probability of a word sequence to appear in a text (Y. Li and T. Yang 2017). Formally, the probability P of a given word sequence $(w_1, ..., w_N)$ of length N if defined as the product of conditional probability on each word w_i:

$$P(w_1, ..., w_n) = \prod_{t=1}^{N} P(w_t \mid w_1, ..., w_{t-1}) \tag{2.7}$$

Note that we used the term "word" for simplicity, but the term "token" as it is a more accurate denomination, given the variety of granularity in language models (sub-words, n-grams etc).

2.2.1.2 The Skip-Gram Model

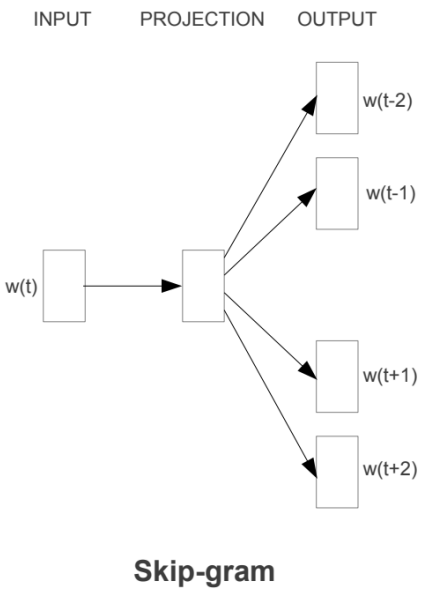

Figure 2.3 – **Illustration of the Skip-gram model**. As presented by Mikolov et al. (2013b)

The skip-gram model learns word representations that can be used to infer their context, as illustrated on Figure 2.3. The model is trained to maximize the average log of Equation 2.8.

$$\frac{1}{N}\sum_{t=1}^{N}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j} \mid w_t) \tag{2.8}$$

2.2.1.3 Negative Sampling

The Negative Sampling objective is introduced as an alternative to the hierarchical softmax and a simplification of the Noise Contrastive Estimation (NCE).

In practice, it is an objective that enforces the fact that words belonging to the same context window should have representations that are close (first term of Equation 2.9), while words that do not appear together frequently should have representations that are far from each other (second term of Equation 2.9)(Goldberg and Levy 2014) .

The Negative Sampling Objective is formally defined by Equation 2.9. For simplification purposes, let us rewrite the words within the vicinity of w_t as w_I. This yields:

$$p(w_t \mid w_I) = \log \sigma(\mathbf{v}'_{w_t} \top \mathbf{v}_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-\mathbf{v}'_{w_i} \top \mathbf{v}_{w_I}) \right] \tag{2.9}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.10}$$

The form of $p(w_t \mid w_I)$ of Equation 2.9 is then plugged into Equation 2.8.

Note that the "negative examples" i.e., the k words that do not appear in the vicinity of w_t are randomly drawn according to a unigram distribution raised to the power $\frac{3}{4}$ (Mikolov et al. 2013b).

The original paper also introduced several parameters to learn their word representations, such as subsampling of frequent words and dynamic context-window sizing.

The word representations learned by maximizing the objective defined by equations Equation 2.8 and Equation 2.9 present interesting properties. In particular, the latent space in which they evolve presents some regularities illustrated in Figure 2.4.

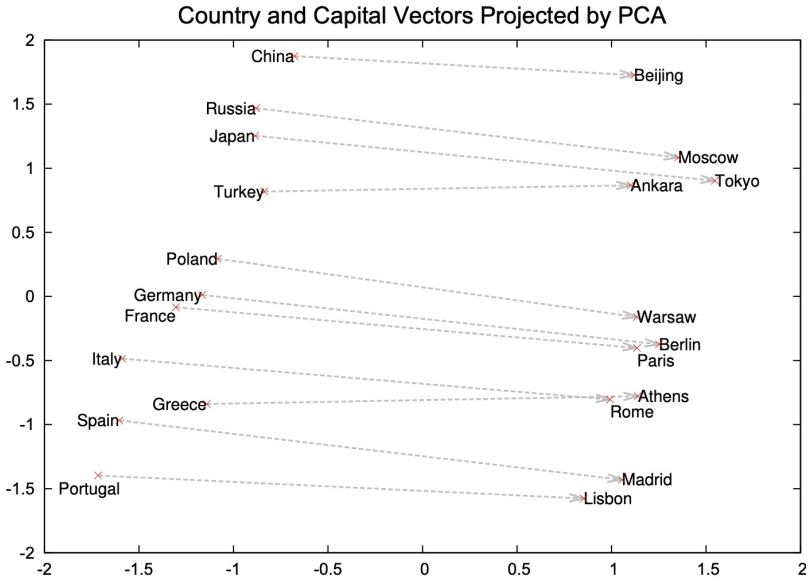

Figure 2.4 – **PCA projection of words representing countries and their capital**. Illustration of the original paper Mikolov et al. (2013b)

2.2.1.4 Sub-word embedding

However, the word2vec algorithm suffers a major limitation, which is that it cannot handle out-of-vocabulary words, and is thus helpless to represent new

words. This particular limitation has namely been addressed by Bojanowski et al. (2016) in the form of sub-word embedding.

FastText. Bojanowski et al. (2016) define subword embeddings by representing a word as the sum of the n-grams that composes it. They give the example of the word "where" that can be decomposed into the set of n-grams : "<wh, whe, her, ere, re >, where" for $n = 3$. Note that the special symbols <and >respectively denote the beginning and and the end of the word, and that the word itself it contained into its set of n-grams. The scoring function of a word w and it context c then becomes:

$$s(w, c) = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g^\top \mathbf{v}_c \tag{2.11}$$

with $\mathcal{G}_w$ the set of n-grams that represents w.

This formulation in subwords allows for the handling of out-of-vocabulary words as well as a reliable representation of rare words and has become widely used.

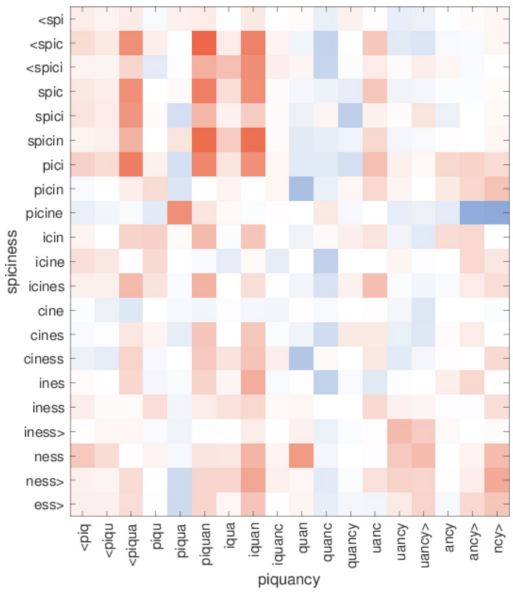

Figure 2.5 – **Illustration of the similarity between character n-grams in out-of-vocabulary words**. As presented by Bojanowski et al. (2016). The word on the x-axis is out-of-vocabulary, while the word on the y-axis is not.

Byte-Pair Encoding. Despite Bojanowski et al. (2016)'s intuitive algorithm, nowadays's preferred approach is the 1994 Gage (1994) Byte-Pair Encoding (BPE) algorithm.

The BPE data compression algorithm works by iteratively replacing common pairs of consecutive bytes (or characters) by a byte that does not appear in that data. This process is repeated until the desired vocabulary size (that is, the number of bytes) is reached.

The BPE algorithm is namely used in the BERT architecture.

Word embeddings permitted the semantic representations of textual data with a richness that we would not achieve before. Those semantic representations have allowed, for instance, Dias et al. (2017a) to learn a meaningful latent space that they leverage to recommend jobs to a user, given their previous jobs. More generally, semantic word representations allow for the extraction of information that would otherwise have been lost in the noisiness of textual data.
Word embeddings are essential building blocks for representing sentences or documents, but they do not directly allow for sequence representation.

2.2.2 Sequence Modeling

The next granularity level in NLP representations is the sentence-level. The main challenge of sequence modeling is to deal with the possibly variable lengths of the sentences. This issue has been mostly investigated through the prism of Neural Machine Translation (NMT).

2.2.2.1 Recurrent Cells

There are several techniques to represent sequences but the most widespread are RNN and in particular the LSTM (Hochreiter and Schmidhuber 1997) implementation and its simplified version, Gated Recurrent Unit (GRU) (Cho et al. 2014). Unlike traditional Feed-Forward Neural Networks, in RNN the output of the network at time t, $\mathbf{y}_t$, is a function of both the input at time t, $\mathbf{x}_t$, and the previous input's representation, $\mathbf{h}_{t-1}$. That makes them naturally well-suited for represented temporal sequences such as sentences. However, RNNs suffer from the vanishing or exploding gradient problem and thus struggled to represent long sequences. To address this issue, Le et al. (2015) proposed a simple initialization coupled with Rectified Linear Unit (ReLU) activations that they found presented similar performances as LSTMs.

The latest formulation of an LSTM (Gers et al. 2000) is presented in Equation 2.12. LSTM are typically composed of a cell c (also referred to as "memory cell"), an input gate i, an output gate o and a forget gate f.

4. https://github.com/dvgodoy/dl-visuals, image licensed under CC 4.0: https://creativecommons.org/licenses/by/4.0/

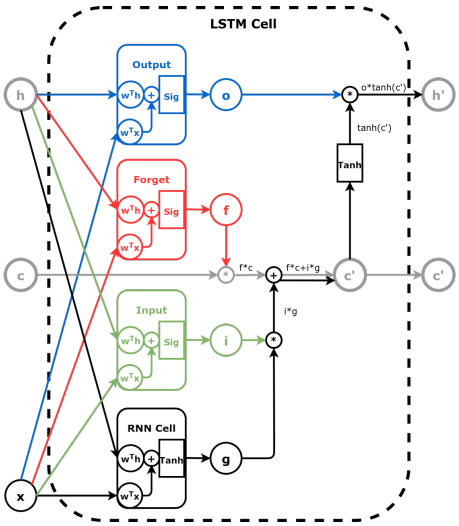

Figure 2.6 – **Illustration of a LSTM Cell**. Image by dvgodoy [4]

$$
\begin{aligned}
\mathbf{f}_t &= \sigma_g\left(W_f\mathbf{x}_t + U_f\mathbf{h}_{t-1} + \mathbf{b}_f\right) \\
\mathbf{i}_t &= \sigma_g\left(W_i\mathbf{x}_t + U_i\mathbf{h}_{t-1} + \mathbf{b}_i\right) \\
\mathbf{o}_t &= \sigma_g\left(W_o\mathbf{x}_t + U_o\mathbf{h}_{t-1} + \mathbf{b}_o\right) \\
\tilde{\mathbf{c}}_t &= \sigma_c\left(W_c\mathbf{x}_t + U_c\mathbf{h}_{t-1} + \mathbf{b}_c\right) \\
\mathbf{c}_t &= \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tilde{\mathbf{c}}_t \\
\mathbf{h}_t &= \mathbf{o}_t \circ \sigma_h\left(\mathbf{c}_t\right)
\end{aligned}
\tag{2.12}
$$

The GRUs was introduced in 2014 by Cho et al. (2014). The minimal gated unit equation is presented in Equation 2.13. Unlike the LSTM cells, minimal GRU cells only present a forget gate f.

$$
\begin{aligned}
\mathbf{f}_t &= \sigma_g\left(W_f\mathbf{x}_t + U_f\mathbf{h}_{t-1} + \mathbf{b}_f\right) \\
\hat{\mathbf{h}}_t &= \phi_h\left(W_h\mathbf{x}_t + U_h\left(\mathbf{f}_t \odot \mathbf{h}_{t-1}\right) + \mathbf{b}_h\right) \\
\mathbf{h}_t &= \left(1 - \mathbf{f}_t\right) \odot \mathbf{h}_{t-1} + \mathbf{f}_t \odot \hat{\mathbf{h}}_t
\end{aligned}
\tag{2.13}
$$

Both LSTM and GRU have been proven to have similar performances (Chung et al. 2014; Greff et al. 2017).

From now onwards, when the term RNN is used, it will by default refer to a RNN consisting of LSTM cells.

2.2.2.2 Bidirectional RNN

A widespread variation of RNNs are Bidirectional Recurrent Neural Network (BiRNN)s. Originally presented by Schuster and Paliwal (1997), BiRNNs consist

of two independent[5] RNNs, going in opposite directions. The forward RNN , $\overrightarrow{f}$, encodes a sequence $\mathbf{s} = (\mathbf{x}_1, ..., \mathbf{x}_T)$ from left to right, thus outputting a sequence of forward hidden states $(\overrightarrow{\mathbf{h}_1}, ..., \overrightarrow{\mathbf{h}_T})$. Conversely, the backward RNN $\overleftarrow{f}$ encodes s from right to left, outputting backward hidden states $(\overleftarrow{\mathbf{h}_1}, ..., \overleftarrow{\mathbf{h}_T})$. The annotation $\mathbf{h}_t$ of each token $\mathbf{x}_t$ is the (possibly transformed) concatenation of both annotations $\overrightarrow{\mathbf{h}_t}$ and $\overleftarrow{\mathbf{h}_t}$.

$$\mathbf{h}_i = \left[\overrightarrow{\mathbf{h}_t}; \overleftarrow{\mathbf{h}_t}\right]$$

This representation ensures that the annotation $\mathbf{h}_i$ contains information about the tokens both following and preceding $\mathbf{x}_t$ (Bahdanau et al. 2015).

BiRNNs have been widely used in applications such as speech recognition (Graves et al. 2013) and, more recently, Language Modeling (Peters et al. 2018).

2.2.2.3 Contextual Word Embeddings: ELMo

More recently, Peters et al. (2018) proposed a more versatile type of word embeddings: the deep contextualized word embeddings. Contrary to Mikolov et al. (2013b) and Bojanowski et al. (2016), Peters et al. (2018)'s model Embeddings from Language Models (ELMo) word representations are a function of the entire input sentence.

More precisely, ELMo's embeddings are a learned linear combination of the internal layers of a Bidirectional Language Model (biLM) (BiRNN are detailed in Section 2.2.2.2). Note that in this equation, the forward and the backward LSTMs share some weights and are both tied to the token representation parameters Θ_x and the Softmax Layer parameters Θ_s. This process is illustrated on Figure 2.7.

$$\begin{aligned}\mathcal{L}_{biLM} = \sum_{t=1}^{N} \Big(&\log p\left(w_t \mid w_1, \ldots, w_{t-1}; \Theta_x, \vec{\Theta}_{LSTM}, \Theta_s\right) \\ &+ \log p\left(w_t \mid w_{t+1}, \ldots, w_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s\right)\Big)\end{aligned} \tag{2.14}$$

The representations are learned by a 2-step process. First, the biLM is pre-trained on a large corpus of text by optimizing the objective of Equation 2.14.

Then, ELMo computes R_t, the set of $2L + 1$ representations of token w_t, with L being the number of layers of the biLM. An example is shown on Figure 2.8.

5. In the sense that they do not share weights.

6. https://github.com/dvgodoy/dl-visuals, image licensed under CC 4.0: https://creativecommons.org/licenses/by/4.0/

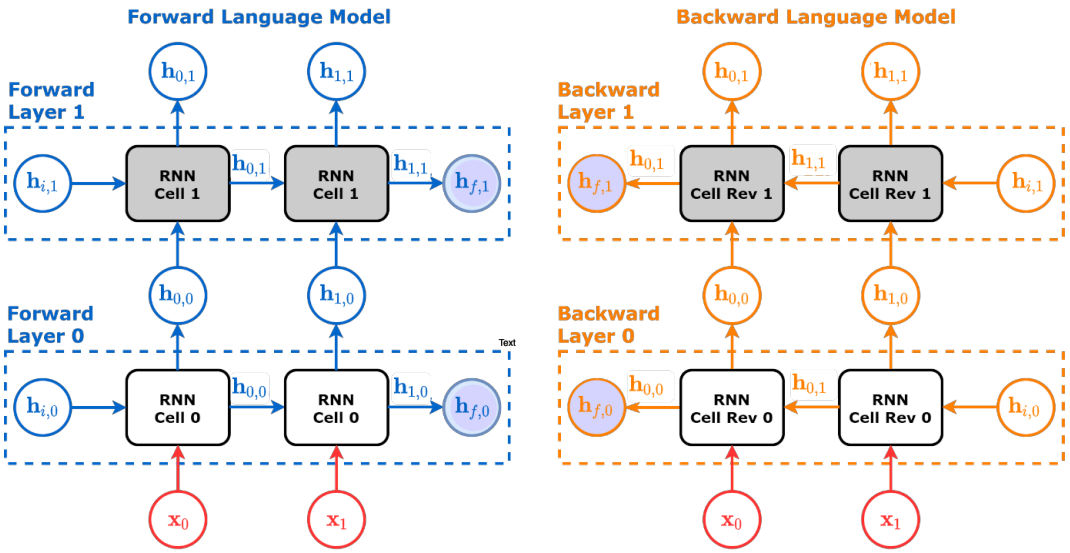

Figure 2.7 – **Illustration of an ELMo's biLM**. Image by dvgodoy [6].
In this Figure, the forward LSTM is presented in blue, and the backward one in orange. The $\mathbf{h}_{i,0}$ and $\mathbf{h}_{f,0}$ represent the initial and final hidden states of the LSTM.

$$\begin{aligned}\mathbf{R}_t &= \left\{\mathbf{x}_t^{LM}, \overrightarrow{\mathbf{h}}_{t,j}^{LM}, \overleftarrow{\mathbf{h}}_{t,j}^{LM} \mid j = 1, \ldots, L\right\} \\ &= \left\{\mathbf{h}_{t,j}^{LM} \mid j = 0, \ldots, L\right\}\end{aligned} \tag{2.15}$$

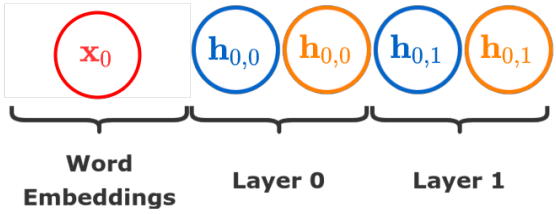

Figure 2.8 – **Illustration of an ELMo's set of representation**. Image by dvgodoy [7].
This Figure represents the set of representation $\mathbf{R}_0$ of the first token $\mathbf{x}_0$ for a number of layer $L = 2$.

So, after Equation 2.15, each token is associated to a set of $L+1$ representations, since the j^{th} forward LSTM representation $\overrightarrow{\mathbf{h}}_{t,j}^{LM}$ is concatenated to the j^{th} backward LSTM representation $\overleftarrow{\mathbf{h}}_{t,j}^{LM}$: $\mathbf{h}_{t,j}^{LM} = \left[\overrightarrow{\mathbf{h}}_{t,j}^{LM}, \overleftarrow{\mathbf{h}}_{t,j}^{LM}\right]$, for $j \in [0, L]$. Note that if $j = 0$, then $\mathbf{h}_{t,j}^{LM} = \mathbf{x}_t^{LM}$, the initial token representation.

7. https://github.com/dvgodoy/dl-visuals, image licensed under CC 4.0: https://creativecommons.org/licenses/by/4.0/

Then, this set of token representation R_t is collapsed into a single vector according to a downstream task.

$$\mathbf{ELMo}_t^{\text{task}} = E\left(\mathbf{R}_t; \Theta^{\text{task}}\right) = \gamma^{\text{task}} \sum_{j=0}^{L} s_j^{\text{task}} \mathbf{h}_{t,j}^{LM} \tag{2.16}$$

The fine-tuning of the token representation, illustrated in Equation 2.16, consists in aggregating the different layer representations in R_t in a weighted sum, which weights s_j^{task} depend on the selected task. The term γ^{task} allows the model to scale the entire $\mathbf{ELMo}_t^{\text{task}}$ vector, which can be useful if the downstream task expects values within a certain range for instance.

This approach allows for the modeling of complex characteristics of word use and for the disambiguation of variable linguistic contexts (typically, polysemy). It is interesting to note that the ELMo model treats all tokens as possibly polysemic.

While ELMo proposes word embeddings, it is important to highlight that the representation they propose depend on the previously words, and is thus close to sequence modeling. Additionally, Peters et al. (2018) propose methods to include an ELMo module as an integrated part of existing sequence-modeling architectures for tasks like SQuAD.

Like word embeddings, the advances of Sequence Modeling changed the way we could represent users in Recommendation by removing the technical barrier of word aggregation. Rather than a trivial mean of aggregation (like the average), Sequence Modeling allowed for the representation of sequential information, and thus, a better user representation (Smirnova and Vasile 2017; Hidasi et al. 2016).

2.2.3 Attention Mechanism

Attention is a mechanism that has really been a game-changer in NLP. On top of allowing improvements of the state-of-the-art on numerous tasks, namely by acting as a regularization, Attention also introduced a new form of input/feature selection and thus, of explainability. We leverage this property in Chapter 3.

Early Formulation. Recurrent sequence modeling methods allow for the encoding of sentences into a fixed-length vector. One can either pool the encoder's hidden states over time (generally, max-pooling is used) or define the sequence's representation as the last hidden state outputted by the encoder. These methods however present the shortcoming of decreasing performances when the input sentence is very long, especially if it is longer than the sequences seen at training time.

Bahdanau et al. (2015) addressed this issue by proposing an extension of the encoder-decoder (this architecture will be detailed in Section 2.3.1) model that learns to ″align and translate". The main idea of the extension is to find the most relevant position in the input sentence with regard to the token that has just been outputted. This makes the model predict the next token by taking into account all the previously outputted tokens and the context vector associated to the positions where the information is concentrated.

Formally, the context vector $\mathbf{c}_t$ (Equation 2.17) associated to the t-th token of the input sequence is a weighted sum of the encoder's previous hidden-states (also called *annotations*) $\mathbf{h}_j^e$. The weight α_{tj} (Equation 2.18) of hidden-state $\mathbf{h}_t^e$ is the softmax over the alignment scores e_{tj}. This alignment score e_{tj} (Equation 2.19) is a function of both the encoder's hidden-state at step j and of the decoder's hidden-state at step $t-1$.

$$\mathbf{c}_t = \sum_{j=1}^{N} \alpha_{tj} \mathbf{h}_j^e \tag{2.17}$$

$$\alpha_{tj} = \frac{\exp\left(e_{tj}\right)}{\sum_{k=1}^{N} \exp\left(e_{tk}\right)} \tag{2.18}$$

$$e_{tj} = a(\mathbf{h}_{t-1}^d, \mathbf{h}_j^e) \tag{2.19}$$

In Bahdanau et al. (2015), the scoring function a is a learned feed-forward neural network, trained jointly with the other components of the model. Later implementations will give a the form of a dot product.

The context vector $\mathbf{c}_t$ is then fed to the decoder's RNN and is used to compute the next hidden state. This mechanism can be understood as what the model should be "focusing on" in the input sentence to predict the next token. It helps the model cope with longer sentences. Note that this early formulation of attention is called "alignment" in the original paper, and is sometimes referred to as "additive attention".

2.2.4 Self-Attention

While Attention's early formulation by Bahdanau et al. (2015) in 2016 was already a promise of better performance, the 2017 Vaswani et al. (2017) "Attention is All you Need" proposed to let go of RNNs altogether and introduced the now-widespread concept of *self-attention*. The success of the multi-head self-attention and the Transformer architecture gave rise to a family of state-of-the-art models: BERT.

The creation of the self-attention mechanism in 2017 by Vaswani et al. (2017) was motivated by two things. Firstly, it aimed at leveraging the performances of classical attention. Secondly, most sequence modeling architectures were based on RNNs, making the parallelization of computation impossible. Self-attention addresses this problem by removing recurrence from its architecture, allowing for a great gain of computation time.

Vaswani et al. (2017) describe self-attention using the same notations as those of the memory networks (Sukhbaatar et al. 2015). This formulation presents attention as a function of a query and a set of key-value pairs. Equation 2.20 illustrates the attention function as the product of :

- a softmaxed scaled dot-product between a matrix of query Q, a matrix of keys K and

- V, the matrix of values of dimension d_k.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.20}$$

$$\text{with} \quad \text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{2.21}$$

In self-attention, the nature of the query, keys and values depends on where we are at in the architecture.

