Recommendation System for the News Market

Miguel Ângelo Pontes Rebelo
Mestrado em Estatística Computacional e Análise de dados
Departamento de Matemática
2022

Orientador
Prof. Dr. João Vinagre, Professor Auxiliar convidado, Faculdade de Ciências

Co-orientador
Prof. Dr. Álvaro Figueira, Professor Auxiliar, Faculdade de Ciências

Co-orientador
Prof. Dr. Ivo Pereira, Coordenador de Estágios, E-goi

U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas
pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, ______/______/________

MSC

UNIVERSIDADE DO PORTO

MASTERS THESIS

Recommendation System for the News Market

Author:
Miguel Ângelo Pontes Rebelo

Supervisor:
Prof. Doutor João Vinagre

Co-supervisor:
Prof. Doutor Álvaro Figueira

Co-supervisor:
Prof. Doutor Ivo Pereira

A thesis submitted in fulfilment of the requirements
for the degree of Master of Science in Computational Statistics and Data Analysis

at the

Faculdade de Ciências da Universidade do Porto
Departamento de Matemática

December 19, 2022

Sworn Statement

I, Miguel Ângelo Pontes Rebelo, enrolled in the Master Degree Computational Statistics and Data Analysis at the Faculty of Sciences of the University of Porto hereby declare, in accordance with the provisions of paragraph a) of Article 14 of the Code of Ethical Conduct of the University of Porto, that the content of this project reflects perspectives, research work and my own interpretations at the time of its submission.

By submitting this project, I also declare that it contains the results of my own research work and contributions that have not been previously submitted to this or any other institution.

I further declare that all references to other authors fully comply with the rules of attribution and are referenced in the text by citation and identified in the bibliographic references section. This project does not include any content whose reproduction is protected by copyright laws.

I am aware that the practice of plagiarism and self-plagiarism constitute a form of academic offense.

Miguel Ângelo Pontes Rebelo

December 19, 2022

“Amedrontados pelo Presente funesto e ruidoso,
Despertámos ansiosos de um Sul arcano
Uma quente era desnuda de instintiva pose,
Sabor vivaz numa boca inocente.

Vinda a noite, em casa, sonhámos dançar
Nos salões do futuro: cada ritual labiríntico,
Um plano musical, que um coração musical
Segue na perfeição os seus perfeitos rumos.

Que inveja dos ribeiros e das casas, falíveis
Sem dúvida, mas nunca fomos
Desnudos e mansos como uma grande porta,

E jamais perfeitos como as fontes:
Vivemos, por necessidade, livres,
Povo montês que pelos montes erra.”

W. H. Auden Sonetos da China, XVIII

Agradecimentos

Quase em choro e de fadiga a face extinta, reconheço todos os que, de uma forma ou outra, estiveram a meu lado durante este ano.

Agradeço à Faculdade de Ciências da Universidade do Porto, nobre instituição, pela oportunidade concedida de desenvolver este trabalho.

Ao Doutor João Vinagre, meu Orientador, por toda a sabedoria que me emprestou, de qualidade científica inquestionável. Foi um contacto permanente e constante, corrigindo-me com o seu rigor e dando a sua opinião ao longo do caminho, forjando metas exigentes e traçando possíveis cenários, rumo à excelência. Não sei se cumpri com todas as expectativas que eventualmente terá posto sobre mim, mas espero ter estado à altura.

Agradeço ao meu Co-orientador, o Doutor Álvaro Figueira, a fé que depositou em mim, sem de lado algum me conhecer, tendo sido um grande impulsionador, acima de tudo, desta parceria. Desde início mostrou entusiasmo pelo projecto, tendo também entusiasmado, e dando-me a conhecer o que viria a ser o meu Orientador.

Manifesto a minha gratidão a todos os membros da equipa de Inteligência Artificial da E-goi, que acreditaram em mim e nas virtudes deste projecto desde o momento zero, não tendo posto quaisquer entraves, ao invés, incentivando e motivando.

Ao Doutor Ivo Pereira, pedra angular para que isto fosse possível, nunca mostrou reservas e sempre se pontificou de imediato para ajudar, mediar ou dar a sua opinião.

Destaco, em particular, o director do departamento de IID, Daniel Alves Oliveira, que me mostrou uma forma diferente de liderar, com o seu carácter distinto, e que nunca viu silvas ou pedras que não pudessem ser cortadas ou removidas.

Recordo a Doutora Nádia Pinto, que muito embora não tenha tido um envolvimento directo na elaboração deste projecto, foi pólvora e rastilho para que tudo isto se desenrolasse, quais sinuosos caminhos.

Às pessoas que me são mais próximas, elas sabem quem são, por todo o auxílio moral que me emprestaram. Em particular à minha família, aprendi com eles o carácter e a rectidão, a generosidade e a independência de espírito, a integridade e o optimismo na adversidade, a aguentar a fadiga e a não fazer exigências, a trabalhar por mim mesmo, sem me sobrecarregar, e a não desperdiçar o meu tempo em futilidades ou frivolidades.

UNIVERSIDADE DO PORTO

Abstract

Faculdade de Ciências da Universidade do Porto

Departamento de Matemática

Master of Science in Computational Statistics and Data Analysis

Recommendation System for the News Market

by Miguel Ângelo Pontes Rebelo

One of the biggest challenges that news recommender systems face is the articles' short life cycle. Since the relevance of the items tends to be proportional to their novelty, decaying rapidly, recommenders have to permanently deal with the item *cold-start* problem, a well-known issue in the field of recommender systems. Fortunately, recent contributions proposing content-based neural approaches to news recommendation have shown to have a great potential at addressing that problem. However, these contributions consist of highly centralized models involving complex neural architectures. We argue that content-based methods can easily be used in a highly decentralized environment, and thus we propose a decentralized content-based news recommender. In this proposal, we train a separate neural network for each user, able to provide personalized relevance scores for brand new items, based on their text content. The input layer consists of item embeddings learned directly from the news titles and other variables. These personal neural networks are lightweight and can be trained within the user realm, provided that consistent item representations are available. Experiments with the MIND dataset show that the accuracy of our method can rival State-of-the-Art centralized models. Another important finding is that the negative item sampling technique in this dataset can be crucial for the accuracy of the model. Finally, we discuss how decentralized models can help improve privacy and scalability and enhance user sovereignty over data, algorithms, and models.

UNIVERSIDADE DO PORTO

Resumo

Faculdade de Ciências da Universidade do Porto

Departamento de Matemática

Mestrado em Estatística Computacional e Análise de Dados

Sistema de Recomendação para o mercado de notícias

por Miguel Ângelo Pontes Rebelo

Um dos maiores desafios que os sistemas de recomendação de notícias enfrentam é o ciclo de vida curto dos artigos. Como a relevância dos items tende a ser proporcional à sua novidade, decaindo rapidamente, este tipo de sistemas tem que lidar permanentemente com o problema *'cold-start'*. Felizmente, contribuições recentes mostraram que abordagens neuronais baseadas na análise de conteúdo têm um grande potencial. No entanto, essas contribuições consistem em modelos altamente centralizados envolvendo arquiteturas neuronais complexas. Defendemos que as abordagens baseadas na análise do conteúdo podem ser facilmente usadas em ambientes altamente descentralizados e, portanto, propõe-se aqui um sistema de recomendação de notícias descentralizado baseado na análise e modelação do conteúdo. Aqui, propomos uma rede neural separada para cada leitor, capaz de fornecer pontuações de relevância personalizadas para novos items, com base no seu conteúdo textual. A camada de entrada consiste em *embeddings* de items aprendidas directamente a partir dos títulos das notícias e outras variáveis. Essas redes neuronais pessoais são leves e podem ser treinadas dentro do domínio do utilizador, desde que as representações compactas estejam disponíveis para os items em causa. Experiências com o conjunto de dados MIND mostram que a precisão do nosso método pode rivalizar os modelos centralizados de última geração. Outro ponto importante é que a técnica de amostragem negativa de items, neste conjunto de dados, pode ser crucial para a precisão do modelo. Por fim, discutimos como os modelos descentralizados podem ajudar a melhorar a privacidade e escalabilidade, assim como contribuir para a soberania do usuário sobre dados, algoritmos e modelos.