Let us illustrate the mechanism of self-attention at encoding time. Q, K, V are the representations of the input (in this case, the tokens' embeddings Z_w) projected by the query, keys and value matrixes, respectively U_Q, U_K and U_V (along with their possible biases). Note that U_Q, U_K must be the same dimensions because of the scaled dot product, but not U_V.

$$Q = U_Q \times Z_w + \mathbf{b}_Q \quad \text{with} \quad U_Q \in \mathbb{R}^{d_w \times d_{\text{model}}} \tag{2.22}$$

$$K = U_K \times Z_w + \mathbf{b}_K \quad \text{with} \quad U_K \in \mathbb{R}^{d_w \times d_{\text{model}}} \tag{2.23}$$

$$V = U_V \times Z_w + \mathbf{b}_V \quad \text{with} \quad U_V \in \mathbb{R}^{d_{\text{model}} \times d_v} \tag{2.24}$$

In Equation 2.22, d_w is the dimension of the token's embeddings, d_{model} the dimension of the model's representation and d_v the dimension of the value.

2.2.4.1 Multi-Head Attention

Vaswani et al. (2017) extend their concept of self-attention by learning not one linear projection but h: one for each head. The outputs of the h heads are then concatenated and linearly projected with a ReLU activation.

$$\text{MultiHeadAttention}\ (Q,K,V) = \begin{bmatrix} \text{head}_1(Q,K,V) \\ \text{head}_2(Q,K,V) \\ \vdots \\ \text{head}_h(Q,K,V) \end{bmatrix} W^O \tag{2.25}$$

$$\text{with} \quad \text{head}_i(Q,K,V) = \text{Attention}\left(QW_i^{(q)}, KW_i^{(k)}, VW_i^{(v)}\right) \tag{2.26}$$

Each head $head_i$ is composed of learned projections matrices:

- $W_i^{(q)} \in \mathbb{R}^{d_{\text{model}} \times d_k}$, that projects query Q into a representation subspace of dimension d_k,
- $W_i^{(k)} \in \mathbb{R}^{d_{\text{model}} \times d_k}$, that projects the keys K into the same subspace,
- $W_i^{(v)} \in \mathbb{R}^{d_{\text{model}} \times d_v}$, that projects the values V into a representation subspace of dimension d_v, and finally
- W^O that projects the concatenated head-attention back into the model's representation space of dimension d_{model}.

The number of heads h is traditionally 8, and and each head's dimension is a subspace of the model's representation space, with $d_k = d_v = d_{\text{model}}/h$. The parallelization of attention computation in different representation subspaces allows for a gain in performances while keeping the computation time similar to the of a single-head attention of full dimensionality.

Note: the matrix $W_i^{(q)}$ of Equation 2.25 is not to be confused with the matrix U_Q of Equation 2.22. The U_Q matrix projects the input embeddings into the model's latent space (of dimension d_{model}), resulting in Q, while $W_i^{(q)}$ reprojects Q in a subspace (of dimension d_k) of the model's latent space.

With the self attention mechanism, the sequential aspect handled by the RNNs is almost gone; it is only enforced by masking at decoding time. This makes the encoding of long sequences easier, as all the information is encoded "at once" and does not get lost along the tokens.

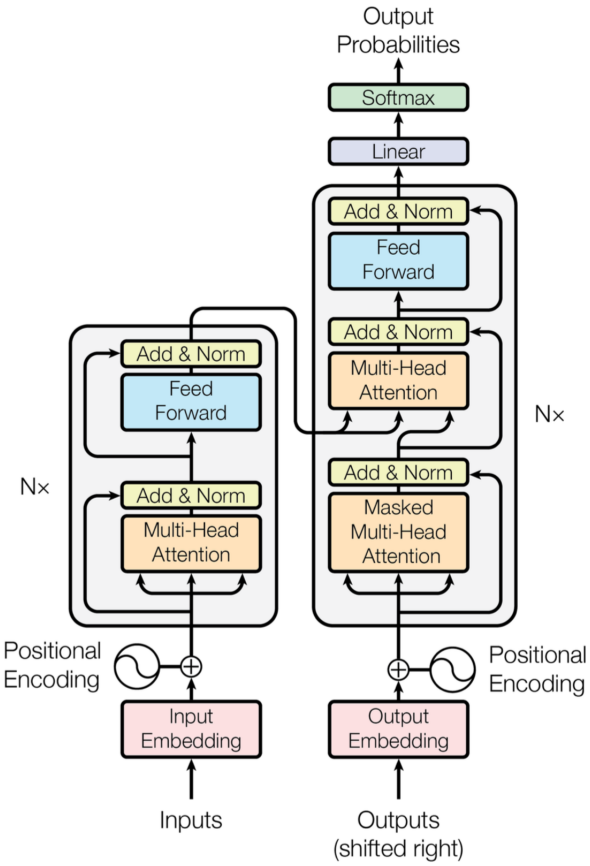

Figure 2.9 – **Illustration of the Transformer Architecture**. From Vaswani et al. (2017).

2.2.4.2 The Transformers

Vaswani et al. (2017) applied their concept of multi-head self-attention to a new family of models called the Transformers. Those models's architecture, presented on Figure 2.9, rely solely on self-attention to model text, abandoning non-parallelizable RNNs.

Figure 2.9 reveals three places where multi-head attention is used: in the encoder (left), in the decoder (bottom right), and in the "encoder-decoder" attention part (middle right).

In both the encoder and the decoder, the multi-head attention modules compute a self-attention. That is to say that the query, the keys and the values all come from the previous layer of the encoder (respectively, of the decoder). In the decoder, the leftward information is masked to prevent illegal connections. In other words, this masking of tokens that are left of the current token ensures that the decoder only accesses the previously decoded tokens.

Finally, the "encoder-decoder attention" is close to the mechanism described in Section 2.2.3. In this, case, the query is the previous decoder's state, and the keys and values come from the encoder.

The Transformer architecture has been shown to advance the state-of-the-art in English to German and English to French machine translation tasks and has also seamlessly tackled the problem of token embeddings aggregation that has long been an issue for sequence modeling.

2.2.4.3 BERT

The Transformer architecture gave rise to several new methods to improve the performances of various NLP tasks.

Among them, Devlin et al. (2019) introduced BERT, a language representation model that addresses the unidirectional limitation of previous Language Model (LM).BERT is based on the optimization of both a Masked Language Model (MLM) and a Next Sentence Prediction (NSP) tasks.

This Multi-Task Learning (MTL) approach is at the core of the pre-trained model (which is later fine-tuned on several other tasks, such as Named-Entity Recognition or SQuAD) and can be seen as a way to train BERT as an LM that operates at both the word-level (MLM task) and at the sentence-level (NSP task).

The MLM task is conceptually close to Mikolov et al. (2013a)'s Continuous Bag of Word model: the model is trained to predict the masked word given its context (both left and right).

The NSP task is addressed by training a classifier $\mathcal{C}$ to determine whether sentence S_2 does follow S_1 in the corpus. In practice, classifier $\mathcal{C}$ takes representations of both sentences $\mathbf{z}_{S_1}$ and $\mathbf{z}_{S_2}$ as inputs and outputs the probability that they are successive sentences.

Those sentence representations $\mathbf{z}_{S_i}$ are comprised in the `[CLS]` (standing for *classification*) token, which is affixed at the beginning of the sentence. This `[CLS]` token is now widely used as a representation of the whole sentence in different models and applications.

It is its performances and its `[CLS]` mechanism that led BERT to be an unmissable model in modern NLP applications and architectures.

Note: While not directly used in this thesis, the BERT model is a central part of our perspectives, hence the detailed description.

Like continuous word embeddings and sequence modeling before that, the attention mechanism permitted a great refinement in user representation. Besides the better handling of long-range dependancies, the attention mechanism also

allowed for a better understanding of what is important in a sentence for a given query. The next chapter presents a contribution that leverages an attention mechanism to create a personalized reading model, understanding what aspect of a product matters to the user from their reviews. Lei Li et al. (2021) follow this path towards explainability by using personalized Transformers. Sun et al. (2019) also used Transformers to encode a user's behavior from both directions on a task of sequential recommendation.

2.2.5 Conclusion

NLP is a vast, fast-growing research domain that encompasses numerous tasks and applications. The effort to learn rich representations of textual data initially separated the task of learning word representation from sequence modeling. Recent works demonstrate the efficiency of end-to-end approaches, learning word embeddings and language models jointly, often offering a simple way to specialize a model with one or several down-stream tasks (ELMo, BERT). Another growing trend is the use of attention mechanisms in NLP architecture, especially since the Transformers, attention-based models, became state-of-the-art on notoriously hard tasks such as NMT.

2.3 Generative Models

In this work, we call generative models architectures that output sequences of text.

Generative models have mostly been developed on tasks such as Machine Translation and Text Summarization. They can be seen as the purest form of RL: for a model to be able to generate data, it must have understood it completely.

Those models arose from the need to either explicitly generate (potentially missing) data or to estimate the distribution of the input data. Along with data reduction algorithms, they also address the Minimum Description Length problem (i.e., the hypothesis that the best description for a given sample is the shortest.), thus compressing the input data in a smaller, possibly denoised representation.

Generative models play a crucial role in the framework of RL: generating a (possibly unseen) sample from the input data amounts to understanding the way it is organized. This is especially well illustrated in Computer Vision, where models such as Generative Adversarial Networks (GANs) can produce ever-more realistic pictures.

In NLP, generative models were historically developed in NMT and Automatic Text Summarization. They solve an arguably harder task that Computer Vision generative models, as the noise can have a much bigger impact on the understandability of the output (a noisy, blurry image can be potentially understandable while it gets quickly very hard to make sense of a sentence where the wrong words are predicted).

Textual generative models also holds a great explainability advantage. The outputted text depends only on the input data and the inner (or latent) representation of this data in the model, thus giving humans a good idea of why the model outputted such or such sentence. One can view the outputted text as a direct explanation of how the model understands the data.

Furthermore, textual generative models have taken a very important place in NLP in the past few years, especially with regard to Representation Learning. Indeed, most of today's architectures are pre-trained on text generation tasks (such as fill-in-the-blank, text-infilling *etc*.) before being eventually fine-tuned on downstream tasks.

Text generation is of particular importance to this work because of its descriptive power which we leverage with a job generation task. We use text generation both as a mean to ensure the richness of our representation and as a way to better understand and explain the representation we built.

In this section, we will present several key generative architectures, first general and then specifically text-oriented, as well as the imperfect metrics used for evaluating computer-generated sequences.

2.3.1 General Generative Architectures

Generative models are present in every field of Deep Learning, as they can be applied to almost any type of data. In this section, we detail the Variational Auto-Encoder (VAE), one of the most studied generative architecture. We will start by explaining the concept of Encoder-Decoder Architecture before diving into the specifics of Auto-Encoders, Denoising Auto-Encoder (DAE) and, finally, VAE. *Note: although Generative Adversarial Networks have produced very interesting results in Computer Vision, they are not much used in NLP and will therefore not be detailed in this work.*

2.3.1.1 Encoder-Decoder Architecture

The Encoder-Decoder architecture, illustrated in Figure 2.10 is a widely used one in Deep Learning. The idea is quite simple: an encoder E transforms an input

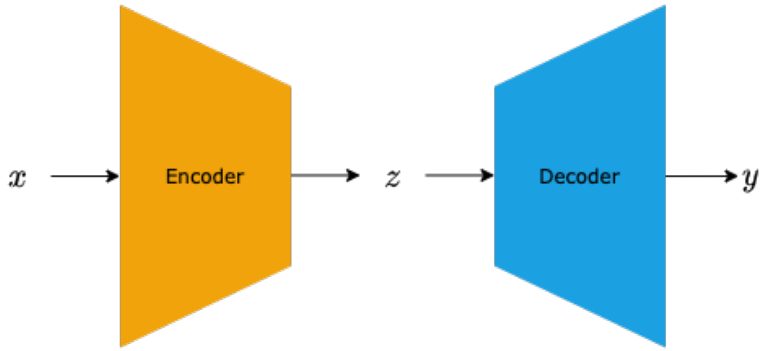

Figure 2.10 – **Illustration of the Encoder-Decoder Architecture**. *Note: in the case of auto-encoders,* $\mathbf{y} = \hat{\mathbf{x}}$.

$\mathbf{x}$ into an intermediary representation $\mathbf{z}$, which is decoded into $\mathbf{y}$ by the decoder D. The latent representation $\mathbf{z}$ often lives in a space (called the latent space) of much lower dimensionality than the input space. This architecture is useful to achieve data compression, but also for representation learning as it can extract features from $\mathbf{x}$ that are relevant to the down-stream task at hand.

2.3.1.2 Auto-Encoders

An Auto-Encoder (AE) is a fairly popular architecture used in numerous research domains. It is a special case of the Encoder-Decoder architecture, where the Decoder outputs a reconstructed version $\hat{\mathbf{x}}$ of the input x.

AEs are trained to learn a compressed latent representation of an input x via an Encoder (E) and then reconstruct it with a Decoder (D), with a reconstruction loss function (usually, the MSE).

$$\mathcal{L}(\mathbf{x}) = \|\mathbf{x} - D(E(\mathbf{x}))\|_2^2 \tag{2.27}$$

AEs can be used for learning compressed representations of an input directly (since the output space of Encoder E is often of lower dimensionality than the input space) or to extract robust features that best represent the data.

2.3.1.3 DAE

A Denoising Auto-Encoder (DAE) offers a more robust implementation than AEs, as it is trained to reconstruct a target $\mathbf{x}$ from a corrupted input, where $\mathbf{x}$ is added a random noise ϵ.

$$\mathcal{L}(\mathbf{x}) = \|\mathbf{x} - D(E(\mathbf{x} + \epsilon))\|_2^2 \tag{2.28}$$

Both AEs and DAEs are largely used in NLP tasks, but while they do output words sequences, they do not *generate* sequences in the sense that they can only output sequences that exist in the training set.

2.3.1.4 VAE

A VAE on the contrary is specifically designed for generation. Proposed by Kingma and Welling (2014), they are conceptually close to DAEs and rely on the same building blocks as AEs, but they *learn* the noise of the input, to determine the probabilistic distribution of the data. VAEs are a special case of regularized AEs with a structured latent space. In a VAE, both the encoder E and the decoder D are described as distribution. The encoder E is noted $q_\phi(\mathbf{z} \mid \mathbf{x})$, the distribution of the latent representation $\mathbf{z}$ knowing input $\mathbf{x}$. The decoder D is noted $p_\theta(\mathbf{x} \mid \mathbf{z})$ because it aims at reconstructing the initial input $\mathbf{x}$ from $\mathbf{z}$. E and D are trained jointly to optimize the Evidence Lower BOund (ELBO) objective:

$$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x} \mid \mathbf{z})\right] - \mathrm{KL}\left(q_\phi(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z})\right) \tag{2.29}$$

with KL the Kullback–Leibler divergence. The VAE aims at reproducing the real distribution $p(\mathbf{x})$ of the data in $p_\theta(\mathbf{x})$, the encoder's distribution. That is ensured by the first term of Equation 2.29 (also referred to as the reconstruction term). But it also aims at constraining its latent space to be structured. The second term of of Equation 2.29 (the regularizing term) ensures that the representation $\mathbf{z}$ of $\mathbf{x}$ is encoded according to the distribution $p(\mathbf{z})$.

Note that, since the decoder is fed a latent representation $\mathbf{z}$ sampled from $p(\mathbf{z})$, back-propagation of the gradient is compromised (the sampling procedure "cuts off" the gradient graph). This issue has been addressed with what is commonly called the "Re-parametrization Trick", and consists in using a third independent variable $\epsilon \sim p(\epsilon)$. Since $\mathbf{z}$ is sampled from a Gaussian distribution of parameters (μ_ϕ, σ_ϕ), we can re-parametrize $\mathbf{z}$ as:

$$\mathbf{z} = \mu_\phi + \sigma_\phi \cdot \epsilon, \quad \text{with} \quad p(\epsilon) = \mathcal{N}(0, I). \tag{2.30}$$

Thanks to Equation 2.30, $\mathbf{z}$ still contains randomness, and the gradient graph is not discontinued.

2.3.2 Text Generation

Text Generation is a special case of generation because it requires encoding and decoding *sequences*. Thus, the Encoder and Decoder of the architectures have

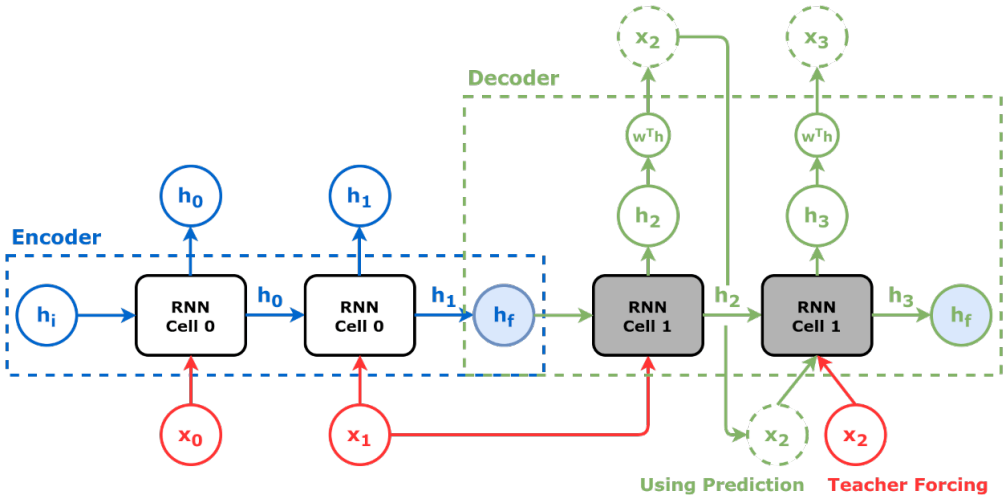

Figure 2.11 – **Illustration of the Sequence-to-Sequence Architecture**. Image by *dvgodoy*[8].

long been Recurrent models, until the recent advent of the Transformers. Most evolutions of Text Generation have been studied and developed in the context of Neural Machine Translation (or NMT).

NMT is the task to automatically translate a text from a source natural language to a target natural language. One of its core problematics is thus text generation, as the output of a model is supposed to be the translation of the input in another language.

2.3.2.1 Seq2seq Architecture

Sutskever et al. (2014) first introduced the Encoder-Decoder Architecture illustrated in Figure 2.10 in the context of NMT.

This architecture was motivated by the need to handle variable-length inputs, crucial in NMT, Speech Recognition and Question Answering to name a few domains. Since all those applications depend on sequences, they named their approach a "Sequence-to-Sequence" (Seq2Seq) model. The idea of the Seq2Seq architecture is to encode a variable-length input sequence into a fixed-length representation vector ($\mathbf{h}_f$ in Figure 2.10) that is then fed to a decoder, itself trained output tokens of the target sequence.

Note that Figure 2.10 presents 2 possible conditioning for the decoder. At training time, it is not unusual to feed the decoder with the expected previous token rather than with the one that has actually been outputted. This method is called "Teacher Forcing" and is useful to prevent the decoder from drifting too far away

from the target, as well as accelerating convergence. At inference time however, the decoder is fed its own previous output at each time step.

When in the special case of Autoencoders, the Encoder and the Decoder are trained jointly to optimize the reconstruction loss of the input. Autoencoders are a common method to learn compact and rich representations of an input. The decoder is trained to ensure that the representation contains enough information for the original input to be reconstructed, thus influencing the encoder's gradient.

In Sutskever et al. (2014) both the Encoder and the Decoder are composed of LSTM cells. This implementation (with variations such a pooling layers, number of directions etc) has long been the most wide-spread until the Transformers came along.

2.3.2.2 Back Translation

On top of the variable-length input problem, NMT faces the problem of parallel data. Since NMT aims at translating a text, written in a source language, to a target language, supervised training requires large amounts of parallel data. That is, corpora of text available in the source language *and* the same text in the target language. Such corpora are not only expensive (translations are human-generated), they are also hard to evaluate with the traditional metrics such as the BiLingual Evaluation Understudy (BLEU) score (we detail the evaluation of generated text in Section 2.3.3). These limitations motivated Lample et al. (2017) to explore a mechanism that could allow for the unsupervised training of translation models: the Back Translation.

Note that Lample et al. (2017) did not invent the mechanism of Back-Translation, but rather improved it by pairing it with a DAE (see Section 2.3.1.3), resulting in better performances in classical NMT tasks. Back Translation was initially introduced by Sennrich et al. (2015) as a mean to improve the decoder of a target language model.

Back Translation is a mechanism that enables the training of NMT models in an unsupervised fashion. Initially, Back Translation (BT) consisted in iteratively training the same denoising auto-encoder on two languages in parallel, with a common latent space.

Figure 2.12 illustrates the training procedure for BT. The encoder and the decoder operate on both the source and the target languages, and switch lookup tables according to a language identifier that is given as input. Firstly, the model

8. https://github.com/dvgodoy/dl-visuals, image licensed under CC 4.0: https://creativecommons.org/licenses/by/4.0/

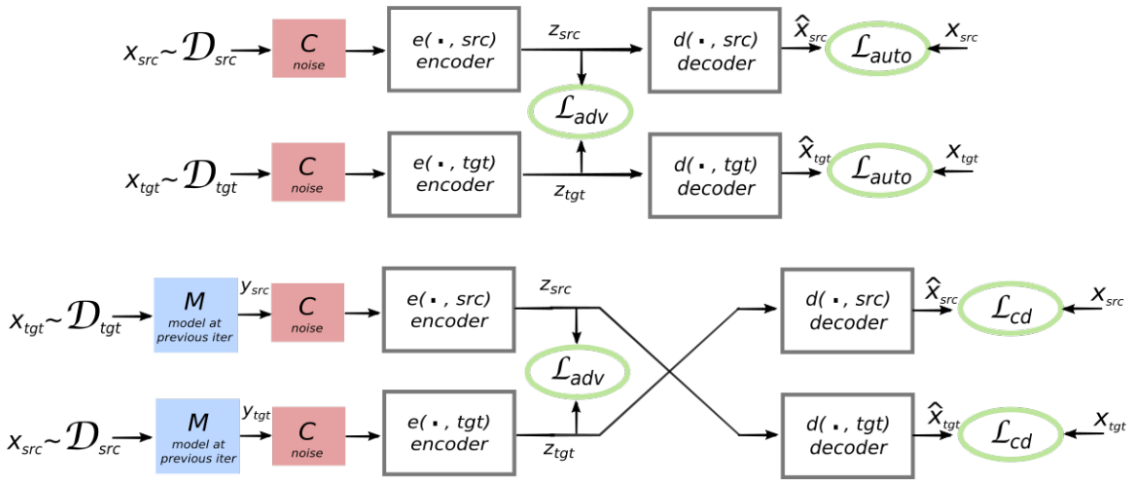

Figure 2.12 – **Illustration of the Back Translation Process**. *Image taken from the original article Lample et al. (2017).*

learns to denoise inputs (sentences) in each domain (the source and the target language s) via an auto-encoding loss ($\mathcal{L}_{\text{auto}}$). Note that both domains latent representations $\mathbf{z}_{\text{src}}$ and $\mathbf{z}_{\text{tgt}}$ are bound to the same latent space by an adversarial loss $\mathcal{L}_{\text{adv}}$. It is the auto-encoding part.

Secondly, we proceed as before except that we encode a sentence from the other language and use the translation outputted by the previous-iteration model M_{t-1} as input. Then, we compute the cross-domain loss $\mathcal{L}_{\text{cd}}$ which measures how well the model can reconstruct $\mathbf{x}_{\text{src}}$ (respectively, $\mathbf{x}_{\text{tgt}}$) given $\hat{\mathbf{x}}_{\text{tgt}}$ (respectively $\hat{\mathbf{x}}_{\text{src}}$) as input.

2.3.2.3 Transformer-based Text Generation

GPT. One of the undoubtedly most groundbreaking generative models of the decade is the group de Generative Pre-Training (GPT) models. In 2018, Radford and Narasimhan (2018) introduce a model that overcomes two majors limits of most NLP models: the amount of annotated data they require to be trained, and their inability to generalize to tasks different from the one they were trained for. They propose a Transformer-based model that will be first pre-trained with a generative objective and then fine-tuned on specific tasks.