Contents

List of Figures

Glossary

AI	Artificial Intelligence
API	Application Programming Interface
AUC	Area Under the (Receiver Operating) Curve
BERT	Bidirectional Encoder Representations from Transformers
CF	Collaborative Filtering
CNN	Convolutional Neural Networks
CPU	Central Processing Unit
DKN	Deep Knowledge-aware Network
DNNR	Decentralized Neural News Recommendation system
FFNN	Feedforward Neural Network
GPU	Graphics Processing Unit
MEC	Mobile Edge Computing
MIND	Microsoft News Dataset
MLP	Multi-Layer Perceptron
ML	Machine Learning
MSE	Mean Squared Error
NLP	Natural Language Processing
NPA	News Recommender with Personalized Attention
NRMS	New Recommender with Multi-head Self-attention
NRS	News Recommender Systems
OBSM	Ontology Based Similarity Model

OSI	Open Source Initiative
PCA	Principal Component Analysis
PLM	Pretrained Language Model
ReLU	Rectified Linear Unit
RS	Recommender System(s)
SaaS	Software as a Service
SoA	State-of-the-Art
Tanh	Hyperbolic Tangent
UI	User Interface

Chapter 1

Introduction

1.1 Background and motivation

The news media landscape displays certain properties related to their business model that are not often, if at all, observed in other domain areas. One key difference is that the speed at which the relevance of the items decay is much higher than, say, in music, movies, or the retail market. The relevance of news articles can change very rapidly concomitant with daily happenings and events. News recommendation systems need to address this permanent item *cold-start* problem, since they cannot uniquely exploit correlations between past interactions. Fortunately, news articles are rich in content that can be extracted and analysed.

Novel content-based neural approaches from recent contributions have shown great potential at dealing with these characteristics. The problem is that these bodies of work consist of highly centralized monolithic models involving very complex neural architectures. They run in a *black-box* fashion that hinders adaptation, development, explainability and maintenance.

The lack of more data-centric approaches to these content-based methods inflamed the souls and captivated the minds to think that this gap could be filled with a less complex model while maintaining a competitive accuracy relatively to State-of-the-Art (SoA) approaches. Indeed, the way negative feedback is sampled is, most of the time, an avoided or ignored topic across most of the papers, with only brief mentions and vague allusions. Then, the question that naturally follows this becomes: if this sampling issue could be improved, how would this impact the final model and how would this perform?

1.2 Objectives

The main goal of this work is to study, produce and propose a general training framework for an accurate and scalable news recommendation system, that is also easier to implement, maintain, adapt, explain and decentralize than comparable SoA approaches. To achieve this, its individual components should be addressed, namely:

- the neural network design, that is intended to be small and lightweight, easy to implement and explain;
- the creation of compact representations from news content, to then use it as variables in the network;
- a negative sampling technique, to try and capture reliable negative feedback data inferred from the readers' tendencies and feed it into the network to train the preferences.

After this, the method is thoroughly assessed to check its performance gains or bottlenecks. This includes:

- choosing the optimal number of samples per user;
- testing the impact of the negative sampling technique compared to other more common strategies;
- a side-by-side comparison between the proposed approach and some SoA approaches;
- optimizing the speed for batch processing;
- and evaluating variable importances.

The final phase consists of packaging the proposed solution into an high-performance API that automatically trains and serves results.

1.3 Development environment and context

1.3.1 The Company

With its roots in Maxideia, a company founded by Miguel Gonçalves in 1999, E-goi (Figure 1.1) established itself in 2004 in Matosinhos. With a mission to empower businesses with the building blocks of an onmnichannel marketing solution, the software as a service (SaaS) platform integrates across different communication channels including email, SMS, push, voice and social media. It provides key features that fit right into the customer-centric toolbox of any company, such as contact segmentation, campaign automation, costumer loyalty managment and detailed reporting. Although E-goi's tools are meant to slot together, they are not bound to it and can be used individually. Its solid foundation enables E-goi to reach a broad range of customers, from small businesses wishing to advertise and grow (Base plan), to more mature companies requiring advanced features and more capabilities (Pro plan), and even tailor-made solutions for large enterprises (Corporate).

e-goi

FIGURE 1.1: E-goi logo

E-goi has reached over 675 thousand customers, sent over 100 billion emails and impacted more than 824 million users, reaching over 20 thousand medium and large enterprises, and still counting. The company works with several large enterprises with significant presence within the Portuguese, Brazilian and Spanish markets.

1.3.2 Framing the work

When it comes to the Machine Learning (ML) products, more precisely, our team (E-goi's AI team) has had the experience of implementing pilot projects for some of the major Portuguese companies (among others) across a diverse set of markets, from supermarket chains, to retail and fashion. This has been done in close partnership with E-goi Digital Solutions.