Formally, the first part of the training consists in optimizing a traditional language modeling objective:

$$L_1(V) = \sum_t \log p\left(w_t \mid w_{t-i}, \ldots, w_{t-1}; \Theta\right),$$

with V the unsupervised corpus of tokens, i the size of the context window, and p, the conditional probability for token w_t to be observed given its context $w_{t-i}, \ldots, w_{t-1}$. Function p is a Transformer decoder with 12 layers and 12 attention heads.

For the fine-tuning part, the input sequence $\mathbf{x}$ of length N is ran through the pre-trained network to be represented as activated output of the Transformer. Then it is projected into a space of dimension $\mathcal{C}$, $\mathcal{C}$ being the number of possible labels for the fine-tuning task. This yields:

$$L_2(\mathcal{C}) = \sum_{(\mathbf{x},\mathbf{y})} \log P\left(\mathbf{y} \mid \mathbf{x}^1, \ldots, \mathbf{x}^N\right).$$

Additionally, Radford and Narasimhan (2018) found that combining both objectives L_1 and L_2 when fine-tuning the model helped to accelerate convergence and improve the generalization abilities of the supervised model.

GPT was pre-trained on the BooksCorpus dataset of over 7000 unpublished books and fine-tuned on 4 different tasks, over a total of 12 datasets.

BART. Bidirectional and Auto-Regressive Transformers (BART) is a model that aims at bridging the gap between the versatile representations learned by models like BERT, described in Section 2.2.4.2 and performant generative models like GPT. Lewis et al. (2020) address this objective by combining (and generalizing) the masking mechanism and bidirectional encoder of BERT with the auto-regressive decoder of GPT.

The bidirectional encoder, trained on a masked language model task, allows the model to learn about short-ranged bidirectional token dependancies. BART extends this process by masking not only tokens but entire bits of the sentence, forcing the model to reason on the sentence length to perform completion. They refer to this noise function as "Text Infilling". On top of this extended range of corruption, BART is also pre-trained to reconstruct inputs that have undergone transformations such as document rotation, sentence permutation, token deletion or even a combination of all of the above.

After the pre-training of the denoising autoencoder, BART's decoder is fine-tuned on a specific downstream task. The left-to-right autoregressive decoder of BART forces the model to output the next token using only the past (or leftwards) information, like most generative textual model. Lewis et al. (2020) reported improvements of performances in tasks such as token classification, sentence generation, machine translation and sequence classification.

The success of the BART architecture led Y. Liu et al. (2020) to propose a multi-lingual, extended version of BART called mBART where the model is pre-trained on multi-lingual denoising. This initialization showed to have improved the performances of several NMT tasks.

T5. The Text-to-Text Transfer Transformer (T5, for short) is another successful Transformer-derived architecture introduced by Raffel et al. (2020). Building on the power of Transfer Learning and Multi-Task Learning, the T5 framework is pre-trained on a very large amount of data (the Colossal Clean Crawled Corpus) and allows for the combination of several tasks (NMT, sentence similarity...) at training time (either pre-training or fine-tuning). The originality of the approach that the the model only ever outputs text. It achieves this with prefixing the input with the task indicator and is trained on all the tasks with the same model, loss function, and hyper-parameters.

Note: the models detailed in the above section are too recent to have been used in this thesis. However they have had a sufficient impact to be explained here.

2.3.3 Evaluation

Evaluation of textual generation comes with a lot of challenges for different reasons. Firstly, labels are more often than not human-generated. For Machine Translation, supervised learning requires large corpora of parallel data (i.e., with sentences in the source language and in the target language); in Text Summarization, labeling requires a tremendous amount of time and human-power. That makes labeled data very expensive and rather scarce. Aside from this, when labels are available in either NMT or Text Summarization, they are just one possible label rather than an absolute truth, as it is possible to translate / summarize the same document in different correct manners. It is partly what motivated Lample et al. (2017) to develop a framework that relied on monolingual corpus. On the other hand, the lack of semantic methods makes the evaluation of generative models rely mainly on statistical methods, such as BLEU or ROUGE.

BLEU score. The BLEU acronym stands for Bilingual Evaluation Understudy. Introduced by Papineni et al. (2002b), it aims at evaluating the resemblance between a candidate word sequence (typically, a proposed translation) and a human-generated reference: the label. The BLEU score is said to be a precision-oriented metric, as it evaluates how much the n-grams in the candidate sentence appeared in the human reference, with respect to the candidate's length. The BLEU score can be divided into BLEU-1 through BLEU-4, each metric respectively accounting

Reference: Bud Powell was a legendary pianist

	BLEU	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Candidate 1: Bud Powell was a legendary pianist	100	100	100	100	100
Candidate 2: Bud Powell was a historic piano player	41.11	57.1	50.0	40.0	25.0
Candidate 3: Bud Powell was a New Yorker	50.81	66.7	60.0	50.0	33.3

Figure 2.13 – **Illustration of the BLEU score.**

for the number of unigrams through four-grams that the candidate has in common with the reference.
An example is presented in Figure 2.13. When comparing the first candidate sentence to the reference, we get a perfect BLEU score because both sentences are identical. However, when we compare the scores of candidates 2 and 3, a major flaw of the BLEU metrics stands out: although candidate 2 is semantically closer to the reference than candidate 3, it obtains a lower BLEU score due to having one more word than the reference.

On Figure 2.13 we can see that the second candidate yields a lower BLEU score than the third one, despite being semantically closed, due to candidate 2 being longer than the reference. Callison-Burch et al. (2006) also illustrate the limitations of the BLEU score by arguing that there are millions of variations of a predicted translation that would get the same BLEU score despite not being syntactically, grammatically or even semantically equivalent.

Papineni et al. (2002b) define their modified precision score, p_n, for each n-gram length by

$$p_n = \frac{\sum_{S \in R} \sum_{\text{ngram} \in S} \text{Count}_{matched}(\text{ngram})}{\sum_{S \in R} \sum_{ngram \in S} \text{Count}(\text{ngram})}. \tag{2.31}$$

In Equation 2.31, S represents a sentence, R the corpus of references, and ngram a word in sentence S. The term $\text{Count}_{matched}(\text{ngram})$ corresponds to the count of tokens in common between the reference and the candidate.

The formulation of the BLEU score includes a term of "Brevity Penalty" to penalize overly short candidate sentences. In Equation 2.32, c refers to the length or the candidate sentence, while r refers to the length of the reference sentence?

$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases} \tag{2.32}$$

The complete equation of the BLEU score (Equation 2.33) is obtained by combining Equation 2.31 with Equation 2.32 and assigning a weight w_n to each value of p_n (in practice, all w_n are equal).

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{2.33}$$

ROUGE Score. ROUGE (Lin 2004) stands for Recall-Oriented Understudy for Gisting Evaluation. Contrary to the BLEU score, the ROUGE metric is rather recall-oriented as it evaluates how many n-grams of the reference are represented in the candidate sentence, with respect to the reference's length. It is usually declined in 3 metrics: ROUGE-1, ROUGE-2 and ROUGE-L. The first two metrics are equivalent to the BLEU-1 and BLEU-2 recall-oriented, while ROUGE-L is a Longest Common Subsequence based statistic. The ROUGE metric can also be declined in ROUGE-precision and ROUGE-F1 because it was conceived as an extension and improvement of the BLEU score. However in this work we will only report the recall value of the ROUGE for illustration purposes.

Reference: Bud Powell was a legendary pianist

	ROUGE 1_{recall}	ROUGE 2_{recall}	ROUGE L_{recall}
Candidate 1: Bud Powell was a legendary pianist	100	100	100
Candidate 2: Bud Powell was a historic piano player	66.6	60.0	66.6
Candidate 3: Bud Powell was a New Yorker	66.6	60.0	66.6
Candidate 4: Bud Powell is a New Yorker	50.0	20.0	50.0

Figure 2.14 – **Illustration of the ROUGE score.**. test

Figure 2.14 re-uses the example of Figure 2.13. Since the Recall-Oriented Understudy for Gisting Evaluation (ROUGE) score is recall-oriented, both candidates 2 and 3 have the same score because they have the same number of n-gram in common with the reference. However, candidate 4 has one less common n-gram with the reference, dropping both ROUGE-N scores, and it is in the middle of the sentence, which affects the ROUGE-L score.

$$\text{ROUGE-N}_{recall} = \frac{\sum_{S \in C} \sum_{\text{ngram} \in S} \text{Count}_{matched}(\text{ngram})}{\sum_{S \in C} \sum_{ngram \in S} \text{Count}(\text{ngram})}. \tag{2.34}$$

In Equation 2.34, S represents a sentence, C the corpus of candidate sentences, and ngram a word in sentence S. The term $\text{Count}_{matched}(\text{ngram})$ corresponds to the count of tokens in common between the reference and the candidate.

Perplexity. The Perplexity metric is often used to assess the *fluency* of a generated sequence. It evaluates the likelihood of an outputted sequence given a reference LM's probability distribution.

2.3.4 Conclusion.

Generative models are an ever-growing part of Deep Learning (DL) and specifically NLP research. The Encoder-Decoder architectures allow for both a compression of the input data and the learning of a meaningful, versatile latent representation. Recent advances such as the Transformers allowed for impressive results, both qualitatively and quantitatively in NMT and Automatic Summary. However the current methods to evaluate those models are insufficient, and the scientific community is actively working on proposing new ones.

2.4 Recommendation, NLP and Generative Models for User Representation

The work presented in this thesis is at the crossroads of three major domains: Recommendation, Natural Language Processing (NLP) and Generative Models. We tackle User Representation (UR) through the prism of Representation Learning (RL) and in this chapter, we detailed the work related our approaches.

The work presented in this thesis stems from Recommendation, as it revolves around UR (a crucial part of the domain). Our work draws from both Collaborative Filtering (CF) and Content-based (CB).

We are interested in the interactions between the users and the items (in our case, the jobs [9]) and adopt a RL approach, philosophically close to Matrix Factorization (MF)-based models that learn latent, abstract representations of both the users and the items of their systems. However, we put the focus of our work on textual descriptions, which is traditionally a CB approach.

The originality of our work partly lies in the fact that the user interactions with the system are textual, thus requiring an NLP-based hybrid approach.

9. This is detailed in Chapter 4

The textual nature of our data naturally led us to consider NLP models and methods. Indeed, Chapter 3, Chapter 4 and Chapter 5 all rely on the textual interactions with the system, which are text reviews for Chapter 3 and professional experiences for the other chapters.

But the choice to focus our approaches on NLP are also motivated by the hypothesis that user-generated texts are a great way to faithfully and uniquely represent a user. Since there is an infinity of possible ways to describe the same opinion or the same professional experience, we consider the words and syntax chosen by a user to be unique, and to some extent, defining of the user.

Additionally, we believe that work on NLP around user profiles could be beneficial beyond our application, as this kind of data is widely available.

Finally, we are very interested in proposing explainable predictions, in line with the work of Explainable Artificial Intelligence (xAI). To this end, we use NLP methods to provide a level of explainability in our work, namely by leveraging textual generative models.

We leverage Generative Models for several reasons and purposes. We initially considered a generative model in order to address our next job prediction task, as we quickly set aside the classification approach. The lack of standard and the noise of the data made it unreasonable to adopt a classification framework, so instead we chose to generate a person's next job (title and description).

Besides, as we mentioned above, models that generate text are a great approach to explainability, as they literally speak to you. The text they output is a direct explanation for their prediction.

From a broader point of view, Generative Models have played a crucial role in the last years' rise of RL, and with good reason: the only way for a model to generate new data from a given input if to have understood said input completely. This is especially true in NLP, where they are at the heart of most recent architectures, which are typically pre-trained on a self-supervised generation task. The representations given by those pre-trained models are versatile and reusable, and can be fine-tuned on other down-stream tasks.

CHAPTER 3

REFINING USER UNDERSTANDING IN RECOMMENDATION VIA NLP

Chapter abstract

In this chapter, we aim at learning relevant profiles from both ratings and textual reviews in the context of Recommendation. We attempt to make better use of review texts and seek to build a Recommender System (RS) that does not fully rely on Matrix Factorization (MF).

In order to achieve this goal, we combine a classic recommendation model to a hierarchical neural network, which was first dedicated to sentiment analysis. The addition of a sentiment analysis model allows us to better understand a user's tastes, and to refine the recommended items given those tastes.

We propose to add some personalized attention parameters to the hierarchical neural network, and demonstrate that this modified attention is useful to build effective recommendation profiles: not only does it improve the global recommender system performance, but it also enables us to provide suggestion explanations by exploiting the underlying learned textual latent space.

This latter point is a noteworthy way to overcome the classical black-box phenomenon in Collaborative Filtering (CF) approaches. The work in this chapter has led to the publication of a conference paper:

- Charles-Emmanuel Dias, Clara Gainon de Forsan de Gabriac, Vincent Guigue, and Patrick Gallinari (2018). "RNN & modèle d'attention pour l'apprentissage de profils textuels personnalisés". In: *CORIA 2018 - 15ème COnférence en Recherche d'Informations et Applications*. Rennes, France. URL: https://hal.archives-ouvertes.fr/hal-02503473.

Contents

In order to help users access an ever-increasing number of online resources, RSs have been developed to select and sort out content likely to appeal to them. They can, for example, suggest where to dine, compose newsletters, provide individualized advice on which products to buy or, more simply, which movies to watch. These systems, which seek to model user preferences and product attributes, aim to provide truly personalized access to information.

To achieve this goal, the learned profiles must be able to provide accurate recommendations. However accurate, it is equally important to explain why these recommendations are appropriate (Tintarev and Masthoff 2007), as detailed in Section 2.1. Recommendation algorithms have reached a first step of efficiency (and popularity) with the Netflix challenge (Bennett, Lanning, et al. 2007), which established CF by MF as a powerful framework (Koren et al. 2009), as described in Section 2.1.1. In the last few years, these algorithms have become more and more efficient thanks to the consideration of a growing number of factors such as time (Koren 2010; J. J. McAuley and Leskovec 2013; Sánchez and Bellogín 2018b), social links (Guy 2015) or even text (J. McAuley and Leskovec 2013).

Still, these quantitative improvements have done little to improve the explanatory side of the recommendation and recent CF methods are still often described as black boxes. This lack of interpretability in favor of performance is a direct result of the choices made in modeling profiles. Indeed, since the Netflix competition in 2006, user (and product) profiles are derived from a MF process of a set of scores that structures an abstract latent space where the dimensions are difficult to interpret. However, in practice, ratings are often accompanied by written opinions. Section 2.1.4.2 details how these texts have already been successfully

exploited to improve the quality of prediction (J. J. McAuley and Leskovec 2013; Ling et al. 2014). Almahairi et al. (2015) have also proposed to re-project the recommendation into the review space to provide explanations and thus an interpretable recommendation. Textual data also represents an elegant way to deal with cold start, so problematic in CF approaches (Dias et al. 2017b).

In this paper, our objective is to present an architecture based on Deep Neural Networks (DNNs) capable of defining highly relevant user profiles and items from the ratings and review texts. It is worth nothing that this work was conducted before the advent of readily-available Transformers, and can thus be considered as an extension of the work laid by Word2Vec. We build on work in sentiment analysis which, based on a hierarchical attention model that extends the work of Z. Yang et al. (2016), can automatically detect words and phrases of interest. The original assumption of our approach is that we consider the sentiment classifier as a personalized reading model and believe that a user's attention to words and phrases can define their profile in an original and relevant way. Besides, this sentiment classifier makes use of the attention mechanism to propose a way to explain the suggestions outputted by the system, in line with the Explainable Artificial Intelligence (xAI) movement.

Our system is perfectly integrated, so it is possible to see it as a sentiment classification system with personalized weighting parameters on words and sentences or, conversely, as a recommendation system using text and attention as a way to regulate learning, while allowing to interpret the suggestions made. Moreover, we believe that collaborative profiles mainly encode user and item biases while text allows us to better understand the important aspects of a product for the user, focusing on examples where the aforementioned biases are insufficient to reconstruct the observed rating.
We will first detail the model associated with our work (Section 3.1), before demonstrating the quantitative and qualitative interest of our approach (Section 3.2).

3.1 The model

Our model, called Hierarchical Recurrent Attentive Neural Network for Recommendation (HRAN), aims at using text as a medium to model users and products in an interpretable vector space. It is composed of two parallel modules.

The first one is a classical Multi-Layer Perceptron (MLP) with two layers. It takes as input the user and product profiles to estimate a score. As such, it plays the same role as the MF in CF. The first hidden layer of this module extracts low

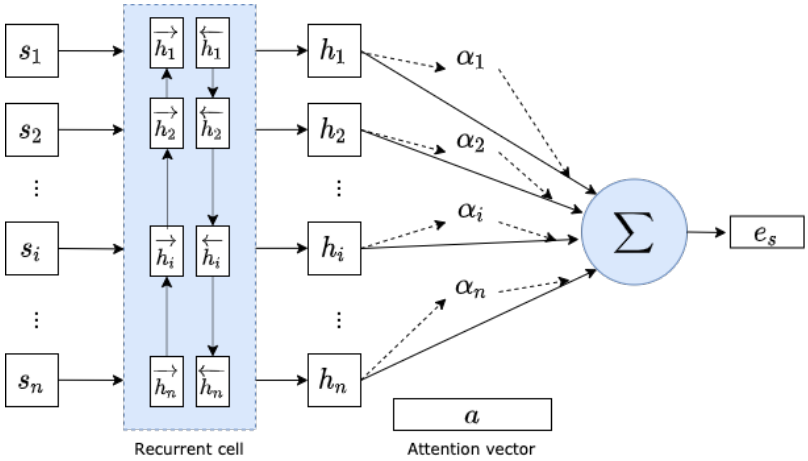

Figure 3.1 – **Detail of a bidirectional-attentive recurrent module (RBA)**. The s_i designate either words or sentences depending on the hierarchical level considered.

(word) level features about the user's preferences, while the second extracts high (sentence) level features.

The second one is a hierarchical Recurrent Neural Network (RNN) composed of two Recurrent Bidirectional Attentive modules (RBA) (as introduced by Z. Yang et al. (2016)) which allow to encode opinions by successively analyzing the words of a sentence, then the sentences of the review. Its role is to analyze the review of a user on a product and to predict its polarity. This last network is dependent on the first one because its two RBA are linked to the hidden layers of the first one: the RBA's total attention vector is a combination of a global attention vector and a learned personalized attention vector that is outputted by the corresponding MLP layer (the first for the word level, the second for the sentence level). This is illustrated on Figure 3.2. This attention combination leverages the fact that the words and sentences that a user choses allow to define his profile for recommendation applications.

First, we detail the MLP-for-recommendation part of our model. Then, we explain the construction of a RBA. Finally, we formally describe the two-headed architecture of the global network.

3.1.1 Multi-Layer Perceptron for Recommendation

The first part of the network is a classical MLP composed of two hidden layers and a regression layer. It takes as input a pair of representations $(\text{user}, \text{item})$ which it concatenates and transforms sequentially into two sub-representations

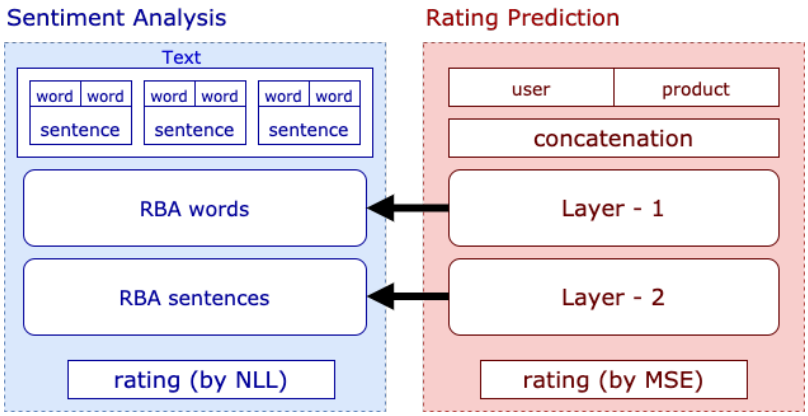

Figure 3.2 – **General view of the model**.

ℓ_w and ℓ_s.

The MF implementation of CF allows to extract continuous latent profiles of users $P = \{\mathbf{p}_u\}_{u=1,\dots,N_u} \in \mathbb{R}^{N_u \times Z}$ and products $Q = \{\mathbf{q}_i\}_{i=1,\dots,N_i} \in \mathbb{R}^{N_i \times Z}$directly from the score matrix, so that the predicted score $\hat{r_{ui}}$ results from the dot product between the product i and the user u profiles:

$$\hat{r_{ui}} \approx \mathbf{q}_i^T \mathbf{p}_u, \qquad \mathbf{q}_i, \mathbf{p}_u \in \mathbb{R}^{Z \times Z} \tag{3.1}$$

In the manner of topic modeling methods, each latent dimension of the profiles is assumed to correspond to a concrete aspect. The presence of the same aspect in two profiles causes a local correspondence that raises the score.

In recommendation, the scores are highly biased. Some people tend to always give good/bad ratings and some products tend to be over/under rated. Classically, these biases are modeled by adding a global mean μ, a product bias b_i and a user bias b_u to Equation 3.1. The prediction then takes the following form:

$$\hat{r_{ui}} = \mu + b_i + b_u + \mathbf{q}_i^T \mathbf{p}_u \tag{3.2}$$

and the profiles are learned in a regularized way to avoid overfitting:

$$\begin{aligned} q^\star, p^\star, b^\star = \arg\min_{q,p,b} \sum_{(u,i)} & (r_{ui} - \mu + b_i + b_u + \mathbf{q}_i^T \mathbf{p}_u)^2 \\ & + \lambda(||P||_F^2 + ||Q||_F^2 + b_u^2 + b_i^2) \end{aligned} \tag{3.3}$$

In this formulation, the latent profiles capture only the deviation from the mean, allowing an estimate to be offered even when a profile is missing.

Formally, let $\ell = [\mathbf{u}; \mathbf{i}]$ be the concatenation of a pair of user and product representations. The predicted score $\hat{r}_{ui}$ is simply obtained by regression, successively transforming ℓ into ℓ_w, ℓ_w into ℓ_s and finally ℓ_s into $\hat{r}_{ui}$.

$$\ell_w = \tanh(W^{\ell w}\ell + b_\ell), \quad \ell_s = \tanh(W^{ws}\ell_w + b_w), \quad \hat{r}'_{ui} = \tanh(W^{sr}\ell_s + b_s) \tag{3.4}$$

Plugging Equation 3.4 into Equation 3.2 yields:

$$\hat{r_{ui}} = \mu + b_i + b_u + \tanh(W^{sr}\ell_s + b_s) \tag{3.5}$$

The objective of this network is to minimize the Mean-Squared Error (MSE) on the score prediction.

The choice of a two-hidden-layer perceptron allows us to extract low level ℓ_w and high level ℓ_s features. We will use ℓ_w to encode attention on the words of the pair (user $\mathbf{u}$, product $\mathbf{i}$) while ℓ_s will encode attention on the sentences. The transition from one network to the other is done using the projection described in Equation 3.8.

3.1.2 Recurrent Bidirectional Attentive module

This module is the main block of the sentiment prediction model. It takes as input a sequence and an attention vector, indirectly weighting the elements of the sequence, to return a representation of the sequence (Figure 3.1). The attention mechanism serves many purposes: it acts as an aggregator of the RNN's hidden representations $\mathbf{h}_i$, as a regularizer and as an explanation module at inference time. The originality of our approach, at this level, lies in the personalization of the attention calculation.

In order to model textual reviews, we rely on Recurrent Neural Network (RNN)s, that have the advantage of naturally handling sequences of variable length. As detailed in Section 2.2.2, the hidden state of a RNN cell at time t is a function of the current input x_t and of the previous hidden state h_{t-1}. This formulation allows for the modeling of sequential information such as text, and in this case, sentences of the review.

Formally, let us consider a sequence $S = \{\mathbf{s}_1, \ldots, \mathbf{s}_i, \ldots, \mathbf{s}_n\}$ composed of n elements. To obtain its representation $\mathbf{e}_s$, the sequence is first passed through a bi-directional RNN $RF = \{\overrightarrow{RF}, \overleftarrow{RF}\}$ which, by traversing the sequence in both directions, encodes the intra-sequence content.

The outputs of the RNN are concatenated at each time step to obtain the set of hidden representations $\mathbf{h}_i$ (Equation 3.6). Here, we use a Long Short-Term Memory (LSTM) as the recurrent cell.

$$\mathbf{h}_i = [\overrightarrow{\mathbf{h}_i}; \overleftarrow{\mathbf{h}_i}], \qquad \overrightarrow{\mathbf{h}_i} = \overrightarrow{RF}(\mathbf{s}_i), \qquad \overleftarrow{\mathbf{h}_i} = \overleftarrow{RF}(\mathbf{s}_i), \tag{3.6}$$

We also make use of the concept of attention to improve the representation of the review. Indeed, the attention mechanism enables the model to learn which word of the review is most relevant to the task at hand (here, rating prediction). The attention parameters weight the information to be considered in the sequence and thus focus on the discriminating elements.

Thus, each $\mathbf{h}_i$ element is non-linearly projected into the attention space to compute the α_i affinity with an attention vector $\mathbf{a}$ according to the following formula:

$$\mathbf{e}_s = \sum_{i=1}^{n} \alpha_i \mathbf{h}_i, \qquad \mathbf{t}_i = \tanh(W^u \mathbf{h}_i + \mathbf{b}_u), \qquad \alpha_i = \frac{\exp(\mathbf{a}^\intercal \mathbf{t}_i)}{\sum_i \exp(\mathbf{a}^\intercal \mathbf{t}_i)} \tag{3.7}$$

These affinities α are normalized using a function *softmax* so that they sum to 1. The attention vector $\mathbf{a}$, which can be seen as an average representation of what is important, automatically learns the discriminating items with respect to the task.

In order to personalize the attention, we define the vector $\mathbf{a}$ as a combination of a global attention vector $\mathbf{a}_g$ and a personalized attention vector $\mathbf{a}_{u,i}$ from the neural network dedicated to note prediction. Formally, the final $\mathbf{a}$ vector is defined as:

$$\mathbf{a} = \tanh(\mathbf{a}_{u,i} + \mathbf{a}_g), \qquad \mathbf{a}_{u,i} = W^\ell \ell \tag{3.8}$$

where $\mathbf{a}_{u,i}$ comes from the recommendation network and corresponds to the merged profile of the item and the targeted user ℓ (see next section). The matrix W^ℓ allows to switch from one network to the other in an efficient way.

The global vector $\mathbf{a}_g$ aims at encoding general adjectives related to polarity prediction (i.e. *bad*, *great*, *awesome*) whereas $\mathbf{a}_{u,i}$ allows the network to focus on more precise words related to the affinities of a person with respect to a product (i.e. *brand name*, *specificities*, *attributes*). Examples of words selected by attention are shown in Figure 3.5.

3.1.3 General Architecture

As shown in Figure 3.1 and Figure 3.3, our architecture (HRAN) is composed of two distinct-yet-linked neural networks. A MLP for recommendation (taking

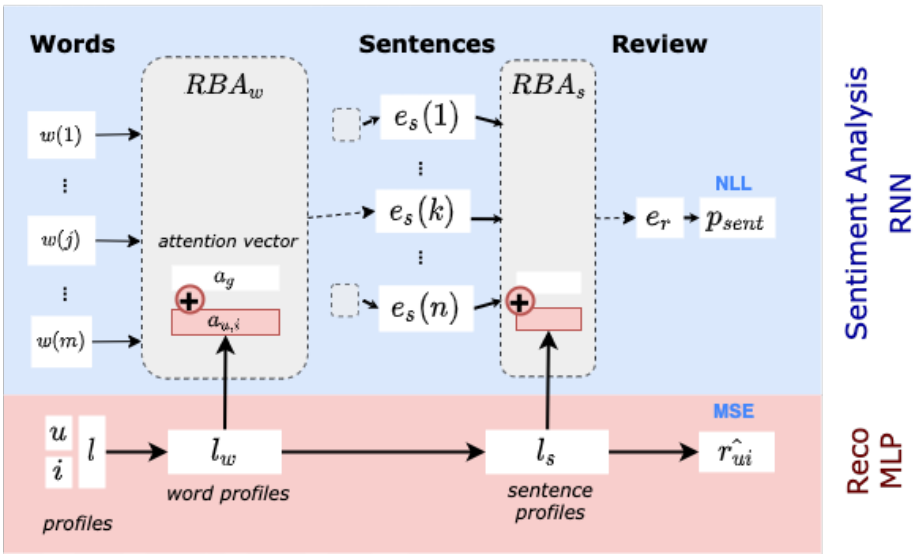

Figure 3.3 – **Detailed view of the model for an input of n sentences of m words.** (Top) The sentiment analysis RNN , consisting of two RBAs, one for words (RBA_w) and one for sentences (RBA_s). (Bottom) Multi-layer perceptron for recommendation.

profiles as input and predicting a score) and a hierarchical RNN for sentiment analysis (taking text and estimating its polarity). HRAN is thus trained in an Multi-Task Learning (MTL) fashion, since both its neural networks are optimizing distinct losses. This MTL framework is a common strategy in Representation Learning (RL) as it allows for the representation of complex, heterogeneous entities (in this case: the ratings and the reviews). We first describe the MLP before returning to the hierarchy of the sentiment analysis module.

3.1.3.1 Hierarchical Sentiment Analysis Network

The second network takes only the opinions as input and tries to predict their polarity. It is composed of two RBAs, used one after the other in order to hierarchically encode the opinions (first, at the level of words, and then, of sentences) and of a classification layer. After the double encoding, the review is represented by a vector $\mathbf{e}_r$:

$$RBA_w : (\{w\}, \ell_w) \mapsto \mathbf{e}_s \qquad RBA_s : (\{\mathbf{e}_s\}, \ell_s) \mapsto \mathbf{e}_r \tag{3.9}$$

The words w of the sentences are encoded in $\mathbf{e}_s$ then the sentences are aggregated in $\mathbf{e}_r$.

Finally, this e_r representation passes through a classification layer of weights W^{pred}. We use an *softmax* to obtain a distribution in the space of possible scores:

$$p_{\text{sent}} = \text{softmax}((W^{\text{pred}}\mathbf{e}_r) + \mathbf{b}_p) \tag{3.10}$$

The goal of this network is to minimize the Negative Log-Likelihood (NLL) on the score prediction, which is equivalent to trying to predict the polarity of the review.

3.1.3.2 Hyper-parameters, Objective and Training

Our models are implemented using *Pytorch* [1]. We jointly minimize two objective functions: the MSE (for the recommendation MLP) and the NLL (for the sentiment classification network). Note that optimizing this joint objective allows us to alleviate the shortcomings of MSE mentioned in Section 2.1.5, while still allowing us to compare ourselves to other models. The entire network parameters are trained by gradient descent, using the Adam optimization algorithm (Kingma and Ba 2015). The final minimized cost is the sum of the MSE and the NLL. The hyper-parameters are the same for all the experiments that follow. The hidden layers are the same size as the word and phrase embeddings: 200 for the RBA and words and 40 for the MLP (20 for users, 20 for products).

3.2 Experiments and Results

First, we present the data used to evaluate our model. The first set of evaluations proposed is quantitative, it consists in evaluating the performances of our models in sentiment analysis (classification of scores) and in recommendation (prediction gap). Then, in order to qualitatively evaluate the model, we propose to observe the learned representations.

3.2.1 Data and Pre-processing

For our experiments, we rely on consumer reviews extracted from Amazon (J. McAuley et al. 2015a). We have selected five review bases from different thematic areas whose statistics are detailed in the Table 3.1. The cold start problem is not addressed in this chapter and the original review bases are subsampled: only the reviews of users and products with at least five reviews are kept.

To separate reviews into lists of sentences and sentences into lists of words, we use the library *spacy* [2]. The words are encoded into representations learned during

1. http://pytorch.org/
2. https://spacy.io/

Dataset	#Reviews	#Users	#Items	p(1)	p(2)	p(3)	p(4)	p(5)
Instant Video	37126	5130	1685	4,6	5,1	11,3	22,7	56,3
Digital Music	64706	5541	3568	4,3	4,7	10,5	25,6	55,0
Video Games	231780	24303	10672	6,4	5,9	12,2	23,6	51,9
Clothes S.J.	278677	39387	23033	4,0	5,5	10,9	20,9	58,6
Movies	1697533	123960	50052	6,1	6,0	11,9	22,6	53,4

Table 3.1 – **Statistics of the different scores for the Amazon databases used**. The scores are in percentage. Each column $p(r)$ reports the percentage of items in the dataset with the rating r.

training. Only the 10,000 most used words are kept, the others are replaced by a representation dedicated to unknown words. Finally, the data are separated into five equal sets for the cross-validation procedure. For each evaluation campaign, we use four sets (80 % of the data) for training, and one set (divided in two) for validation (10 %) and evaluation (10 %).

3.2.2 Sentiment Analysis Evaluation

Our first task is sentiment analysis, which involves predicting the polarity of a text. We propose to compare the performance of our model against three benchmark models that also use text representations:

- **FastText** (Bojanowski et al. 2016). Coming from the work around *word2vec*, the idea is to learn representations of words specific to the classification task. These representations are averaged (over the review) and classified by logistic regression.

- **HAN** *Hierarchical Attention Networks for Document Classification* (Z. Yang et al. 2016): This model is equivalent to our hierarchical RNN, without the personalized attention.

- **NSUPA** *Neural Sentiment Classification with User & Product Attention* (Chen et al. 2016): This model is an evolution of **HAN** aiming at taking into account the user/item biases, as in our approach. To do so, the authors propose to project the text into an attention space parameterized by user and item representations. In their formulation, this amounts to replacing t_i in the Equation 3.7 by $\mathbf{t}_i = \tanh(W^{tu}[\mathbf{h}_i; \mathbf{i}; \mathbf{u}] + \mathbf{b}_u)$. Where Chen et al. (2016) propose to inject the profile parameters directly into the sentiment classification model, we propose a more flexible approach, relying on two independent models connected by the W^{ℓ} parameters (Equation 3.8).

- **SVM** *Support Vector Machines*. SVMs have given very interesting results in sentiment classification in a binary context (merging negative reviews on one hand, positive reviews on the other hand and eliminating ambiguous reviews –3 stars–) (Pang, L. Lee, et al. 2008). Previous papers (Z. Yang et al. 2016; Chen et al. 2016) report poor performance under fine classification on the five stars. We indeed did not obtain satisfactory results in this setting, regardless of the preprocessing considered. For this reason, the SVM results are not present in the following result table.

We evaluate the sentiment analysis capabilities of the models in terms of correct classification rate (accuracy). The results are reported in Table 3.2. Our first benchmark, FastText, is largely outperformed by hierarchical models that take sentence structure into account. The richness of the word representation space and the relevance of this space for opinion analysis are not enough to compensate for the loss of document structure and the weakness of the word aggregation function (a simple average).

The comparison of the three hierarchical models shows the importance of modeling user/product biases to pass a performance threshold: these biases are present in NSUPA and in our approach (HRAN) but not in HAN. Between our proposal and NSUPA, the performances are very close. The challenge of our formulation was to reach the state of the art performances in sentiment classification and to better transfer the relevant information to the recommendation profiles.

	FastText	HAN	NSUPA	HRAN
Instant Video	62.60	64.50	65.88	**66.60**
Digital Music	63.58	68.03	**70.08**	68.80
Video Games	62.51	67.67	68.60	**69.11**
CSJ	67.83	71.96	**71.99**	71.49
Movies	64.56	68.95	71.20	**71.62**

Table 3.2 – **Accuracy metric on the sentiment analysis task**. The reported scores are averaged over 5 splits (in %).

3.2.3 Recommendation Evaluation

The second objective of our approach is score prediction, a classical task in CF. We opted for a standard reference in the domain: MF, which infers the scores only from the user profiles and items learned from the scores, without taking the reviews into account.

Dataset (#reviews)	Mean (μ)	w/offset	MF	TransNet	HRAN
Instant Video (37.126)	1.25	1.137	1.024	1.526	**0.937**
Digital Music (64,706)	1.19	0.965	0.903	1.522	**0.838**
Video Games (231,780)	1.45	1.281	1.267	1.313	**1.076**
CSJ (278,677)	1.215	1.123	1.365	1.285	**1.081**
Movie (1,697,533)	1.436	1.148	1.118	1.359	**1.058**

Table 3.3 – **RMSE in rating prediction**. The reference values are the global mean μ, the global bias (Equation 3.2), a MF and the Transnet model. The values presented are the average RMSE over the five ensembles.

On the contrary, the second reference, TransNet (Catherine and Cohen 2017), only takes the text into account: TransNet adopts an approach based on the *matching* between the texts of the user's past and the texts associated with the target item, in the way that was proposed by Dias et al. (2016). The $\hat{r}_{ui}$ score is directly predicted from the existing textual profiles, without going through a factorization of the observations.

As usual in recommendation, we put these performances in perspective against trivial models predicting respectively the overall mean of the dataset (μ) and the mean corrected for user and item bias (*w/offset*). The results are presented in Table 3.3. We use Root Mean Square Error (RMSE) as the evaluation measure [3].

In terms of rating prediction, our model is systematically better than the reference models. The TransNet model is disappointing since it is almost systematically below average -except on the most supplied dataset-: being based only on text, it requires a lot of data to be competitive. Our model is far superior to the MF, which demonstrates the ability of our architecture to extract relevant profile information from textual data.

3.2.4 Attention Visualization

One of the major advantages of our model is the possibility of using attention modules for introspection in the context of the sentiment classification task. Indeed, the attention vectors allow to isolate the discriminating elements (words and sentences) in the corpus. We proceeded as follows: the set of attention vectors of the RBA_w module of the entire n evaluation sentences is retrieved: $att_w = \{\alpha^{\mathbf{0}}, ..., \alpha^{\mathbf{n}}\}$. The index of the maximum value of each of these vectors indi-

3. Our experiments initially incorporated the hybrid thematic factorization model (J. McAuley and Leskovec 2013). However, the code provided by the authors consistently gives inferior results to the standard MF. For this reason, the results from this approach are not presented in the table below

cates which word represents the most interest in each sentence; these words are therefore the most discriminating. For the corpus *Video Games*, the words resulting from the introspection of the model are represented as a word cloud in Figure 3.4. Globally, the cloud shows feeling words (often very positive, in relation with the unbalanced distribution of the data presented in Table 3.1).

Some domain-related words (e.g. *game*, *playing*) also appear.

Figure 3.4 – **Cloud of the most discriminating words in the dataset *Video Games* according to the global attention vector**. The size is proportional to the number of times the word is the most important in a sentence of the corpus.

Introspection becomes more interesting when we consider separately the two components of attention $\mathbf{a}_g$ –general term– and $\mathbf{a}_{u,i}$ –personalized attention– (cf Equation 3.8). For general attention, as in the previous experiment, the extracted words are mainly generic polarizing words like *good*, *worst*, *love*(Figure 3.5 - left).

The Figure 3.5 (right) illustrates on the contrary the words put forward by personalization. By considering the whole population, we have highlighted the most frequently recurring words: personalization allows the model to focus on words that are much more specific than those coming from the generalist attention; these words are similar to attributes in the video game domain. We observe characteristic entities such as *Wolfenstein*, *Lara (Croft)*, *Zelda* or publishers such as *Sony*, *Blizzard*, *Capcom*. It is this information that, integrated with user and product profiles, allowed us to reach a performance threshold on the recommendation task.

Figure 3.5 – **Word clouds**. (Left) words discriminating for the model using only the generalist component of attention in the equation (Equation 3.8) – (Right) words only discriminating in the sense of the personalized component of attention in the equation (Equation 3.8).

3.2.5 Explaining suggestions with attention

One of the challenges that drives us to deal with recommendation in a textual space, beyond the improvement in score prediction, lies in the ability to automatically explain to the user of the system the suggestions that are offered to him. Our personalized sentiment analysis model should act as a filter to replace the user in reading product reviews: we want to automatically isolate the sentences and words that are likely to attract their attention.

To study this possibility, we ran our architecture on a user-item pair (Figure 3.6): the figure on the right gathers all the outputs of our two-headed model, a score prediction, an extraction of the important words, and finally, thanks to the attention module on the sentences, an extraction of the sentences important for the user among the reviews on the target item. The sentences presented are ordered by attention scores (in blue). The figure on the left shows what the user has actually written on this item [4].

Commercial recommendation systems are starting to integrate explanation mechanisms to get out of the black box logic. These explanations are generally centered on the target item. We demonstrate here the possibility of personalizing the explanations to each user of the system.

4. The attentions on the sentences written by the user himself are higher than on the sentences of other authors, which seems logical.

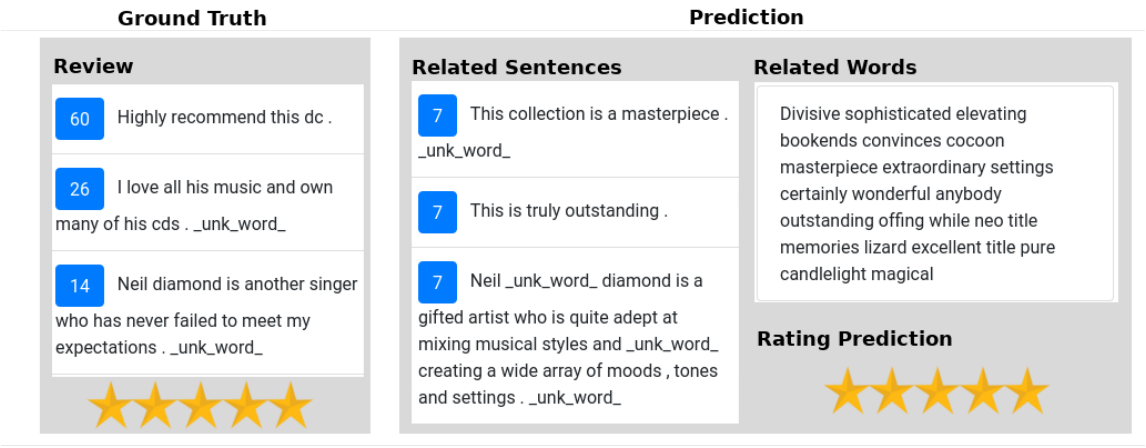

Figure 3.6 – **Example of attention analysis for recommendation explanation**. On the left, the review written by the user and its associated score. On the right, the different outputs of our system: the predicted score, the sentences we consider useful for the user –extracted from the reviews on the targeted item–, the keywords associated with the target.

3.3 Conclusion

In this paper, we focused on user and product modeling via online review corpora.

Our goal was to better integrate text in the recommendation process. Based on work in sentiment analysis, we presented an architecture composed of two neural networks operating in parallel and linked by an attention mechanism.

We have shown that such a model allows to improve the recommendation performance (accuracy of predicted scores) while offering an elegant way to explain the system's suggestions.

HRAN leverages the representation power of the MTL setting in which it is trained as well as the text-modeling properties of an attention mechanism, at a time where Bidirectional Encoder Representations from Transformers (BERT) did not exist.

Possible extensions of this work would replace the sentiment analysis, RBA-based part with a Transformer-based architecture. The model's performances could then benefit from the computational efficiency and the modeling power of a Transformer. It would indeed remove the need for stacked Recurrent Networks and still make the attention mechanism available for inference, assuming the Transformer's self-attention is coupled to the global attention vector.

CHAPTER 4

A NLP APPROACH TO PROFESSIONAL PROFILE LEARNING AND EVALUATION

Chapter abstract

Regardless of the use-case, a user profile is often composed of several fields or attributes of different nature.

Our work focuses on leveraging the natural-language fields of a user profile to build a representation that contains the information of the categorical attributes present in the user profile.

We set our work in the context of an under-studied field: Professional Profile Extraction – a crucial challenge for any Human Resources (HR) department.

In this paper, we propose an approach to learn and evaluate professional embeddings.

We first highlight the technical issues associated with this specific data. Then, we propose an architecture that compares different word embedding models to encode the textual information. Finally, we learn user profiles and propose three original evaluation tasks to illustrate the strengths and weaknesses of our approach.

The work in this chapter has led to the publication of a conference paper :

- Clara Gainon de Forsan de Gabriac, Vincent Guigue, and Patrick Gallinari (2020). "Resume: A Robust Framework for Professional Profile Learning & Evaluation". In: *ESANN 2020 - 28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. Bruges, Belgium. URL: https://hal.archives-ouvertes.fr/hal-02503464.

Contents

Learning to match candidates with job offers is a major challenge for any institution's HR department. The fast development of online job-boards (*Monster*, *JobTeaser* etc.) and professional social networks such as *LinkedIn* [1], makes this task increasingly crucial (Adikari and Dutta 2014). As such, improving profile modeling of users on their past experiences (jobs, education) may allow the development of new tools to suggest relevant skills and to connect unformatted job titles and descriptions to standardized ontologies like ESCO [2].

In this work, we propose a method to learn and evaluate professional profiles of users using the information contained in their LinkedIn profile. In doing so, we extend what could be considered a Recommendation problem (where we would aim at recommending an applicant the best suited job offer) into a new applicative domain: Professional Profile Learning.

Unlike traditional Expert Matching methods that mainly rely on categorical data (W. Tang et al. 2010; Karimzadehgan et al. 2008; Wenbin Tang et al. 2012), we aim at building meaningful professional representations using user-generated texts only (their job titles and descriptions, and their educational background) during training in a self-supervised setting. Our hypothesis is that a professional representation is well constructed if we can learn to retrieve users-specific information from it.

In other words, we want our profiles to encode a sufficient amount of information to predict the future of users' careers, the skills, and the industrial field associated with their profile. This approach is in line with the of Representation

1. https://www.linkedin.com/
2. https://ec.europa.eu/social/main.jsp?catId=1326

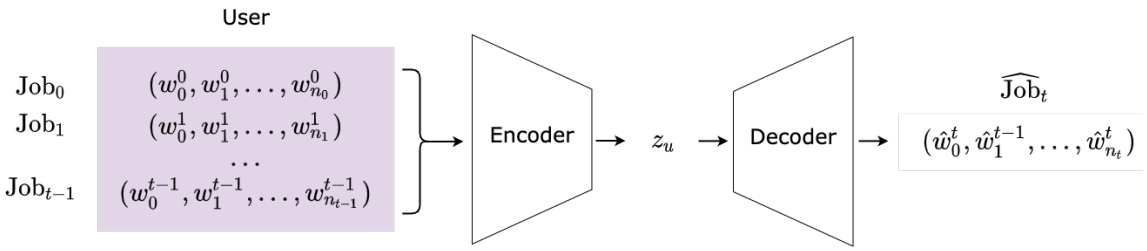

Figure 4.1 – **A high-level illustration of the Next Job Generation Task**. The user profile is built by encoding their t first professional experiences (or jobs). Then, a decoder is trained to generate the user's last job, $\widehat{\text{Job}}_t$. This formulation is a practical proxy for learning to predict a user's next job.

Learning (RL) and to some extent, Multi-Task Learning (MTL). We aim at leveraging the implicit links between all the data present in a profile to evaluate the constructed user representation. Besides, the originality of our evaluation method lies in the textual generation of a user's next professional experience (illustrated on Figure 4.1). We feel this task is a demanding and relevant one, as it can only be made possible by fully understanding the user's career. Note that we address the next job prediction task with a generative approach rather than a classification one for two reasons. Firstly, the high noise level of the data makes it irrelevant to turn professional experiences into classes (because no two people will write about the same experience in the same way). Secondly, the generative approach also provides the Explainable Artificial Intelligence (xAI) part of our model, as it outputs text that is understandable and possibly explainable with regard to the user's career, which would not have been the case with a classification task. This design choice however brought its lot of challenges, namely regarding the evaluation metrics, as explained in Section 2.3.3.

Related Work. The topic of professional profile representation learning has been gaining traction over the past years, either through fake profile detection (Adikari and Dutta 2014; Adikari and Dutta 2020), job recommendation (Kenthapadi et al. 2017) or job prediction (Liangyue Li et al. 2017; Paparrizos et al. 2011). Similarly to these works, we are interested in learning to represent a user's professional profile. However, we differ from those work on four main aspects.

Firstly, we only use the past experiences or the education of a user to learn their representation, then evaluate the representation with the remainder of the data available on the profile (skills, industry and next experience). We indeed propose an evaluation framework that can be applied and extended to broader applications (other social network profiles, recommendation...).

Secondly, while Kenthapadi et al. (2017), Liangyue Li et al. (2017), and Paparrizos et al. (2011) predict a user's next job, they only focus on the job title, when we also generate a job description. This enables us to better understand a user's individual professional career, and to identify their specificity, interests and so on more precisely.