At E-goi, although we have been successful at implementing tailor-made solutions for the different business needs of our customers, never have we had the chance of working with a news company. Given that there was no past experience with this kind of market, that presented different and unique challenges (addressed in the next chapter 2). So we returned to the drawing board, since none of our previous developments addressed some of the key issues that arise from this type of business model. This thesis is the product of that incursion.

Chapter 2

State of Art

2.1 In the beginning was the Word

The Printing Revolution had its beginning in the fifteenth century Germany, by the hands of the goldsmith Johannes Gutenberg, who invented the movable-type printing press. This happening would actually change the course of history. It kicked into high gear the Renaissance by republishing long-lost classical thoughts by figures like Plato, Aristotle and Cicero, vastly accelerating the rediscovery and sharing of knowledge. By increasing the democratization of knowledge in the Enlightenment era, it powered the development of public opinion and its power to topple the ruling elite. Writing in pre-Revolution France, Louis-Sebástien Mercier declared: "A great and momentous revolution in our ideas has taken place within the last thirty years. Public opinion has now become a preponderant power in Europe, one that cannot be resisted... one may hope that enlightened ideas will bring about the greatest good on Earth and that tyrants of all kinds will tremble before the universal cry that echoes everywhere, awakening Europe from its slumbers". With all this knowledge now available, it contributed to the production of more knowledge by contributing to the Scientific Revolution. Since literacy rates were very low during the fifteen-hundreds, people would gather in their towns to hear a paid reader recite the latest news which radically changed the consumption of news, says Palmer, "it made it normal to go check the news every day". During the last twenty years the newspaper industry has gone through a dramatic transformation. Publishers can now distribute new or updated content in real-time, and readers benefit from having various sources of news online, both on digital news sites or on news aggregation platforms (Google, Yahoo!, MSN, ...) [1].

2.2 An Introduction to Recommender Systems

As the Web established itself as the medium for electronic transactions, recommender systems (RS) technology has gained momentum [2, 3]. Feedback from the customer side is called ratings, which can be *explicit*, if the customer gives a rating to specify their likes and dislikes, or *implicit*, if one does not have that direct response from the customer but instead have clicks, bought products or time-watched. *Implicit* feedback is still feedback, since the simple act of buying or browsing an item may be viewed as an endorsement for that item [4]. RS infer customer interests by utilizing these various sources of data, on the grounds that past proclivities are often good indicators of future choices [2, 3]. In that spirit, different models approach the problem in slightly different ways.

The first approach is to predict the rating for a user-item pair. For m users and n items, this corresponds to an incomplete $m \times n$ matrix, where the observed rating values are used for training. The missing values are then predicted using this model (also referred to as the matrix completion problem). For a merchant what is valuable here is to identify the *top-k* items for a particular user, or determine the *top-k* users to target for a particular item. In this second case the *top-k* can be derived by solving the first formulation for various user-item combinations and then ranking the predictions [2, 3, 5].

It is important to keep in mind that the primary goal of a RS (but not the only one) is to increase product sales (or more specifically, their profit). By recommending carefully selected items to users, RS bring relevant items to the attention of users. In order to achieve the broader business-centric goals, the technical goals of RS are as follows:

1. **Relevance**: a RS has to recommend items that are relevant to the user at hand, otherwise, it will be ignored (if not worst) [6];

2. **Diversity**: The recommended list should contain items of different types, because that raises the probability that the user might like at least one of the items [2, 3];

3. **Novelty**: RS can be win-win if it recommends items that the user has not seen in the past. The opposite, in fact, can lead to the reduction in sales [6];

4. **Serendipity**: Serendipitous recommendations are truly surprising to the user, rather than simply something they did not know about before. Many users may only be consuming items of a specific type, although they may often have latent interests for items of other types which the users themselves might find surprising. This tricky

to tune characteristic can increase sales diversity or begin a new trend of interest in the user, which has long-term and strategic benefits for the merchant [7].

From the perspective of the users, recommendations can be extremely helpful and improve the overall user satisfaction with the web site. Amazon was one of the pioneers in RS, being one of the few retailers that had the foresight to realize the enormous potential of this technology. This contributed to its expansion from being a book retailer to selling virtually all kinds of products [5].

Basically, RS models work with two kinds of data, which are:

1. **User-item interactions**, such as buying behaviour or ratings. These methods are referred to as collaborative filtering (CF) methods [8];

2. **User and/or item variables**, such as age and gender, or product descriptions and keywords. These are called content/contextual-based RS [9, 10].