Thirdly, our approach is closer to Text Summarization (Mani and Maybury 1999; Paulus et al. 2017) in the sense that we do not rely on supervision to learn profiles and use the BiLingual Evaluation Understudy (BLEU) score as an evaluation metric.

Lastly, those works do not study the professional profile representation of a person using only their educational background.

The robustness of our approach relies on recent advances in the field of Natural Language Processing (NLP) and in particular regarding embedding models. While text representation learning has long been performed at document level in a bag-of-words setting (Blei et al. 2003), Word2vec Mikolov et al. (2013b) allows us to predict words in a local context, opening the way for meaningful word embeddings and text generation applications. A second generation of word embedding models introduces solutions to take into account out-of-vocabulary words through subword information encoding (Bojanowski et al. 2016) and even more recent proposals focus on contextual embeddings and generative settings to improve sentence understanding (Peters et al. 2018).

Professional Profile Learning: A new application domain. Several technical issues make this problem difficult: professional and educational experiences are free texts: not only are they noisy and subject to typos, but they also prevent us from adopting a classification framework, as there are as many experiences as there are users. Moreover, aggregating different jobs to build a single user profile is non-trivial, as users often mention several times the same job before obtaining a promotion. Finally, we chose a challenging framework where the final tasks –skills, industry and next-job predictors– are used for evaluation purposes only, without helping to refine the profiles. We want to demonstrate our ability to encode relevant information in a very compact embedding from a noisy data source.

Our framework *Resumé* relies on robust word embedding models to encode textual information (Bojanowski et al. 2016; Peters et al. 2018). We then aggregate job or education embeddings obtained via those models to build a user representation. On top of this framework, our main contribution lies in the evaluation approach; we were provided with more than 700,000 LinkedIn pages (illustrated in Figure 4.2) which enables us to measure our ability to predict skills and industrial field for our set of users. We also provide an original Recurrent Neural

Network (RNN)-based generative approach for the next job prediction task as well as an evaluation procedure relying on a Neural Machine Translation (NMT) metric (Papineni et al. 2002a).

Ethics of Professional Profile Learning. Learning use profiles from such sensitive data is as ethically tricky as it is technically challenging. For starters, our user representations are necessarily biased because of the LinkedIn profiles have been scraped at a given time, yielding incomplete careers, buzz words that do not reflect today's trends and so on. Besides, while our data do not contain the name of the users, they are not totally anonymous since they contain personal information like high school names and job locations. Moreover, our data are biased by the fact that they come from LinkedIn profiles, which means that they represent people with an access to the internet and an ease of use of social networks. They may also be biased by the fact that they might contain profiles of people that have suffered from discriminations in their professional life.

The biases of this data coupled with the inherent opacity of Machine Learning (ML) models make it is easy to imagine how the work presented in this manuscript could be used to harmful ends. One could design a model that learns to discriminate people coming from a certain region, country or even high-school and deflect the blame onto the learned model, arguing that algorithms cannot be xenophobic.

However, learning professional user profiles can also do a lot of good. From a HR standpoint, professional user profiles could be used to better match job offers with applicants, all the while adding some novelty and serendipity to the traditional process (which often consists in selecting candidates that have similar backgrounds than actual employees). For an employment agency, it could be used to uncover discriminative behaviors from companies, as well as an overview of the job market at a given time. Finally, from a user's point of view, it could be a tool for highly personalized professional coaching, giving insights regarding what training would help them attain the job they want.

In this chapter, we introduce our models, detail the experimental settings in which we built our representations and, finally, we show that the noise level in the raw data leads to a surprising ranking of our approaches.

4.1 Models

Each user's raw profile consists in a set $\mathcal{E}_u$ of chronologically ordered textual experiences: $\mathcal{E}_u = \{e_0, \ldots, e_T, e_{T+1}\}$. An experience can either be a professional (it is then referred to as a "job") or an educational one ("edu"). The user is also

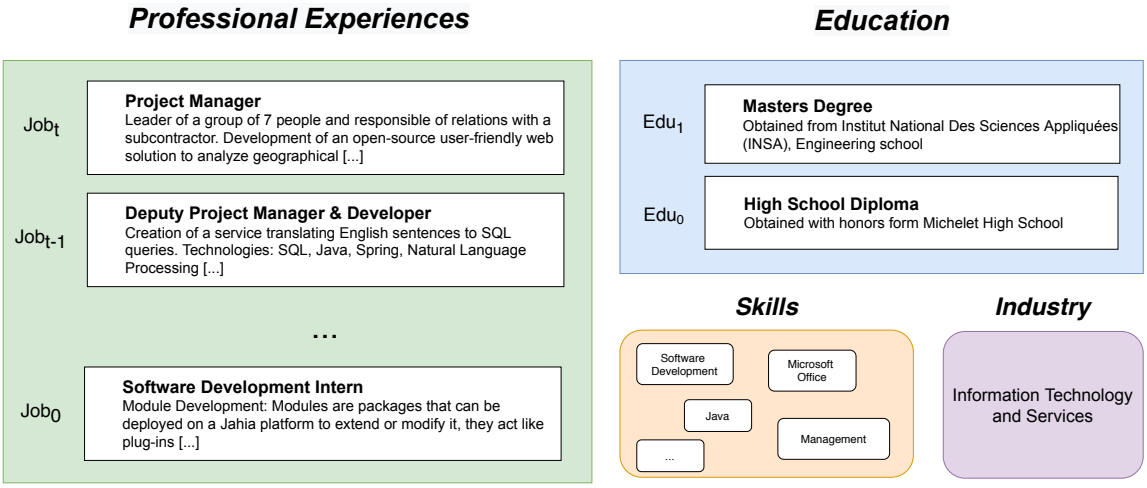

Figure 4.2 – **A user profile schema**. A user is composed of their professional experiences, their education, their industry and their skills.
Note that the skills and industry are categorical values, whereas both the professional experiences and the education are free texts.

associated with a set of skills described as a binary vector in the skill domain: $\mathbf{s}_u \in \{0,1\}^S$. The industrial field is denoted $b_u \in \{1,\dots,B\}$.

Our models are composed of an experience encoder that deals with raw texts and an experience aggregator that outputs a user profile: $\mathbf{z}_{e_t} = \text{enc}(e_t)$, $\mathbf{z}_u = \text{agg}(\mathbf{z}_{e_0},\dots,\mathbf{z}_{e_T})$.

Then we train three independent components tackling the next job prediction task $\widehat{e}_{T+1} = \text{dec}(\mathbf{z}_u)$, the skill prediction $\widehat{\mathbf{s}}_u = f_s(\mathbf{z}_u)$ and the industrial field categorization $\widehat{b}_u = f_b(\mathbf{z}_u)$.

4.1.1 Text, Past Experiences and User Representations

We aim at representing a user's professional information using only their past experiences. Such data is noisy and contains a lot of out-of-vocabulary or rare tokens (e.g., product names, degree names, misspelled words ...). We thus choose text-encoding models capable of leveraging subwords information: FastText (Bojanowski et al. 2016) and Embeddings from Language Models (ELMo) (Peters et al. 2018).

Word Embedding models. FastText is a word embedding model based on a skip-gram formulation and optimized using negative sampling. It is more robust than Word2vec since it relies on subword encoding. Each character n-gram will correspond to a $\mathbf{z}_g$ embedding and a word embedding is simply the sum of its

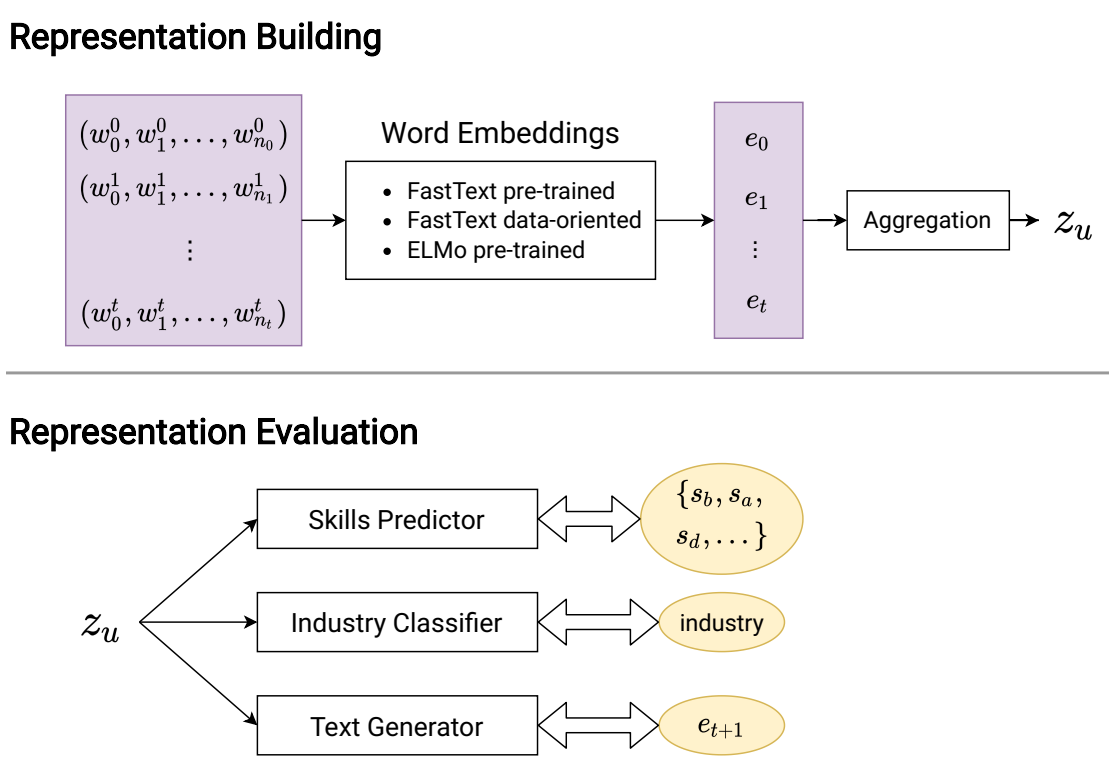

Figure 4.3 – **A schematic representation of our architecture**.

subwords' representations $\mathbf{z}_w = \sum_g \mathbf{z}_g$. Note that the word itself is part of its set of n-grams. At the word level, the skip-gram formulation is implemented using a Binary Logistic Loss sliding on the text of size T with a context-window C_t for the word w_t:

$$\text{BLL}(w) = \sum_{t=1}^{T} \left[\sum_{c \in C_t} \ell(\mathbf{z}_{w_t}^\top \mathbf{z}_{w_c}) + \sum_{\bar{c} \notin C_t} \ell(-\mathbf{z}_{w_t}^\top \mathbf{z}_{w_{\bar{c}}}) \right] \tag{4.1}$$

with $\ell(x) = \log(1 + e^{-x})$. As FastText is light and easy to train, we will compare pre-trained (referred to as FT_{pt}) and specifically trained embeddings (FT_{CV}, FT_{edu}) on our different tasks.

ELMo is a recent word embedding model relying on contextual embeddings: words' representations depend on their contexts. In practice, $\mathbf{z}_w$ is obtained after running a Bidirectional Recurrent Neural Network (BiRNN) over the text: $\mathbf{z}_w$ is the representation of previous words until w in one direction and of the following words in the other direction. As opposed to FastText, ELMo relies upon millions of parameters and we will only use the pre-trained version of this word embedding model.

Profile representation. In this work, an experience e_t is simply encoded by averaging all its words' representations (Equation 4.2, Equation 4.3). Then, we represent users as an aggregation of their experiences (eq. Equation 4.4).

$$e_t = (w_1^{(t)}, ..., w_N^{(t)}) \tag{4.2}$$

$$\mathbf{z}_{e_t} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{z}_{w_n^{(t)}} \tag{4.3}$$

$$\mathbf{z}_u = \frac{1}{T+1} \sum_{t=0}^{T} \mathbf{z}_{e_t} \tag{4.4}$$

Note that in the case of ELMo representations, individual word representations $\mathbf{z}_{w_n^{(t)}}$ are not averaged into $\mathbf{z}_{e_t}$. We rather define $\mathbf{z}_{e_t}$ as the representation of the word sequence, that is $\mathbf{z}_{e_t} = \mathbf{z}_{w_N^{(t)}}$ with N the length of experience e_t.

4.1.2 Tasks, Predictors and Decoder

We evaluate the meaningfulness of our user representations through three tasks: the prediction of their skill set, the prediction of their industrial field and the generation of their next job. We refer to them as the Skill Predictor, the Industry Predictor, and the Next Job Decoder.

4.1.2.1 Predictors

Since we aim at evaluating our embeddings' quality, we made the predictors simple. Both the Skill Predictor f_s and the Industry Predictor f_b consist of two linear layers separated with $\tanh$ activation functions and followed by a sigmoid for the former and a softmax for the latter:

$$f_s(\mathbf{z}_u) = \text{sigmoid}(W_2^\top(\tanh(W_1^\top \mathbf{z}_u)))$$
$$f_b(\mathbf{z}_u) = \text{softmax}(U_2^\top(\tanh(U_1^\top \mathbf{z}_u)))$$

When predicting a user's skill set, we are interested in predicting all the right skills and only the right skills, placing us in a multi-label classification task. This predictor is trained to optimize a Binary-Cross-Entropy Loss function. The industry prediction is a simple mono-label classification task. The Industry Predictor is trained to optimize a Cross-Entropy Loss function.

4.1.2.2 Decoder

The Next Job prediction, however, is a harder task: it cannot be addressed as a classification task since the number of unique jobs is of the same order of magnitude as the number of users. Thus, we choose to generate the title and the description of a user's next job e_{T+1} from their representation $\mathbf{z}_u$.

In this context, we implemented a RNN with a Long Short-Term Memory (LSTM) as the recurrent cell, followed by a linear layer (of weights V), that decodes $\mathbf{z}_u$ into a sequence of words $\widehat{e}_{T+1}$. At each time step (Equation 4.5), we feed the decoder both the user representation $\mathbf{z}_u$ and the last-predicted token.

$$\text{dec}(\mathbf{z}_u, w_n^{(T+1)}) = V^\top \text{LSTM}([\mathbf{z}_u, w_n^{(T+1)}]) \tag{4.5}$$

$$= \widehat{w}_{n+1}^{(T+1)} \tag{4.6}$$

It is trained to optimize a Cross-Entropy Loss function between the predicted word $\widehat{w}_n^{(T+1)}$ and the label $w_n^{(T+1)}$, for every word in the actual sequence.

4.2 Experiments

4.2.1 Dataset presentation

Our dataset consists of over 700,000 LinkedIn user profiles, and almost 5,000,000 professional experiences. General information concerning the dataset are reported in Table 4.2.1. Finer statistics about both professional and educational experiences can be found in Table 4.2.1.

Global Dataset Information	
Total Profiles	740,983
Train Split	445,097
Valid Split	147,909
Test Split	147,977
Skills Number	523
Industries Number	150

Table 4.1 – **General dataset information**. The training, validation and test splits are built following a 60/20/20 partition.

	Number of experience per profile		Experience sequence length	
	Professional	Education	Professional	Education
Average	5.8	2.4	47.5	10.7
Median	5	2	34	10
Min	3	1	5	5
Max	8	4	64	64

Table 4.2 – **Table of experience information**. The left part presents statistics of experience number throughout the dataset. The right part presents information about the length of the experiences.

In this work, we are interested by 4 types of information on a LinkedIn profile: their professional experiences, their education, their skills, and the industrial domain they work in.

Figure 4.4 is a screenshot of a real-life LinkedIn profile. It perfectly illustrates both the richness and the challenges of our dataset.

The professional experiences of a user can be of very variable lengths, depending on the user's age, their personal tendency to change jobs and their industrial domain. One can note that they are also formatted in a very specific way: they contain a lot of abbreviations, proper nouns (product names, for instance) and are often organized in short, non verbal bullet points.

On the other hand, the educational background of a user tends to be more concise, because their is no "description" field that the user can fill in. This example highlights an interesting aspect of our data, which the fact that even if the user did fill their profiles in English, the education section is quite often composed of country-specific degrees and untranslatable institution names.

Finally, Figure 4.4 illustrates how repetitive and redundant the skills of a profile can be: the user has the skills *Digital Marketing, Marketing Strategy, Online Marketing*, etc. Of course it makes sense for a user to add as many keywords as possible, so as to appear in as many queries as possible, thus making our dataset quite noisy.

There are more than 95,000 different skills cited in the dataset but we only retain those appearing in at least 3000 profiles, leaving us with 523 skill classes. The retained skills are thus neither exhaustive nor disentangled. Dataset exploration revealed on the contrary a certain overlap in skills due to the fact that user try to be as precise and covering as possible when filling them. Similarly, we kept the 150 most common industrial fields, also referred to as industries.

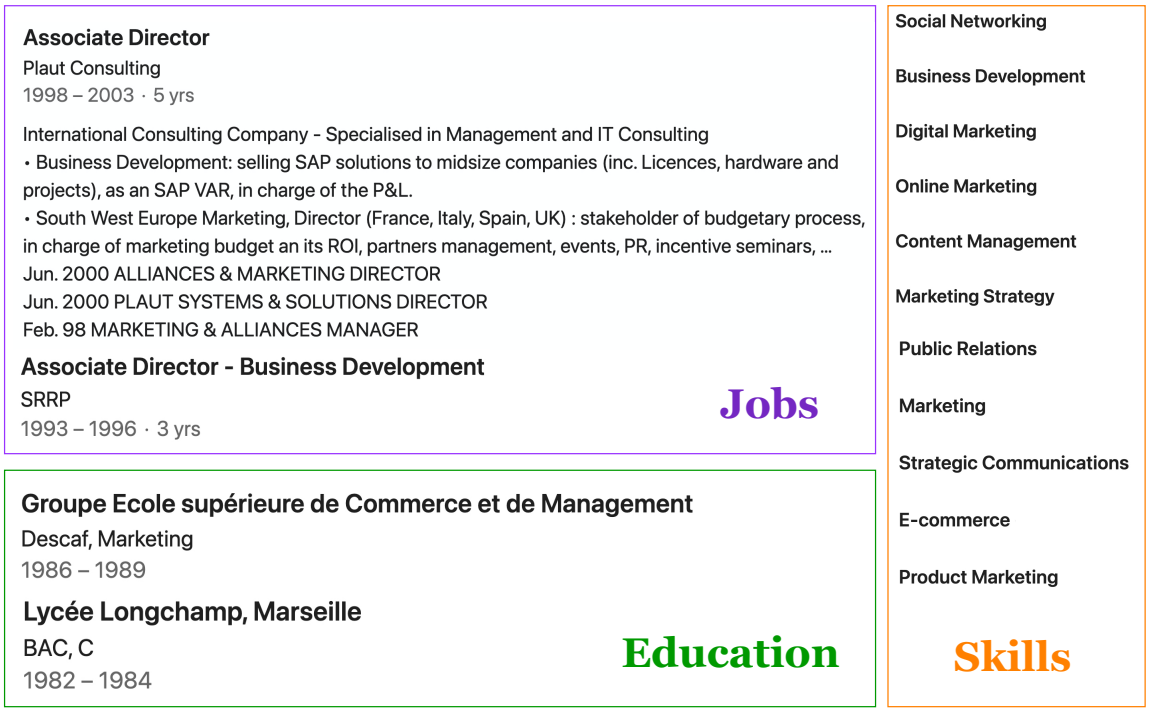

Figure 4.4 – **A LinkedIn Profile's screenshot**.

4.2.2 Experiments

We compare three different word embedding models and study the predictability of a career given a person's past education and past jobs.

Our three user representations rely respectively on a FastText model trained on our data FT_{CV}, a pre-trained FastText FT_{pt} [3] and a pre-trained ELMo. [4]

Since our primary goal is to learn a user's professional representation, using a user's past jobs is a natural choice of input. However we are also interested in studying the impact of a person's education on their career, leading us to apply the same framework of professional representation to the educational experiences of a person (the degrees they obtained, and the institution they got it from).

In both cases, we aim at building a representation that is rich enough so that simple models can learn to retrieve skills and industrial fields from it, as well as predict the user's next job. Whether we consider the jobs or the education of a user does not change the training setup of either the Skill Predictor or the Industry Predictor, but when it comes to the next jobs prediction we have to adapt the task. In the case where we take jobs as an input, the next job prediction label is the latest professional experience on a user's profile. When we build a user's

3. https://fasttext.cc/docs/en/crawl-vectors.html
4. https://github.com/allenai/allennlp/blob/master/tutorials/how_to/elmo.md

representation from their degrees, we use their first professional experience as a label for the next job prediction task.

4.2.3 Training details

4.2.3.1 Trained FastText Models

Both FT_{CV} and FT_{edu} are trained in an unsupervised setting with a dimension of 300. All other parameters are FastText library's defaults.

4.2.3.2 Evaluation Models

Common parameters. All models are implemented using the Deep Learning framework Pytorch (Paszke et al. 2019). The data is split according to the ratio reported in Table 4.2.1. The Text Generators are trained with a vocabulary of 40,000 words.

Experiments on jobs data. All three evaluation models (Skill Predictor, Industry Classifier and Next Job Predictor) are trained on a maximum of 100 epochs with an early stopping mechanism (with a patience of 3), an Adam optimizer and a learning rate of 1e-3.

Both classifiers have a batch size of 128. The size of their hidden layer is 300.

The Next Job Predictor has a batch size of 80 when trained on ELMo representations, 160 when trained on FastText ones. It has a drop out ratio of 0.5, the hidden size of the LSTM cells is 256.

Experiments on education data. Both classifiers are trained on a maximum of 100 epochs with an early stopping mechanism (with a patience of 3), an Adam optimizer and a learning rate of 1e-1. They are trained with a batch size of 16 and a hidden layer of size 300. The Next Job Predictor is trained similarly to the one trained on the jobs data.

4.3 Results

In this section we present and analyze the results of our experiments. Tables are divided in three parts: the upper part is the baselines results, the middle part reports the results on the jobs, and the bottom part shows the results on the education, except for Table 4.5 where the education does not appear.

	Accuracy		Precision		Recall		F1	
	Top 1	Top 10	Top 1	Top 10	Top 1	Top 10	Top 1	Top 10
Random	0.68	6.80	N/A	N/A	N/A	N/A	N/A	N/A
Most Common	6.25	31.32	0.39	25.59	6.25	31.32	0.74	26.03
				Professional Profiles				
FT_{CV}	38.40	79.87	37.49	81.47	38.40	79.87	36.38	79.24
FT_{pt}	35.55	77.09	34.61	79.20	35.55	77.09	33.18	76.27
ELMo	**39.18**	**80.37**	**39.31**	**81.91**	**39.18**	**80.37**	**37.22**	**79.75**
				Education				
FT_{edu}	**14.60**	**53.66**	**21.02**	**79.82**	**14.60**	**53.66**	**13.24**	**55.45**
FT_{pt}	12.12	51.45	20.50	78.75	12.12	51.45	10.51	53.03
ELMo	13.58	44.83	14.57	66.84	13.58	44.83	9.73	41.89

Table 4.3 – **Industry classification results on 150 classes**. Each column reports the metric on the test set, either on top 1 or top 10. For each metric, the higher the score, the better.

In the upper part of each table, "MC" stands for "Most Common", which is the model that always outputs the majority class (for industry classification) or classes (for skills prediction).

The two bottom parts of the tables are themselves divided into three rows, each accounting for a specific embedding model representation:

- FT_{CV} and FT_{edu} stand for FastText models trained on our dataset (professional and educational respectively),
- FT_{pt} is a pre-trained FastText model, [5]
- ELMo is a pre-trained ELMo model. [6]

4.3.1 Quantitative Analysis

The results of the Industry Classification task are reported in Table 4.3, those of the Skills Prediction task in Table 4.4.

We can observe that ELMo achieves the best results, both in Skills Predictions and Industrial Classification when it comes to professional profiles. However, the FastText CV-oriented model achieves close results. Our intuition is that while ELMo performs especially well in disambiguation, professional experiences are

5. https://fasttext.cc/docs/en/crawl-vectors.html
6. Available at https://allennlp.org/elmo

	Hamming	Precision	Recall	F1
Most Common	3.00	24.45	23.06	23.73
		Professional Profiles		
FT_{CV}	2.44	36.82	39.26	34.63
FT_{pt}	2.39	36.54	34.83	32.31
ELMo	**2.33**	**37.72**	**37.76**	**34.85**
		Education		
FT_{edu}	**2.91**	5.94	23.10	9.19
FT_{pt}	3.50	6.64	**28.13**	**10.39**
ELMo	10.34	**6.74**	7.88	5.67

Table 4.4 – **Skills Prediction Results on 523 classes**. For each metric, the higher score, the better, except for the Hamming loss, where a lower score indicates a higher resemblance to the ground truth.

written in a way that leaves more importance to rare words than ambiguous ones. This allows FastText models to achieve comparable performances, and to be considered a serious alternative to ELMo in situations where training time is critical.

Another interesting trend that emerges from our results is that professional experiences of a user are more informative than their education when it comes to predicting their skills or industrial field. Although it is relieving to see that one's education does not completely determine one's career, we observe that the FT_{edu} models still achieves some reasonable performances on the industry classification.

	BLEU score				
	BLEU	BLEU-1	BLEU-2	BLEU-3	BLEU-4
FT_{pt}	1.91	20.6	3.5	0.8	0.2
FT_{CV}	**2.15**	22.2	**3.8**	**0.9**	**0.3**
ELMo	1.74	**22.5**	**3.8**	0.6	0.2

Table 4.5 – Experimental results on job generation (title & description)

Table 4.5 details the BLEU score (Papineni et al. 2002a) for our text generation task. The idea of the BLEU score is to count the number of n-grams that the generated sentence has in common with the reference. The BLEU score represents the overall resemblance between the generated sentence and the reference, and the BLEU-N scores account for the number of common n-grams between the generated sentence and the reference. Let us notice that the BLEU score expresses

a degree of common words between sentences, rather than an actual semantic similarity.

On this task, we can notice that the CV-oriented FastText performs better than both ELMo and the pre-trained FastText. Note that we did not report the next-job generation results for the education dataset because the models systematically output a single generic first job ("Intern") with no real job description.

This ranking among our approaches is surprising, it points out that the CV style does not follow a classical language model: a robust and dedicated model is required to tackle misspellings, abbreviations, ellipses, acronyms that characterize the fast writing style observed on CV. The same kind of conclusions has been drawn in P. Jain et al. (2019).

4.3.2 Qualitative Analysis

Industry Classification. When taking a closer look at the industry predictions, we notice situations where the model misclassifies a profile:

- The actual industry is semantically very close the the predicted industry;
- The profile is assigned a label that a human could not predict, because the label seems inconsistent with the career.

We report examples of those situations in Table 4.6. Note that misclassification may also happen when a user profile is what we call "eclectic", that is to say when a user's jobs are low-qualified and unrelated to one another (i.e., there is no "career path" emerging from their profile). This situation rarely occurs in the education dataset, probably because people tend to emphasize their professional experiences more than their education, which they keep short and consistent.[7]

Text Generation. The next job generation highlights the difficulty for our models to generate long job descriptions as well as very specific sentences. And example of generated text is illustrated in Figure 4.5. It is interesting to note that, even though all models seem to get stuck in a generation loop quite quickly, the FastText CV model expresses the communication aspect where the other models see a client relationship management.

The main point of this example is that the 3 models do express the marketing aspect of the job, which means they all could somehow capture the career

7. The median number of professional experiences per person on our dataset is 5 while the median number of education experiences is 2.

#	Profile		Predicted Industry	Actual Industry
	Degree	Institution		
(1)	Engineer, computer science.	Polytech' paris-sud.	Computer Software	Information Technology and Services
(0)	Exchange program, computer science.	École Polytechnique of Montreal.		
(1)	Specialized sales, merchandising, marketing activities.	Sedima.	Marketing and Advertising	Farming
(0)	High School Diploma in agricultural mechanics	Gustave Eiffel High school.		

Table 4.6 – **Examples of misclassified profiles**. The first column from the left is the rank of their experience in the profile (the higher, the more recent). The second column divides the profile of the user in its degree and institution (the place where the degree was obtained). The last two columns report the predicted industry of the user and their actual industries.
The first profile illustrates the case where the predicted and the actual industries are semantically close. The second profile shows a profile that would be hard to classify for a human without more information.

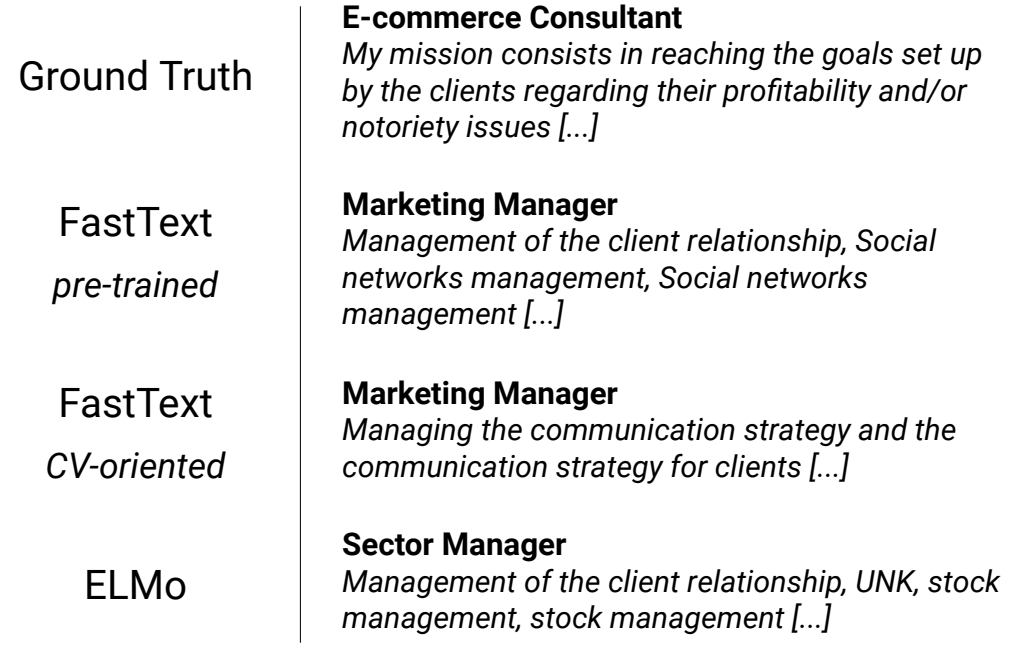

Figure 4.5 – **Illustration of text generated by our decoder**. *Ground Truth* is the actual label of the profile. The bold font indicates the job title, the text in italic is the job description.

evolution of the user, since all models only had access to the person's previous jobs.

Such predictions highlight the complexity and diversity of our data. While not human-like, those predictions can add a lot of value to a CV database as they capture the essence of a career. The analysis of both skills and industry prediction for all three models indicate that a consequent part of the wrong predictions makes sense to a human reader. For instance, a profile containing the skills *Office Pack*, *Photoshop* and *Marketing* is predicted to have the *Microsoft Word*, *Adobe Photoshop* and *Digital Marketing* skills. Similarly, a Developer working in the Pharmaceutical Industry can be either predicted in the "IT" or the "Pharmaceutical" industry. Those observations lead us to believe that the representation of our users is rather satisfactory. Most prediction errors are understandable and could be tackled by a more thorough data pre-processing.

4.4 Conclusion

We propose a novel approach to professional profile learning, *Resumé*, that is self-supervised and relies on free text, along with three evaluation tasks. We compare the impact of the use of a contextual word embedding model to the use of a

more classical one on our evaluation tasks, at a time where the Transformer architectures were not readily available. Our experiments show that using a light word embedding model to represent users is not only sufficient to model their professional information but also compares with if not outperforms heavier architecture on all of our tasks. We also study the impact of the educational background of a user in the pre-determination of their career, and find that the education of a person alone is rarely sufficient to predict either their skills or their first job. It can however be used to determine the industrial field that the user is going to work in.

From a broader perspective, we proposed a framework that uses user profiles' textual traces to build a user representation, and the reminder of the profile to evaluate the richness of the learned user embedding. The conducted experiments confirm the intuition that a user's textual traces are very rich and can be used to describe them, even when using early models such as FastText or ELMo. This framework can be applied to a lot of different use cases, as long as it involves user profiles, thus offering a lot of possibilities.

In future work, we aim at modeling personalized user dynamic as a mean to better understand how their tastes and needs evolve through time. We detail two possible interesting architectures to do so in Chapter 5.

An immediate improvement of the framework would be to train it in an end-to-end fashion, thus leveraging the MTL framework to fine-tune the word-embeddings representation. Besides, it could be interesting to compare those word-embedding models with more recent architectures, such as Bidirectional Encoder Representations from Transformers (BERT).

Another appealing direction would be to address the representation of even more complex entities, such as companies and especially the prediction of even more elaborate signals, like fund-raising or company expansion.

Finally, one could extend the *Resumé* framework to build a job training model, helping users reach their career objectives by indicating the skills they should acquire.

CHAPTER 5

USER DYNAMIC MODELING

Chapter abstract

The two previous chapters proposed methods to learn and represent critical aspects of a user: their tastes, their professionnal experiences, skills and industrial domain.

While there is a form of forecasting in the next-item and next-job prediction tasks, we want to go further in our understanding of the user and learn the way change over time.

For this reason, we want to learn and model a user's dynamic, or evolution. That is to say, we would like to be able to understand how a user is going to change in time. This idea relies on the assumption that a user's tastes and profile can change as they grow older or become more expert in a given domain.

This chapter presents two exploratory architectures to achieve user dynamic modeling as a perspective for future work in User Representation (UR). The work presented in this chapter is still ongoing, and we detail the difficulties faced.

Contents

Modeling user dynamic is a crucial challenge for many applications, but it has seldom been studied from a Natural Language Processing (NLP) point of view.

Namely, J. J. McAuley and Leskovec (2013) used this hypothesis to learn how the expertise level of a user could affect their opinion on beers. In the context of Human Resources (HR), such modeling could allow to refine the prediction of a person's next job, by refining their professional trajectory via their seniority in their field. It could also be used to learn how to disentangle the personal dynamic of a person from the dynamic of the industry they work in, once again refining the user representation.

In this chapter, we propose an exploratory approach to User Dynamic Modeling. We look at this problem from a Professional Profile point of view. In other words, we are interested in modeling a person's career dynamic or trajectory. At a larger scale, such modeling would allow us to compare different types of profiles (for instance, those that obtain promotions quickly and those who do not) but also to compare the impact of the industrial domains and/or the skills on a person's career. This could answer questions such as "Is it longer to gain experience in Information Technologies that in other industries?" or "Should I learn Java rather than C in order to get a promotion faster?".

One of the main difficulty is to find temporally labelled textual data, i.e., user profiles which attributes would change over time. Another challenge is to formulate a reasonable assumptions that can link those textual data to some form of evolution. In our case, we want to explore the career evolution of a user by associating to each of their job j_i with an expertise level e_i. Our assumption is that throughout their career, users will gain knowledge and expertise in the field they work in. This implies that their expertise sequence must be monotonic, as a user cannot lose experience from one job to another (they may however not earn any).

In the following sections, we describe two architectures that rely on the aforementioned hypotheses. They aim at using a user's dynamic to predict their next jobs, as well as the level of expertise associated with such jobs. Both of them contain a Bidirectional Encoder Representations from Transformers (BERT)-based text-generation module as a way to ensure a correct user (or job) representation learning. The idea of those architectures is to learn what pertains to a user and what pertains to the industrial domain when it comes to a person's career evolution. The first one builds on the Back Translation (BT) mechanism and the other on a Variational Auto-Encoder (VAE). Those architectures are part of an ongoing work. We also describe the challenges and obstacles faced during the implementations of those models.

5.1 Job Expertise Rewriting

This first architecture is inspired from the work of Lample et al. (2018), in which they re-generate (or rewrite) textual reviews by changing their attributes (in particular: polarity, age of the author *etc*.).

We extend their idea to the application of job prediction. In this setting, the text we want to rewrite is a user's job, and its attributes are the industrial domain of the user, as well as the expertise level associated to the job.

Such a model would allow for two distinct applications. On the first hand, the rewriting of a job with a higher expertise level could be a proxy for the user's next professional experiences or promotions. On the second hand, rewriting users careers' evolutions in different industrial fields could allow us to distinguish between different types of users in terms of dynamics (some users tend to gain a lot of expertise fast, others do not) but also between industries (some have a lot of turnover and thus a do not favor experience gain in the short term *etc*.).

The intuition behind this architecture is that rewriting a job with another expertise level in a given industry will amounts to understanding what makes a job's expertise. At the latent level, the model will have to learn what makes a job an entry-level position rather than a senior one, in order to rewrite it with a different expertise level. At the human level, the model will produce understandable and possibly explainable sentences that can lead to a lot of interesting qualitative results.

5.1.1 Model

The proposed architecture is presented on Figure 5.1. It is composed of two parts: the DAE and a BT module. Note that both modules actually share the same weights (that is, the encoder of the DAE is the same than the of the BT.)

While Lample et al. (2018) use a Bidirectional Recurrent Neural Network (BiRNN) as the Encoder and an attentive Recurrent Neural Network (RNN) as the decoder, the Job Rewriter model could leverage recent, possibly pre-trained, Transformers architectures such as BERT to encode and decode the jobs.

5.1.2 Training

Let $j = [w_0, w_1, ..., w_{N-1}, w_N]$ be a word sequence of length $N + 1$ describing a textual job description. Its corresponding set of attributes consists of a industry and an expertise level such that $\text{att} = [\text{ind}_j, e_j]$. This set of attribute is converted

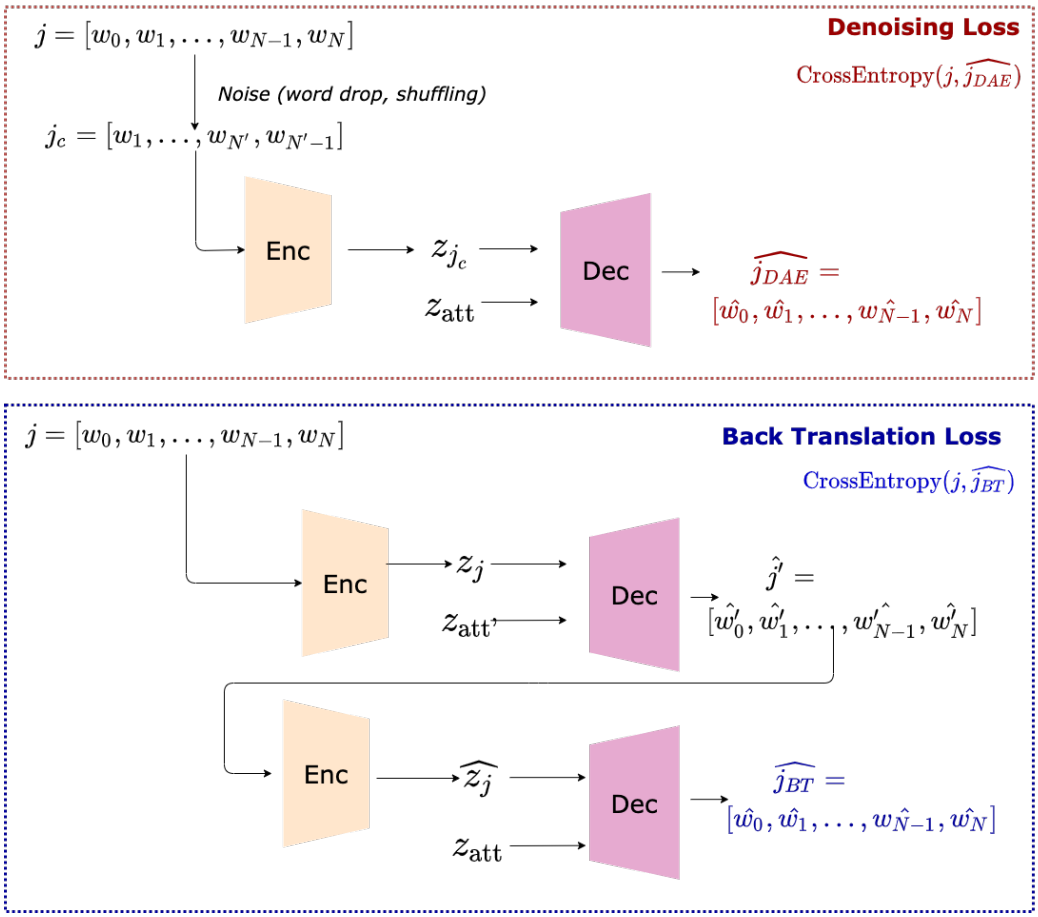

Figure 5.1 – **Illustration of the Job Attribute Rewriting Model**. It is constituted of two parts: a DAE and a BT module.

to embeddings $\mathbf{z}_{\text{att}}$ and then fed to the decoder to serve as conditioning for the text rewriting.

The DAE part ensures that the decoder can reconstruct text j with attribute $\mathbf{z}_{\text{att}}$ from a partial, corrupted representation $\mathbf{z}_{j_c}$. This is to ensure that, in the BT part, the decoder can generate $\hat{j'}$ or $\widehat{j_{BT}}$ from a potentially noisy representation by leveraging the conditioning information contained in $\mathbf{z}_{\text{att}'}$ and $\mathbf{z}_{\text{att}}$ respectively. Note that in the first part of the BT part, the initial job representation $\mathbf{z}_j$ is fed to the decoder with a conditioning $\mathbf{z}_{\text{att}'}$ that has been randomly sampled across the dataset so that j can be re-written with a new set of attribute into $\hat{j'}$

The model would be trained to optimize a composite loss $\mathcal{L}$ that combines both the Denoising and the Back Translation Losses:

$$\mathcal{L} = \alpha \mathcal{L}\text{oss}_{DAE} + \beta \mathcal{L}\text{oss}_{BT} \tag{5.1}$$

with α and β hyper parameters to tune.

5.1.3 Inference & Evaluation

Because we have no actual labels for the rewritten jobs $\hat{j}'$, we resort to a mix of metrics for evaluation. Once the model has converged, we will evaluate the quality of rewritten jobs $\hat{j}'$ through three metrics: the BiLingual Evaluation Understudy (BLEU) score, the perplexity metric, and the F1-score. The BLEU score serves as an evaluation of how many ngrams we kept from the original job j.

In this precise setting, the objective would not be to get a perfect BLEU score (which would mean that we only recopied j into j', thus missing the attribute-rewriting objective) but rather to make sure it is reasonably high.

The perplexity measure would be used as an indicator of the generated text's *fluency*, i.e., whether j' look like it could have been written by a human. It is a proxy to evaluate the quality of the generated text, regardless of the attribute-rewriting objective.

Finally, the F1-score would be used to evaluate how well we can retrieve attributes e' and ind$'$ (rather than e and ind) from j'. It would serve as a quantitative evaluation of the conditional attribute-rewriting objective.

5.2 Industry Latent Space Structuring via VAE

VAEs are a widespread type of generative architecture. They are especially used for their ability to structure their latent representation space.

The idea of a VAE architecture is motivated by the knowledge that entangled representations are quite hard to work with in the context of next job prediction.

In a lot of conducted experiments, it has become apparent that a naive Mean-Squared Error (MSE)-trained model failed to auto-encode a given job, regardless of the Auto-Encoder (AE)'s architecture (Multi-Layer Perceptron (MLP), RNN...).

Our intuition is that a model could learn a professional trajectory if the latent space was structured, or at least, disentangled. We leverage the structuring properties of VAEs in order to achieve the convergence of such a space, and then allow a model to learn transitions between points (i.e., jobs) in that space.

Like in Section 5.1, we dispose of jobs $j = [w_0, ..., w_N]$ that are associated with an industrial field ind_j and a level of expertise e_j. In this project, we aim at learning a latent space that is structured at two different levels: the industry and the expertise level.

5.2.1 Model & Training

The expected latent space is illustrated on Figure 5.2. The first part of the project

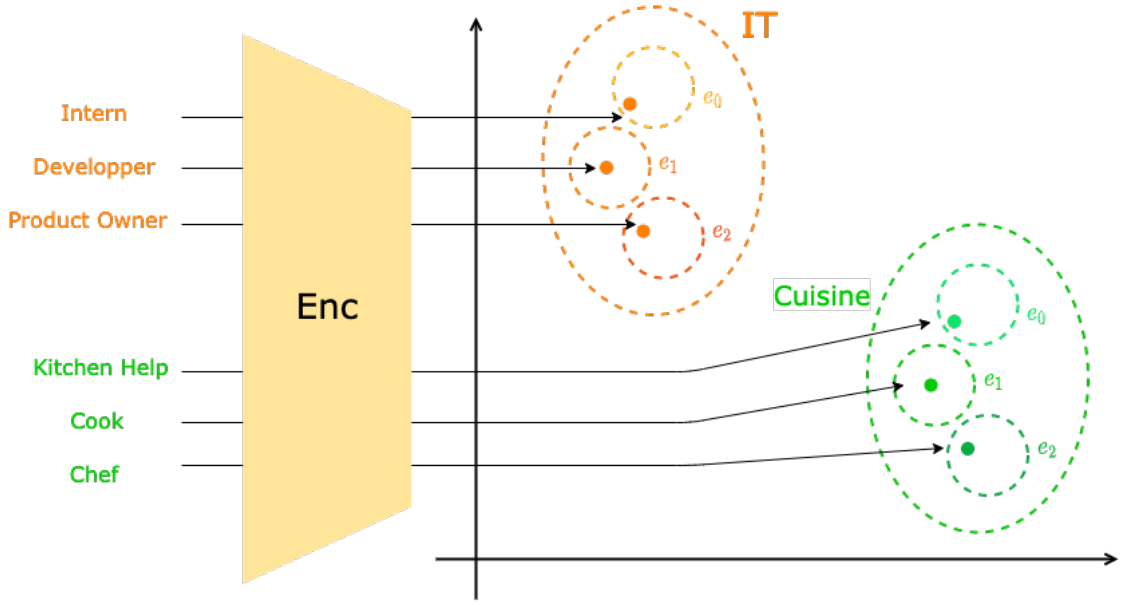

Figure 5.2 – **Illustration of the expected latent space**. The VAE model will have learned to structure the latent space by industry, and to structure each industry subspace by experience level.

consists in an encoder-decoder couple for the VAE and a standalone word decoder, as illustrated on Figure 5.3.

The encoder Enc_{VAE} is fed a representation of the job it has to encode, along with a marker of the corresponding industry and experience level. The decoder Dec_{VAE} will learned to reconstruct the input from a point of the latent space sampled with the reparametrizing trick. This encoder-decoder couple would be trained by optimizing a classical VAE Loss:

$$\begin{aligned}\mathcal{L}\text{oss}_{\text{VAE}} &= \mathcal{L}\text{oss}_{\text{rec}} + \mathcal{L}\text{oss}_{\text{KL}} \\ \mathcal{L}\text{oss}_{\text{VAE}} &= \text{NLL}(\mathbf{z}_{\text{job}_i}, \widehat{\mathbf{z}_{\text{job}_i}}) + \text{KL}\left(q_\phi(\mathbf{h} \mid \mathbf{z}) \| p(\mathbf{h})\right)\end{aligned}$$

with $q_\phi(\mathbf{h} \mid \mathbf{z})$ the distribution of the latent representation h knowing input $\mathbf{z}$, and $p(\mathbf{h})$ the distribution of the latent space variable, following the notations of Section 2.3.1.4.