Some combine these two approaches to create hybrid systems that seek to perform more robustly in a wider variety of settings [11–13].

2.2.1 Collaborative Filtering models

Because most users only view and buy a very small fraction of the available items, most of the ratings are missing. This leads to sparse rating matrices, which poses a challenge to the design of systems that leverage the community interactions [8, 14]. CF methods branch into two general categories:

- **Memory-based methods** (neighbourhood-based CF), in which the user-item ratings are predicted on the basis of their neighbours. They can be user-based, when the similarity functions are computed between users, or item-based, if the similarity functions are computed between items. They however have trouble with sparse matrices [15];

- **Model-based methods**, that develop a model from user ratings. There are two main approaches to developing these models, which are probability calculation or rating prediction. To achieve this, Machine Learning (ML) techniques such as classification, clustering, and rule-based approaches are used. Model-based approaches tend to have better predictions than memory-based, plus it is capable of handling the problem of sparsity and scalability better [15, 16].

Recently, it has been shown that some combinations of memory-based and model-based methods provide very accurate results [17].

2.2.2 Content-based models

In content-based RS, the content information about the items previously rated by a user is analysed to build a model/profile of user interests [10, 18]. They have some advantages in making recommendations for new items with seldom to none interactions, because other items with similar attributes might have been rated by the user at hand [2, 3]. Considering that the community knowledge is not leveraged here, these methods provide *obvious* recommendations, which tends to reduce the diversity of the recommended items [7]. This problem is referred to as overspecialization. It is always desirable to have a certain amount of novelty and serendipity in the recommendations. And although they are effective at providing recommendations for new items, they do not work for new users [2, 3, 10, 18].

These methods have different trade-offs from CF, and are therefore useful in certain *cold-start* scenarios. Despite the disadvantages associated with content-based systems, they often complement collaborative systems quite well because of their ability to leverage content-based knowledge. This complementary behaviour is often leveraged in hybrid RS [11, 12, 18–20].

In case-based RS [9], specific cases are specified by the user as targets or anchor points. Similarity metrics are then defined on the item attributes to retrieve similar items to these cases [2, 3, 3]. Similarly, this approach from knowledge-based systems can be applied to content-based ones, where instead of the user defining the targets explicitly, the items are defined by their buying history. This property will be explored later, in the context of sampling methods.

2.2.3 Context-based: Post-filtering

Contextual information about interactions can also be leveraged to fine-tune recommendations. Such contextual information could include time, location, or social data. For example, the types of clothes recommended by a retailer might depend both on the season and the location of the customer [2, 3, 3].

Some clients may have an issue recommending certain products. What one can do is to extract them *ad hoc*, after the final recommendations have been computed, by masking the unwanted products. This way the full information is presented to train the model and establish correlations between interactions but then only the most relevant choices are kept.

2.2.4 Hybrid Systems

As the three aforementioned systems exploit different sources of input that may work well in different scenarios, many opportunities exist for hybridization, where various aspects from different types of systems are combined to achieve the best of all worlds [3, 12, 13, 19, 20]. The combination of CF and content-based approaches in a way that resolves the drawbacks of each other has demonstrated to improve recommendations when compared to each individual approach [12]. An example of such a recommender system is EntreeC [11].

E-goi's own RS approach, based on the work presented in detail in [13], can be classified as *cascade-hybrid*, combining model, memory and content-based approaches. In this approach, each recommender actively refines the recommendations made by the previous one, sequentially [11, 13, 21]. The first layer acts as a powerful filter and provides a rough ranking, eliminating many of the potential items. The second level of recommendation then uses this rough ranking to further refine it. The resulting ranking is then presented to the user. It uses alternative CF approaches for high and low-interaction users. The adopted methodology also uses a content-based method that employs text analysis for parsing item descriptions, to then calculate similarity between candidate items and past items. It also proposed a straight-forward way to easily incorporate time-awareness into rating matrices. This approach focuses on being intuitive, flexible, robust, auditable and avoid heavy performance costs, as opposed to *black-box* fashion approaches. Although its performance is very satisfying when dealing with matrices derived from retail market interactions, where items circulate during weeks or months before they become irrelevant, it simply does not hold for *cold-item* situations (such as for news recommendations) because the real world relevance of these items decays very quickly, so there is no time to gather user-item interactions.