Once the VAE converged, the decoder $\text{Dec}_{\text{words}}$ would learn how to reconstruct the input job job_i from its representation in the structured latent space, $\mathbf{h}_{\text{job}_i}$. It would thus optimize a reconstruction Loss:

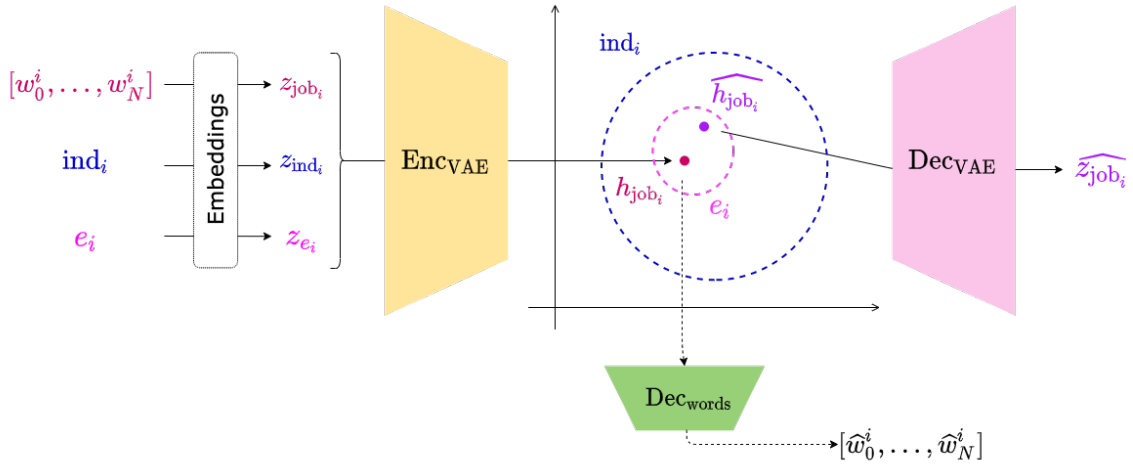

Figure 5.3 – **Illustration of the VAE training procedure.**

$$\mathcal{L}\text{oss}_{\text{words}} = \text{CrossEntropy}(\mathbf{x}, \mathbf{y})$$
$$\mathcal{L}\text{oss}_{\text{words}} = \frac{1}{N}\sum_{i=1}^{N} -\mathbf{x}[\mathbf{y}] + \log\left(\sum_j \exp(\mathbf{x}[j])\right)$$

Both Enc_{VAE} and Dec_{VAE} can be implemented as simple MLP since they do not have to handle variable length inputs. $\text{Dec}_{\text{words}}$ could be implemented as a Transformer Decoder, possibly pre-trained on our target language (French).

The last module of this project is called the Transition module, and is illustrated on Figure 5.4.

This transition module f would learn how to move from one job to the next. Its applications could be two-fold: firstly, it could be used as a "simple" job predictor (combined with the $\text{Dec}_{\text{words}}$ module). Secondly, it could provide higher-level insights regarding a user's expertise evolution. Indeed, at inference time, it could be ran as many times as there are jobs in a career, and provide us with a sequence of expertise levels. This sequence could be averaged and compared between industries, but it could also be used as an answer to the question: what entry-level job will give the user's career the most momentum?

5.2.2 Inference and Evaluation

Since our ultimate goal is to study the user's dynamics, the evaluation will focus on the Transition module (although intermediate models would be tested along

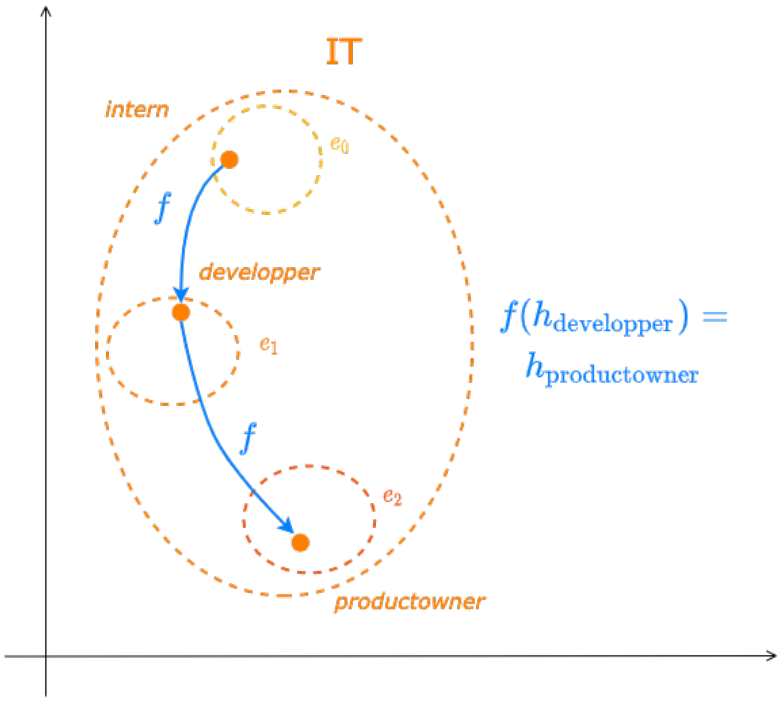

Figure 5.4 – **Illustration of the transition model**. It will take as input a point of the latent space and learn the next point, i.e., the person's next job.

the way for development purposes). The Transition model would be evaluated with an Accuracy metric in a k-nearest-neighbors setting.

5.3 Challenges & Obstacles

The architectures presented in this Chapter are subject to a number of challenges and obstacles that we develop here.

The lack of direct supervision on experience. The biggest obstacle to the previously presented architectures is that we do not have an actual supervision on the level of expertise associated with a professional experience. We tried several naive labeling approaches, namely linearly attributing a level of expertise to a person's jobs, in the spirit of J. J. McAuley and Leskovec (2013). They proved inconclusive, indicating that more complex strategies are required.

Noisy and unstructured data. As we detail in Chapter 4, the data we use (LinkedIn pages) are quite noisy, for a number of reason. At a low-level, they are

subject imprecisions such as wrongly filled job durations, misspelled words, job descriptions filled with emojis only...

From a higher level, they are also noisy because there are almost as many jobs as there are users, since no two people will describe a given professional experience in the same way.

This behavior can also be observed in the case of industrial domain: a person that has made a career as an Information Technologies) (IT) engineer for Pharmaceutical companies could either fill their industry as "IT" or "Pharmaceutical", further blurring the contours of said domains.

Aside from that, job titles and descriptions can often contain proper nouns such as product, software or company names. Furthermore, the skills presented in the users' profiles can be highly redundant (it is common to have a user fill out "Office Pack" as a skill, as well as "Microsoft Excel" or "Excel") and/or hardly discriminative (a big proportion of profiles presented the "communication" and "management" skills).

The noisiness and highly unstructured nature of our data make it quite hard to have a denoised, structured latent space to work with.

Insufficient data. Because the data at our disposal is extremely noisy, the data selection process has led to a dramatic reduction of available profiles.

In order to validate our hypotheses, we needed profiles with at least 3 professional experiences above a minimum word count, written in French, with consistent job durations (a lot of profiles had either no or erroneous job duration filled out) and corresponding to one of the 150 most common industrial domains across the dataset. While various attempts to balance the pre-requisite have been tried, it systematically lead to a drastic, possibly fatal, reduction of the dataset's size.

Are all careers comparable? Another, less technical, possible obstacle lies in the feasibility of comparing any and all careers to one-another, especially in a common latent space. Even from a naive, non-expert point of view, one can see that it is not trivial to compare the career of a researcher to the of a cuisine chef. Their studies and jobs are widely different in number, nature and timeline.

This limitation however could probably be overcome with the help of HR experts, assuming they would help refining assumptions about the common latent space.

5.4 Conclusion

In this chapter, we present two exploratory architectures for User Dynamic Modeling, an aspect that is too often overlooked in UR. Those models are parts of an ongoing work and we presented the issues and difficulties met at implementation time.

We present the underlying ideas and intuitions motivating each architecture, one based on the BT mechanism, the other on a VAE, despite not being able to exploit them in time.

We also present the obstacles faced during the conception of those architectures. The lack of supervision on experience, the highly unstructured and noisy nature of our data and the constraints imposed by data filtering proved to be great difficulties.

We hope to have detailed them enough to encourage future work in this direction.

CHAPTER 6

CONCLUSION

Contents

The work presented in this thesis bridges the domains of Recommendation, Natural Language Processing and professional profile learning by focusing on User Representation. This interest for learning rich and versatile user representations is the result of both the trends of Representation Learning and increasing personalization of systems in Deep Learning, especially in Recommendation.

This work also aims at providing explainable user representations, in the wake of the Explainable Artificial Intelligence movement. Explainability is, in our opinion, one of the biggest challenges that the Deep Learning community is now facing. Indeed, the services of computational power (CPU, GPU) rentals have exploded and it is more and more common to have companies open-source the weights of their Deep Learning models, making it increasingly easy to train, deploy and fine-tune Deep Learning models. However, the challenge lies in the resistance of users to interacting with Deep Learning models which predictions are still often too opaque to be understood.

It is easy to understand why people would not want to have to rely on black-box models in their daily lives, let alone in their professional lives. This is why making Deep Learning understandable and and their decisions explainable and fair is so crucial, even if Artificial Intelligence can only ever be used as an aid and not as a replacement: building trust between the users and the systems is the only way for these systems to be used to their full potential and really serve society beyond movie recommendation.

Fairness Learning is an entire field of Artificial Intelligence that focuses on identifying and correcting biases in Machine Learning models. The approaches to Fairness Learning vary, but the common goal of the domain is to ensure that

fair Machine Learning models's predictions are independent of sensitive and/or discriminatory variables such as gender, ethnicity etc.

It is with this idea in mind that we proposed a User Representation framework that leverages user textual traces to build their representation, and metadata of their profile to evaluate the obtained representation. Its strength resides in two main aspects. Firstly, user textual traces are not only uniquely descriptive of a person, but this kind of data is also omnipresent and easily accessible. Secondly, the intuition of the framework (using natural language to represent a user and other data to evaluate the profile) is generic enough to be applied to numerous use cases aside from professional profile learning.

In this chapter, we first summarize the contributions that we propose in this thesis before discussing interesting directions for future work.

6.1 Summary of Contributions

In this thesis, we propose an approach to User Representation through the lens of Representation Learning using Natural Language Processing in the context of Recommendation and professional profile learning. This work has shown a successful use of Natural Language Processing to refine User Representation, both as an original Representation Learning approach in a Recommendation context in Chapter 3 and as a new application domain: Human Resources, or Professional Profile Learning in Chapter 4.

In Recommendation, Collaborative Filtering and in particular Matrix Factorization yields good performances but tend to be opaque (i.e., it is hard to explain the suggestions outputted by those models).

This observation led us to propose a model that takes advantage of user review texts and uses them to provide insights regarding the suggestions outputted by the model. Leveraging user-generated text allows for a better user representation, as well as understanding what they like and why.

This approach relies on the advances of Deep Learning and Natural Language Processing. The proposed model is composed of a classical Multi-Layer Perceptron and a sentiment analysis module. Both parts share representations to learn a representation of the user's tastes that is a function of both their ratings and reviews. The sentiment analysis is a hierarchical model based on an Recurrent Neural Network coupled to a personalized attention mechanism.

We compared our architecture against several other deep architectures, and our experiments show that our model improves the accuracy of the predicted score

while also offering explanations regarding the suggestions made; thanks to the attention mechanism that provides a personalized reading model for each user.

User Representation and Representation Learning are gaining increasing focus because of their versatility and robustness. It is especially interesting to combine both because of the heterogenous nature of user profiles. Regardless of the use-case, a user profile is generally composed of several fields of different nature.

It is our intuition that amongst those data, user-generated texts are the most uniquely descriptive of a person. We thus propose a Natural Language Processing-centered framework, *Resumé*, that allows the comparison of several word embedding-based User Representation methods in a particularly interesting, yet unexplored application: professional profile representation.

Professional profile representation using online Curriculum Vitae is an ideal application because the available resources are plentiful and the profiles rich from various heterogeneous data, such as the professional experiences, the skills and the industrial domain of a user.

The *Resumé* framework leverages the profile's heterogeneous data as well as textual data in order to provide a comprehensive evaluation of the user profile. It relies on attribute classification tasks as well as text generation, thus evaluating the user representation in depth.

We compared the representations outputted by two types of embedding models (FastText, Embeddings from Language Models), as well as different input data (professional experiences and educational background, both free texts). On one hand, our experiments show that a light feed-forward model trained on the data match or outperform heavier recurrent architectures; on the other hand, they showed that the education of a user is insufficient to predict their first job or skillset, while their professional experiences are far more informative.
However those works present limitations and can be improved upon in different directions, be they either applicative or technical ones.

6.2 Perspectives for future work

Future research extending the present work can go in several directions. We distinguish between technical and applicative perspectives. The former propose technical improvements that we believe could increase current performances, while the latter present possible extensions of the present work to address new applications. Our main technical perspectives are centered around leveraging recent Natural Language Processing architectures and user dynamic modeling. We briefly

introduce two possible dynamic architectures (that are more thoroughly detailed in Chapter 5) before discussing applicative perspectives.

6.2.1 Short-term Perspectives

In this section, we propose two ways of improving the contributions presented in this thesis. Firstly, we propose to improve the existing architectures with more recent models and additional training strategies. Secondly, we propose an extension of the work done on User Representation in the form of User Dynamic Modeling, i.e., learning to represent users' evolution.

6.2.1.1 Improving our contributions

Improving User Representation in Recommendation. We propose two ways of improving the model proposed in Chapter 3.

Firstly, and since the model is already trained to jointly optimize two losses, it would require little work to introduce another loss in the architecture, a serendipity or ranking metric for instance. If it is not obvious that the performances would benefit from such change, it is still interesting to try because it could yield a better user experience both in terms of recommendation accuracy and surprise. However, such an architecture would require special attention regarding the balancing of the losses during training.

Secondly, the sentiment analysis module could be replaced by a newer, more robust architecture like a Transformer. Again, it would not be extremely complicated to replace the Recurrent Bidirectional Attentive modules with Transformers and the model could benefit from both the parallelization of the computation and the representation power of the Transformers while still sharing weights with the Recommendation module.

Improving User Representation in Professional Profile Learning. Chapter 4 presents a framework that, in its present form, is mostly used for representation evaluation purposes. However, one could consider to use an end-to-end approach in order to fine-tune (or learn from scratch) user representations. That is to say, we could train the whole framework on the tasks of the evaluation models: Skills Prediction, Industry Prediction and Next Job Generation. This Multi-Task Learning approach could be highly beneficial to the quality of the representation learned, if trained correctly [1].

1. In practice, mixtures of losses are hard hyper-parameters to tune, hence the condition of training the model correctly.

Besides, and similarly to the model of Chapter 3, the *Résumé* framework could be extended to handle other types of architectures, from Bidirectional Recurrent Neural Network to Transformers.

6.2.1.2 User Dynamics modeling

Understanding and modeling user dynamic is the key to improving any user-centric system's performances durably, be it only for churn detection.

While there is a short-term dynamic description in the generation of a user's next job, it is clearly incomplete because such predictions are only made one time step further in time and they do not allow for contextual analysis (e.g., comparison between industrial sectors, education ...).

Both those propositions have been hindered by several obstacles. On top of the User Dynamic Modeling being an ambitious task, both philosophically and scientifically, we were faced with technical difficulties. Namely, the unstructured and noisy nature of the data which, coupled to strong data filtering constraints led to a dataset of insufficient size. Additionally, the absence of job experience-level supervision (and the insufficiency of naive labelling methods) is still a major obstacle to be lifted.

In this section we briefly remind you of two architectures for professional dynamic modeling that we detailed thoroughly in Chapter 5.

Job Expertise Rewriting. Rewriting a user's job given a certain attribute value can be the cornerstone of many applications.

Attribute Rewriting with expertise level as a job attribute could highlight a user's career evolution. Specifically, it could be the cornerstone of a shift detection model, that could predict when a person is going to gain a level of expertise. Such application of Attribute Rewriting could yield interesting qualitative results comparing the dynamics of different industrial domains, but also the different types of career trajectories among users. The combination of both those studies could in turn help understand what part of a person's career evaluation pertains to the domain they work in, and what pertains to them.

Attribute rewriting relies on a Back Translation mechanism and a Denoising Auto-Encoder, combined so that the model can learn to rewrite text without collapsing. One could leverage pre-trained Transformers models to build upon Lample et al. (2018)'s work and possibly improve the quality of generated text as well as reduce training time due to the high parallelization capacity of Transformers over Recurrent Neural Networks (RNNs).

Such architecture could be applied to tasks different from career prediction, such as J. J. McAuley and Leskovec (2013)'s beer ratings (where the attribute is the level of expertise in beer) or churn detection (where the attribute would be the customer's level of engagement).

Professional Latent Space structuring via VAE. Another approach to user dynamic modeling consists in the structuring of a meaningful latent space through Variational Auto-Encoder, and then to learn a "user trajectory" model, that would learn the transition from one point in space to the next. In the spirit of Word2Vec, the idea is to build an interpretable and navigable latent space.

In the use-case of professional profile learning, said space would be divided in two levels of attributes: the industry, and the expertise level within that industry. The latent space would be structured by a Variational Auto-Encoder conditioned on both those attributes. Once the latent space has converged, one could learn a trajectory predictor that outputs a user's next point in the space. This could allow us to predict a person's next job and / or to compare average user trajectories across different industries.

This idea can also be applied to other situations. For instance, such a structured latent space could be useful in a movie Recommendation setting where the first level of structuring would be the movie genre, and the second one the year in which the movie came out (this is assuming cinematic genres evolve with time). Basically, the structuring of the latent space would allow for a fine-grained evolution prediction and comparison.

6.2.2 Long-term Perspectives

If the quality of the proposed Natural Language Processing-based user representation can be improved and broadened, so do the possible applications for such representations.

6.2.2.1 Recruitment

We see the work proposed in this thesis as a starting point for the development of Artificial Intelligence models that would aid human recruiters in their work. Indeed, it is our strong belief that recruitment can never (and should never for that matter) rely solely on computer programs. Human Resources is too much of a complex and critical application domain to be devoid of human experts. For this reason, we describe long-term applications where the extensions of our work

would only served as an aid or a tool and not as an autonomous, decision-making agent.

Possible applications of our work, especially in the Human Resources domain, lead us to question ourselves about the trends and needs to come in the industrial landscape.

Obstacles to an Artificial Intelligence-aided Human Resources. The thorniest questions are about the ethics of using Artificial Intelligence models trained on such sensitive personal data (even when pseudo-anonymized, Curriculum Vitaes (CVs) still contain personal information such as locations) in such a crucial domain (one can easily understand that the professional domain is more critical than movie recommendation).

The reasons why some people (either working in Human Resources or potentially subject to Human Resources processes) are wary of using Artificial Intelligence in the Human Resources process usually encompass the fear of discriminative decisions and the selection of profiles or CVs on insufficient criteria (and thus, the fear to select irrelevant profiles and to miss some that could have been a match).

On the other hand, it is clear that the process of pre-selecting candidates for a job offer could use some automation, as there can be a lot of applicants.Besides, as of today, a lot of companies require an applicant to give out their Curriculum Vitae but also to fill out a company-made form in order to standardize their application. This forms are expensive to develop for the company and time consuming for the applicant to fill out. A model that is trained to standardize any applicants' Curriculum Vitae into a company-wide profile would be a great gain of time.

On top of that, computer-aided recruitment processes are already used, as predictive recruitment or chatbot-assisted interviews. Besides, it is safe to imagine that LinkedIn, which is a Microsoft-owned company, is working on the valorization and leveraging of the data available on its platform. Thus, it is imperative that publicly available research be done on such topics. Otherwise, it means leaving these matters in the hands of already very powerful companies, at the risk of being eventually forced into using such Artificial Intelligence-aided processes without a way to know how they work exactly.

Future Challenges. That being said, there is still a lot of work to be done before Artificial Intelligence can make its way into any companies Human Resources departments. In our opinion, the future needs of the sector will revolve around three main axes: Understandability, Fairness, and Rewriting.

As we mentioned throughout this thesis, Understandability of models is a necessary step to build trust for job applicants as well as recruiters. It is unreasonable to think that companies would use an Artificial Intelligence model at the risk of not understanding the recruitment process. Similarly, it is safe to say that a lot of applicants would feel reluctant regarding an Artificial Intelligence-aided process, and could clearly feel cheated if not selected without being given a reason.

Additionally, Fairness is decision-making models are an absolute requirement, even if such models were only aiding the decision process alongside a human, so as to limit the possibility of discrimination.

With those three main axes in mind, we propose several possible applications of this work.

Once both those topics are satisfactorily addressed, another industrial need lies in Rewriting. We mentioned that applications can be quite numerous for a job offer, and that candidates' CVs lack standardization. Research in this direction could yield significant gain of time, as well as some more equity for applicants: automatically rewriting their CVs or application could smooth out differences in aspect (for instance, the layout of the Curriculum Vitae, clumsy writing...) and help focus on their skills and qualifications. In such context, the rewriting of CVs would propose a fairness that goes beyond the legal, mandatory aspects of recruitment. This could even be considered to be fairness applied to new criteria, further leveling the field for all applicants.

The work presented in this thesis, and especially in Chapter 4 could easily be extended into a tool for Human Resources or Employment Agency. Indeed the Job Prediction module of the *Resumé* framework is essentially a job recommender. As such, it could be used to help job seekers in their researches, by comparing their profile to available offers. In Human Resources, it could be used to pre-select candidates for a specific job offer. Another way to see it, is that *Resumé* could help standardize applicants' CVs and project them into a common latent space, together with actual employees of the company. One could then easily see what candidate would be a good match for a replacement within the company, for instance.

But the work of this thesis can also be helpful directly to the user. Identifying the profiles that correspond to the user's target dream job could yield indications so as to which skills the user should acquire, what training they should follow *etc*. Namely, it could detect trendy skills and blooming industrial sectors, in the user profiles as wall as in the job offers. Coupled with a human coach, the model could aid the job seeker to refine either a reconversion or a improvement strategy.

6.2.2.2 Other sectors

User segmentation. This work focuses on personalization relative to a single user, but it is possible to consider User Representation in a broader aspect, namely for group-representation purposes. While it is already a widespread practice to adapt one's system to a particular group of users (namely through the mean of customer segmentation) it is, to the best of our knowledge, done on static criteria such as geo-location, age and so on. We argue that the present work could be the basis for refined user segmentation, based on their interactions with the system rather than their metadata.

User compatibility. Building on traditional Recommendation methods, one can imagine a system that would match users based on compatibility. While the first idea that comes to mind is online dating, it could be even more interesting to think about it from a professional point of view. More often than not, the choice between two equally competent candidates for a job leans toward the person that would be the best match for the existing team. Knowing that, one could improve their automated candidate pre-selection by integrating a (possibly learned) similarity measure between the candidates' professional representations.

BIBLIOGRAPHY

Adikari, Shalinda and Kaushik Dutta (2014). "Identifying Fake Profiles in LinkedIn." In: *PACIS*, p. 278 (cit. on pp. 72, 73).

Adikari, Shalinda and Kaushik Dutta (2020). *Identifying Fake Profiles in LinkedIn*. arXiv: 2006.01381 [cs.SI] (cit. on p. 73).

Almahairi, Amjad, Kyle Kastner, Kyunghyun Cho, and Aaron Courville (2015). "Learning distributed representations from reviews for collaborative filtering". In: *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, pp. 147–154 (cit. on pp. 21, 57).

Anelli, Vito Walter, Alejandro Bellog'in, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco Maria Donini, and T. D. Noia (2021). "Elliot: A Comprehensive and Rigorous Framework for Reproducible Recommender Systems Evaluation". In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (cit. on p. 24).

Bahdanau, Dzmitry, Kyung Hyun Cho, and Yoshua Bengio (2015). "Neural machine translation by jointly learning to align and translate". English (US). In: 3rd International Conference on Learning Representations, ICLR 2015 ; Conference date: 07-05-2015 Through 09-05-2015 (cit. on pp. 33, 36).

Bellogín, Alejandro and Alan Said (2018). "Recommender Systems Evaluation". In: *Encyclopedia of Social Network Analysis and Mining*. 2nd ed. Springer-Verlag New York (cit. on p. 21).

Bennett, James, Stan Lanning, et al. (2007). "The netflix prize". In: *Proceedings of KDD cup and workshop*. Vol. 2007. New York, NY, USA, p. 35 (cit. on pp. 17, 56).

Berger, Patrik and Michal Kompan (2019). "User Modeling for Churn Prediction in E-Commerce". In: *IEEE Intelligent Systems* 34.2, pp. 44–52 (cit. on p. 11).

Blei, David M., Andrew Y. Ng, and Michael I. Jordan (2003). "Latent dirichlet allocation". In: *J. Mach. Learn. Res.* 3, pp. 993–1022. URL: http://portal.acm.org/citation.cfm?id=944937 (cit. on pp. 18, 21, 74).

Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov (2016). "Enriching Word Vectors with Subword Information". In: *Transactions of the Association for Computational Linguistics* 5 (cit. on pp. 30, 33, 64, 74, 76).

Callison-Burch, Chris, Miles Osborne, and Philipp Koehn (2006). "Re-evaluating the Role of Bleu in Machine Translation Research". In: *11th Conference of the European Chapter of the Association for Computational Linguistics*. Trento, Italy:

Association for Computational Linguistics. URL: https://www.aclweb.org/anthology/E06-1032 (cit. on p. 50).

Castells, Pablo, Neil J. Hurley, and Saul Vargas (2015). "Novelty and Diversity in Recommender Systems". In: *Recommender Systems Handbook*. Boston, MA: Springer US, pp. 881–918. URL: https://doi.org/10.1007/978-1-4899-7637-6_26 (cit. on p. 24).

Catherine, Rose and William Cohen (2017). "TransNets: Learning to Transform for Recommendation". In: *Proceedings of the Eleventh ACM Conference on Recommender Systems*. RecSys '17. Como, Italy: ACM, pp. 288–296. URL: http://doi.acm.org/10.1145/3109859.3109878 (cit. on pp. 21, 66).

Chen, Huimin, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu (2016). "Neural sentiment classification with user and product attention". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1650–1659 (cit. on pp. 64, 65).

Cho, KyungHyun, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio (2014). "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches". In: *CoRR* abs/1409.1259. arXiv: 1409.1259. URL: http://arxiv.org/abs/1409.1259 (cit. on pp. 31, 32).

Chung, Junyoung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio (2014). "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: *CoRR* abs/1412.3555. arXiv: 1412.3555. URL: http://arxiv.org/abs/1412.3555 (cit. on p. 32).

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *NAACL* (cit. on p. 40).

Dias, Charles-Emmanuel, Clara Gainon de Forsan de Gabriac, Vincent Guigue, and Patrick Gallinari (2018). "RNN & modèle d'attention pour l'apprentissage de profils textuels personnalisés". In: *CORIA 2018 - 15ème COnférence en Recherche d'Informations et Applications*. Rennes, France. URL: https://hal.archives-ouvertes.fr/hal-02503473 (cit. on p. 55).

Dias, Charles-Emmanuel, Vincent Guigue, and Patrick Gallinari (2016). "Recommandation et analyse de sentiments dans un espace latent textuel". In: *CORIA* (cit. on p. 66).

Dias, Charles-Emmanuel, Vincent Guigue, and Patrick Gallinari (2017a). "Passé, présent, futurs : induction de carrières professionnelles à partir de CV". In: *COnférence en Recherche d'Informations et Applications - CORIA 2017, 14th French Information Retrieval Conference*. Marseille, France, pp. 281–296. URL: https://hal.archives-ouvertes.fr/hal-01549568 (cit. on p. 31).

Dias, Charles-Emmanuel, Vincent Guigue, and Patrick Gallinari (2017b). "Text-based collaborative filtering for cold-start soothing and recommendation en-

richment". In: *AISR2017*. Paris, France. URL: https://hal.archives-ouvertes.fr/hal-01640268 (cit. on pp. 20, 57).

Dong, Li, Shaohan Huang, Furu Wei, Mirella Lapata, Ming Zhou, and Ke Xu (2017). "Learning to Generate Product Reviews from Attributes". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, pp. 623–632. URL: https://www.aclweb.org/anthology/E17-1059 (cit. on p. 24).

Gage, Philip (1994). "A new algorithm for data compression". In: *The C Users Journal archive* 12, pp. 23–38 (cit. on p. 30).

Gainon de Forsan de Gabriac, Clara, Vincent Guigue, and Patrick Gallinari (2020). "Resume: A Robust Framework for Professional Profile Learning & Evaluation". In: *ESANN 2020 - 28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. Bruges, Belgium. URL: https://hal.archives-ouvertes.fr/hal-02503464 (cit. on p. 71).

Ganu, Gayatree, Noémie Elhadad, and A. Marian (2009). "Beyond the Stars: Improving Rating Predictions using Review Text Content". In: *WebDB* (cit. on p. 20).

Ganu, Gayatree, Yogesh Kakodkar, and Amélie Marian (2013). "Improving the quality of predictions using textual information in online user reviews". In: *Information Systems* 38.1, pp. 1–15. URL: https://www.sciencedirect.com/science/article/pii/S0306437912000476 (cit. on p. 20).

Ge, Mouzhi, Carla Delgado-Battenfeld, and Dietmar Jannach (2010). "Beyond Accuracy: Evaluating Recommender Systems by Coverage and Serendipity". In: *Proceedings of the Fourth ACM Conference on Recommender Systems*. RecSys '10. Barcelona, Spain: Association for Computing Machinery, pp. 257–260. URL: https://doi.org/10.1145/1864708.1864761 (cit. on p. 24).

Gers, Felix A., Jürgen A. Schmidhuber, and Fred A. Cummins (2000). "Learning to Forget: Continual Prediction with LSTM". In: *Neural Comput.* 12.10, pp. 2451–2471. URL: https://doi.org/10.1162/089976600300015015 (cit. on p. 31).

Goldberg, Yoav and Omer Levy (2014). "word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method". In: *CoRR* abs/1402.3722. arXiv: 1402.3722. URL: http://arxiv.org/abs/1402.3722 (cit. on p. 28).

Gomez-Uribe, Carlos A. and Neil Hunt (2016). "The Netflix Recommender System: Algorithms, Business Value, and Innovation". In: *ACM Trans. Manage. Inf. Syst.* 6.4. URL: https://doi.org/10.1145/2843948 (cit. on p. 18).

Graves, Alex, Navdeep Jaitly, and Abdel-rahman Mohamed (2013). "Hybrid speech recognition with Deep Bidirectional LSTM". In: *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 273–278 (cit. on p. 33).

Greff, Klaus, Rupesh K. Srivastava, Jan Koutnik, Bas R. Steunebrink, and Jurgen Schmidhuber (2017). "LSTM: A Search Space Odyssey". In: *IEEE Transactions*

on Neural Networks and Learning Systems 28.10, pp. 2222–2232. URL: http://dx.doi.org/10.1109/TNNLS.2016.2582924 (cit. on p. 32).

Guy, Ido (2015). "Social recommender systems". In: *Recommender Systems Handbook*. Springer, pp. 511–543 (cit. on p. 56).

Hallac, I. R., S. Makinist, B. Ay, and G. Aydin (2019). "user2Vec: Social Media User Representation Based on Distributed Document Embeddings". In: *2019 International Artificial Intelligence and Data Processing Symposium (IDAP)*, pp. 1–5 (cit. on p. 11).

Hidasi, Balázs, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk (2016). "Session-based Recommendations with Recurrent Neural Networks". In: *CoRR* abs/1511.06939 (cit. on p. 35).

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-Term Memory". In: *Neural Computation* 9.8, pp. 1735–1780 (cit. on p. 31).

Hu, Guang-Neng and Xinyu Dai (2017). "Integrating Reviews into Personalized Ranking for Cold Start Recommendation". In: *PAKDD (2)*, pp. 708–720. URL: https://doi.org/10.1007/978-3-319-57529-2_55 (cit. on p. 20).

Jain, Gourav, Tripti Mahara, and Kuldeep Narayan Tripathi (2020). "A Survey of Similarity Measures for Collaborative Filtering-Based Recommender System". In: *Soft Computing: Theories and Applications*. Ed. by Millie Pant, Tarun K. Sharma, Om Prakash Verma, Rajesh Singla, and Afzal Sikander. Singapore: Springer Singapore, pp. 343–352 (cit. on p. 16).

Jain, Pallavi, Robert Ross, and Bianca Schoen-Phelan (2019). "Estimating Distributed Representation Performance in Disaster-Related Social Media Classification". In: *ASONAM* (cit. on p. 85).

Jiang, Ray, Silvia Chiappa, Tor Lattimore, András György, and Pushmeet Kohli (2019). "Degenerate Feedback Loops in Recommender Systems". In: *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*. AIES '19. Honolulu, HI, USA: Association for Computing Machinery, pp. 383–390. URL: https://doi.org/10.1145/3306618.3314288 (cit. on p. 14).

Karimzadehgan, Maryam, ChengXiang Zhai, and Geneva Belford (2008). "Multi-Aspect Expertise Matching for Review Assignment". In: *Proceedings of the 17th ACM Conference on Information and Knowledge Management*. CIKM '08. Napa Valley, California, USA: Association for Computing Machinery, pp. 1113–1122. URL: https://doi.org/10.1145/1458082.1458230 (cit. on p. 72).

Kenthapadi, Krishnaram, Benjamin Le, and Ganesh Venkataraman (2017). "Personalized Job Recommendation System at LinkedIn: Practical Challenges and Lessons Learned". In: *Proceedings of the Eleventh ACM Conference on Recommender Systems* (cit. on pp. 73, 74).

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: ed. by Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1412.6980 (cit. on p. 63).

Kingma, Diederik P. and Max Welling (2014). "Auto-Encoding Variational Bayes". In: *CoRR* abs/1312.6114 (cit. on p. 44).

Koren, Yehuda (2010). "Collaborative filtering with temporal dynamics". In: *Communications of the ACM* 53.4, pp. 89–97 (cit. on p. 56).

Koren, Yehuda, Robert Bell, and Chris Volinsky (2009). "Matrix factorization techniques for recommender systems". In: *Computer* 42.8 (cit. on p. 56).

Kwon, Young D., Dimitris Chatzopoulos, Ehsan ul Haq, Raymond Chi-Wing Wong, and Pan Hui (2019). "GeoLifecycle: User Engagement of Geographical Exploration and Churn Prediction in LBSNs". In: *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3.3. URL: https://doi.org/10.1145/3351250 (cit. on p. 11).

Lample, Guillaume, Ludovic Denoyer, and Marc'Aurelio Ranzato (2017). "Unsupervised Machine Translation Using Monolingual Corpora Only". In: *CoRR* abs/1711.00043. arXiv: 1711.00043. URL: http://arxiv.org/abs/1711.00043 (cit. on pp. 46, 47, 49).

Lample, Guillaume, Sandeep Subramanian, Eric Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau (2018). "Multiple-attribute text rewriting". In: *International Conference on Learning Representations* (cit. on pp. 91, 103).

Le, Quoc V., Navdeep Jaitly, and Geoffrey E. Hinton (2015). "A Simple Way to Initialize Recurrent Networks of Rectified Linear Units". In: *CoRR* abs/1504.00941. arXiv: 1504.00941. URL: http://arxiv.org/abs/1504.00941 (cit. on p. 31).

Lewis, Mike, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer (2020). "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension". In: pp. 7871–7880. URL: https://aclanthology.org/2020.acl-main.703 (cit. on p. 48).

Li, Lei, Yongfeng Zhang, and Li Chen (2021). "Personalized Transformer for Explainable Recommendation". In: (cit. on p. 41).

Li, Liangyue, How Jing, Hanghang Tong, Jaewon Yang, Qi He, and Bee-Chung Chen (2017). "NEMO: Next Career Move Prediction with Contextual Embedding". In: *Proceedings of the 26th International Conference on World Wide Web Companion* (cit. on pp. 73, 74).

Li, Ming, Ying Li, Wangqin Lou, and Lisheng Chen (2020). "A hybrid recommendation system for Q&A documents". In: *Expert Systems with Applications* 144, p. 113088. URL: https://www.sciencedirect.com/science/article/pii/S095741741930805X (cit. on p. 18).

Li, Yang and Tao Yang (2017). "Word Embedding for Understanding Natural Language: A Survey". In: vol. 26. Springer, Cham. Chap. 1, pp. 83–104 (cit. on p. 27).

Lin, Chin-Yew (2004). "ROUGE: A Package for Automatic Evaluation of Summaries". In: *Text Summarization Branches Out*. Barcelona, Spain: Association

for Computational Linguistics, pp. 74–81. URL: https://www.aclweb.org/anthology/W04-1013 (cit. on p. 51).

Ling, Guang, Michael R Lyu, and Irwin King (2014). "Ratings meet reviews, a combined approach to recommend". In: *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, pp. 105–112 (cit. on p. 57).

Liu, Tie-Yan (2009). "Learning to Rank for Information Retrieval". In: *Foundations and Trends® in Information Retrieval* 3.3, pp. 225–331. URL: http://dx.doi.org/10.1561/1500000016 (cit. on p. 24).

Liu, Yinhan, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer (2020). "Multilingual Denoising Pre-training for Neural Machine Translation". In: *Transactions of the Association for Computational Linguistics* 8, pp. 726–742 (cit. on p. 49).

Lops, Pasquale, Marco de Gemmis, and Giovanni Semeraro (2011). "Content-based Recommender Systems: State of the Art and Trends". In: Springer, Boston, MA, pp. 73–105 (cit. on p. 18).

Lundberg, Scott M. and Su-In Lee (2017). "A Unified Approach to Interpreting Model Predictions". In: *NIPS* (cit. on p. 7).

Mani, I. and M. Maybury (1999). "Advances in Automatic Text Summarization". In: The MIT Press, pp. 123–136 (cit. on p. 74).

McAuley, Julian John and Jure Leskovec (2013). "From Amateurs to Connoisseurs: Modeling the Evolution of User Expertise through Online Reviews". In: WWW '13, pp. 897–908. URL: https://doi.org/10.1145/2488388.2488466 (cit. on pp. 13, 56, 57, 90, 96, 104).

McAuley, Julian and Jure Leskovec (2013). "Hidden factors and hidden topics: understanding rating dimensions with review text". In: *Proceedings of the 7th ACM conference on Recommender systems*. ACM, pp. 165–172 (cit. on pp. 20, 21, 56, 66).

McAuley, Julian, Rahul Pandey, and Jure Leskovec (2015a). "Inferring networks of substitutable and complementary products". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pp. 785–794 (cit. on p. 63).

McAuley, Julian, Christopher Targett, Qinfeng Shi, and Anton van den Hengel (2015b). "Image-Based Recommendations on Styles and Substitutes". In: SIGIR '15, pp. 43–52. URL: https://doi.org/10.1145/2766462.2767755 (cit. on p. 21).

Mikolov, Tomas, Kai Chen, Gregory S. Corrado, and Jeffrey Dean (2013a). "Efficient Estimation of Word Representations in Vector Space". In: *ICLR* (cit. on p. 40).

Mikolov, Tomas, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur (2010). "Recurrent neural network based language model." In: *INTERSPEECH*. Ed. by Takao Kobayashi, Keikichi Hirose, and Satoshi Naka-

mura. ISCA, pp. 1045–1048. URL: http://dblp.uni-trier.de/db/conf/interspeech/interspeech2010.html#MikolovKBCK10 (cit. on p. 27).

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean (2013b). "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems*. Ed. by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger. Vol. 26. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf (cit. on pp. 27–29, 33, 74).

Ni, Jianmo, Zachary C. Lipton, Sharad Vikram, and Julian McAuley (2017). "Estimating Reactions and Recommending Products with Generative Models of Reviews". In: *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Taipei, Taiwan: Asian Federation of Natural Language Processing, pp. 783–791. URL: https://www.aclweb.org/anthology/I17-1079 (cit. on p. 24).

Noordeh, Emil, Roman Levin, Ruochen Jiang, and Harris Shadmany (2020). "Echo Chambers in Collaborative Filtering Based Recommendation Systems". In: *CoRR* abs/2011.03890. arXiv: 2011.03890. URL: https://arxiv.org/abs/2011.03890 (cit. on p. 14).

Pang, Bo, Lillian Lee, et al. (2008). "Opinion mining and sentiment analysis". In: *Foundations and Trends® in Information Retrieval* 2.1–2, pp. 1–135 (cit. on p. 65).

Paparrizos, Ioannis K., Berkant Barla Cambazoglu, and Aristides Gionis (2011). "Machine learned job recommendation". In: *RecSys '11* (cit. on pp. 73, 74).

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu (2002a). "BLEU: A Method for Automatic Evaluation of Machine Translation". In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. ACL '02. Philadelphia, Pennsylvania: Association for Computational Linguistics, pp. 311–318. URL: https://doi.org/10.3115/1073083.1073135 (cit. on pp. 75, 84).

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu (2002b). "Bleu: a Method for Automatic Evaluation of Machine Translation". In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, pp. 311–318. URL: https://www.aclweb.org/anthology/P02-1040 (cit. on pp. 49, 50).

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *CoRR* abs/1912.01703. arXiv: 1912.01703. URL: http://arxiv.org/abs/1912.01703 (cit. on p. 82).

Paulus, R., C. Xiong, and R. Socher (2017). "A Deep Reinforced Model for Abstractive Summarization". In: *CoRR* 1705.04304. arXiv: 1705.04304. URL: http://arxiv.org/abs/1705.04304 (cit. on p. 74).

Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer (2018). "Deep contextualized word representations". In: *Proc. of NAACL* (cit. on pp. 33, 35, 74, 76).

Pudipeddi, Jagat Sastry, Leman Akoglu, and Hanghang Tong (2014). "User Churn in Focused Question Answering Sites: Characterizations and Prediction". In: *Proceedings of the 23rd International Conference on World Wide Web*. WWW '14 Companion. Seoul, Korea: Association for Computing Machinery, pp. 469–474. URL: https://doi.org/10.1145/2567948.2576965 (cit. on p. 11).

Radford, Alec, Rafal Józefowicz, and Ilya Sutskever (2017). "Learning to Generate Reviews and Discovering Sentiment". In: *CoRR* abs/1704.01444. arXiv: 1704.01444. URL: http://arxiv.org/abs/1704.01444 (cit. on p. 24).

Radford, Alec and Karthik Narasimhan (2018). "Improving Language Understanding by Generative Pre-Training". In: *arxiv* (cit. on pp. 47, 48).

Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu (2020). "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *Journal of Machine Learning Research* 21.140, pp. 1–67. URL: http://jmlr.org/papers/v21/20-074.html (cit. on p. 49).

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin (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* (cit. on p. 7).

Sánchez, Pablo and Alejandro Bellogín (2018a). "Measuring Anti-Relevance: A Study on When Recommendation Algorithms Produce Bad Suggestions". In: *Proceedings of the 12th ACM Conference on Recommender Systems*. RecSys '18. Vancouver, British Columbia, Canada: Association for Computing Machinery, pp. 367–371. URL: https://doi.org/10.1145/3240323.3240382 (cit. on p. 24).

Sánchez, Pablo and Alejandro Bellogín (2018b). "Time-Aware Novelty Metrics for Recommender Systems". In: *ECIR* (cit. on p. 56).

Schuster, M. and K.K. Paliwal (1997). "Bidirectional recurrent neural networks". In: *IEEE Transactions on Signal Processing* 45.11, pp. 2673–2681 (cit. on p. 32).

Sennrich, Rico, Barry Haddow, and Alexandra Birch (2015). "Improving Neural Machine Translation Models with Monolingual Data". In: *CoRR* abs/1511.06709. arXiv: 1511.06709. URL: http://arxiv.org/abs/1511.06709 (cit. on p. 46).

Silveira, Thiago, Min Zhang, X. Lin, Yiqun Liu, and S. Ma (2019). "How good your recommender system is? A survey on evaluations in recommendation". In:

International Journal of Machine Learning and Cybernetics 10, pp. 813–831 (cit. on p. 24).

Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis (2016). "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529.7587, pp. 484–489. URL: https://doi.org/10.1038/nature16961 (cit. on p. 1).

Smirnova, Elena and Flavian Vasile (2017). "Contextual Sequence Modeling for Recommendation with Recurrent Neural Networks". In: *Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems*. DLRS 2017. Como, Italy: Association for Computing Machinery, pp. 2–9. URL: https://doi.org/10.1145/3125486.3125488 (cit. on p. 35).

Sparck Jones, Karen (1988). "A Statistical Interpretation of Term Specificity and Its Application in Retrieval". In: *Document Retrieval Systems*. GBR: Taylor Graham Publishing, pp. 132–142 (cit. on p. 18).

Steck, Harald (2010). "Training and Testing of Recommender Systems on Data Missing Not at Random". In: *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '10. Washington, DC, USA: Association for Computing Machinery, pp. 713–722. URL: https://doi.org/10.1145/1835804.1835895 (cit. on p. 23).

Sukhbaatar, Sainbayar, Arthur Szlam, Jason Weston, and Rob Fergus (2015). "Weakly Supervised Memory Networks". In: *CoRR* abs/1503.08895. arXiv: 1503.08895. URL: http://arxiv.org/abs/1503.08895 (cit. on p. 37).

Sun, Fei, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang (2019). "BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer". In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. CIKM '19. Beijing, China: Association for Computing Machinery, pp. 1441–1450. URL: https://doi.org/10.1145/3357384.3357895 (cit. on p. 41).

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le (2014). "Sequence to Sequence Learning with Neural Networks". In: *NIPS* (cit. on pp. 45, 46).

Tang, W., J. Tang, and C. Tan (2010). "Expertise Matching via Constraint-Based Optimization". In: *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*. Vol. 1, pp. 34–41 (cit. on p. 72).

Tang, Wenbin, Jie Tang, Tao Lei, Chenhao Tan, Bo Gao, and Tian Li (2012). "On optimization of expertise matching with various constraints". In: *Neurocomputing* 76.1. Seventh International Symposium on Neural Networks (ISNN 2010) Advances in Web Intelligence, pp. 71–83. URL: http://www.sciencedirect.com/science/article/pii/S0925231211004097 (cit. on p. 72).

Thorat, Poonam B., R. Goudar, and S. Barve (2015). "Survey on Collaborative Filtering, Content-based Filtering and Hybrid Recommendation System". In: *International Journal of Computer Applications* 110, pp. 31–36 (cit. on p. 18).

Tintarev, Nava and Judith Masthoff (2007). "A survey of explanations in recommender systems". In: *Data Engineering Workshop, 2007 IEEE 23rd International Conference on*. IEEE, pp. 801–810 (cit. on p. 56).

Tu, Songgao and Chaojun Lu (2010). "Topic-Based User Segmentation for Online Advertising with Latent Dirichlet Allocation". In: *Advanced Data Mining and Applications*. Ed. by Longbing Cao, Jiang Zhong, and Yong Feng. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 259–269 (cit. on p. 11).

Valcarce, Daniel, Alejandro Bellogín, Javier Parapar, and Pablo Castells (2020). "Assessing ranking metrics in top-N recommendation". In: *Information Retrieval Journal* 23.4, pp. 411–448. URL: https://doi.org/10.1007/s10791-020-09377-x (cit. on p. 24).

Vargas, Saúl and Pablo Castells (2011). "Rank and Relevance in Novelty and Diversity Metrics for Recommender Systems". In: *Proceedings of the Fifth ACM Conference on Recommender Systems*. RecSys '11. Chicago, Illinois, USA: Association for Computing Machinery, pp. 109–116. URL: https://doi.org/10.1145/2043932.2043955 (cit. on p. 24).

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf (cit. on pp. 36–39).

Vinyals, Oriol, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver (2019). "Grandmaster level in StarCraft II using multi-agent reinforcement learning". In: *Nature* 575.7782, pp. 350–354. URL: https://doi.org/10.1038/s41586-019-1724-z (cit. on p. 1).

Wang, Weiqing, Hongzhi Yin, Xingzhong Du, Wen Hua, Yongjun Li, and Quoc Viet Hung Nguyen (2019). "Online User Representation Learning Across Heterogeneous Social Networks". In: *Proceedings of the 42nd International ACM*

SIGIR Conference on Research and Development in Information Retrieval. SIGIR'19. Paris, France: Association for Computing Machinery, pp. 545–554. URL: https://doi.org/10.1145/3331184.3331258 (cit. on p. 11).

Yan, Jun, Ning Liu, Gang Wang, Wen Zhang, Yun Jiang, and Zheng Chen (2009). "How Much Can Behavioral Targeting Help Online Advertising?" In: *Proceedings of the 18th International Conference on World Wide Web*. WWW '09. Madrid, Spain: Association for Computing Machinery, pp. 261–270. URL: https://doi.org/10.1145/1526709.1526745 (cit. on p. 11).

Yang, Zichao, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy (2016). "Hierarchical attention networks for document classification". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1480–1489 (cit. on pp. 57, 58, 64, 65).

Ziarani, Reza Jafari and Reza Ravanmehr (2021). "Serendipity in Recommender Systems: A Systematic Literature Review". In: *Journal of Computer Science and Technology* 36.2, pp. 375–396. URL: https://doi.org/10.1007/s11390-020-0135-9 (cit. on p. 24).