2.2.5 Types of Ratings

Rating systems heavily influence the performance of RS. Ratings are specified on a scale that indicates the ”level of like” of an item. Rating matrices can also be called utility matrices, if their data refers to the amount of profit or other quantities [22]. They can be continuous (such as in the the Jester joke recommendation engine), interval-based (where a discrete set of ordered numbers - the number of stars in IMDB - are used to quantify like or dislike) or binary (where the user only represents like or dislike, *i.e.* 1 or 0) [23]. Then, they can be further clustered into two main kinds:

- *Explicit feedback*, where the user as to explicitly rate in a predefined scale their level of satisfaction. In *explicit* feedback matrices, ratings correspond to highly discriminant preferences, which makes it easier to apply RS [2, 3];

- *Implicit feedback*, as is the case of unary data, where the customer preferences are derived from user interactions with the items, such as the buying behaviour (bought or not) or watch time (continuous) [4, 22]. However, the act of not buying an item does not always indicate a dislike. Due to the lack of information available about whether a user dislikes an item and the fact that there is often not a sufficient level of discrimination between the various observed values of the ratings, RS using this type of matrices have to be dealt with care [2, 3]. This is one of the key areas addressed in this thesis.

Because only a small fraction of the items are rated frequently (referred to as popular items) the distribution of ratings among items often satisfies the *long-tail* property (Fig. 2.1). This translates into a highly skewed distribution, since most of the items are rated only a small number of times [24–26], which has important implications:

1. Most popular items tend to be relatively competitive and leave little profit for the merchant. It is argued that many companies, like Amazon, make most of their profit in the long-tail [24];

2. Many RS tend to suggest popular items, due to the difficulty of providing robust rating predictions for less frequent items [27];

3. Because of the differences in the rating patterns of the two types of items, high-frequency ones are not representative of the low-frequency [2, 3].

More meaningful predictions should be obtained by adjusting the RS to take real-world properties, such as sparsity and the long-tail, into account [25–27].

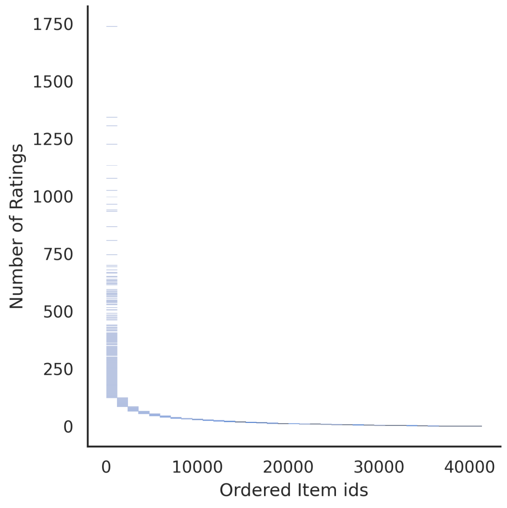


FIGURE 2.1: Long-tail property observed on a dataset from a client

2.3 Introduction to News Recommender Systems

The cheer amount of available news sources and articles, coupled with hourly update cycles, creates an atmosphere of information overload that makes it hard for readers to keep track of news that are most relevant to them Karimi et al. [28]. The power of personalized experiences can be extremely helpful in improving the users' overall satisfaction with the service. By carefully selecting items to users – in this case, news articles – RS bring the most relevant items to the attention of users.

In this thesis, a Decentralized Neural News Recommendation system (DNNR) is proposed. News Recommender Systems (NRS) have certain characteristics related to their business model that are not often, or at all, observed in other domains. The key difference is the speed at which the relevance of the items decay. Unlike item recommendation in music, movies, or the retail market, for example, the relevance of news articles can change very rapidly concomitant with daily happenings and events [28]. This leads to a permanent item *cold-start* problem, since the best news items to recommend have few interactions. Fortunately, news are content-rich, and recent advances in natural language processing (NLP) provide excellent tools to extract rich and compact representations directly from natural text.

2.4 Related Work

According to Jannach et al. [29], CF methods are the most common approach in the RS literature. This is explained by their domain-agnostic application and good overall performance without much information about the business model. Howbeit, things change when it comes to NRS. An analysis of 112 papers that propose one or more recommendation algorithms shows that 59 chose content-based approaches by creating reader's profiles based on past documents of interest and recommending articles that fit the user's pattern [28]. This can be explained by the fact that the main content of News is text, which can be analysed to extract information. In addition, users and community features can also be used, although personal information should be avoided for ethical reasons. Furthermore, because reality changes constantly and people's preferences and interests vary over time, NRS have to keep the user profiles updated.

Akin to other domains, the information gathered from a user can be *explicit* preference information, such as a score in a rating scale, or *implicit*, by simply observing the user's behaviour, such as reading an article, sharing it, printing it, or commenting on it [28].

As an example, The Athena news recommendation system [30] mostly relies on content information. The user profiles are constructed from a set of concepts from the articles the user has read, resulting in a vector with the distinct weighted concepts for applying distance metrics and semantic searches. A similar approach was used for the Ontology Based Similarity Model (OBSM) [31], which calculates news-user similarity based through ontological structures, with user profiles having a *bag-of-concepts* format with DBPedia[*] as a knowledge base in the background.

There are approaches suggesting user segmentation according to their demographic information and article read patterns, weighted term vectors from the topics of the read articles [32, 33].

There are also models that solely rely on click behavior (interactions), like the ones that characterize the Google News Personalization system, which predicts the relevance of an article using both a long-term CF model and a short-term model based on article co-visitations [34]. More recently, an alternative approach was implemented, a Bayesian framework for predicting current news interests from the past predilections of each user and the community trend, combining content-based analysis for the construction of user profiles with an existing CF mechanism to generate personalized recommendations [35].

[*] https://www.dbpedia.org/

There are questions regarding if and how to consider long-term and short-term preferences, for balancing the importance of each article view represents an important point of discussion in news recommendation [36]. There are questions whether two separate models should be built or else a time-decay factor should be included in an integrated model [28].

Lately, novel neural network designs have made considerable progress. NPA [37] is a news recommendation model with personalized attention [38], that uses convolutional neural networks (CNN) to learn hidden representations of news articles based on their titles and learns user representations based on the representations created for their clicked articles. In addition, a word-level and a news-level personalized attention are used to capture different informativeness for different users. Deep knowledge-aware network (DKN) for News Recommendation is a deep learning model which incorporates information from knowledge graph for better news recommendations [39]. It applies knowledge graph representation learning and a CNN framework to combine entity embedding with word embedding and generate a final embedding vector for a news article. An attention-based neural scorer is used for click prediction. NRMS [40] is considered to be the SoA, with its following variants [41, 42]. It consists of a news encoder and a user encoder. It uses multi-head self-attention networks to learn news representations from titles, to model the interaction between words and applies the same principle to learn user representations, by capturing the relatedness between the news read by the user. In addition, additive attention is also used to learn more informative news and user representations by selecting important words and news. It has proven to be very effective. One of its biggest downsides is its black-box nature. The term *cold-start* refers to the situation where there is seldom to none information about an user preferences or no information about a new item that was added.

For a *cold* user, one general approach is to incorporate additional information about the user's context. That can be the location, time of day or demography. An alternative to this is to incorporate features from the news articles for assessing their relevance to an hypothetical generic user, such as the freshness of the news article or its popularity. The "YourNews" system [33], as an example, starts by showing only recently published news, for the penalization process only starts after the first interaction.

For a *cold* item, a content-based approach uses the data from the article to compare with the past content-wise preferences of the individual reader, which solves the problem.

So, the information contained in the articles can be extracted and analysed without it ever being read by anyone, which makes it instantly recommendable without the need for past interactions. This content can be, for example, *named entities* [28].

The belief that the user interest modeling still can be improved, without altering the model nor the embedding process, is one of the premises of this work. Many existing methods for training news RS solely rely on the implicit feedback from user clicks to infer their interests, interpreting the unclicked news as negative samples with a uniform probability – the *missing-at-random* assumption. In the past few years, some contributions have shown that this assumption rarely applies to real-world cases. However, preferences are far from a binary choice of either or, and there can be the case where every news presented to the user could be interesting, but the case was that he only picked one. It is also difficult to accurately sample negative examples without explicit user feedback. The incorporation of negative feedback inferred from the dwelling time of news reading was proposed in [42], to distinguish positive and negative news clicks, via a combination of transformer and additive attention network. The use of factorization machines where also proposed to get negative samples from implicit feedback data when content information cannot be leveraged [43]. This technique has also been used to reduce the amount of negative samples [44]. In [45] negative items are sampled based on how far back in time they have occurred. In this thesis, we propose a new way to approach the negative sampling issue and tackle the training process for news interest modelling, by sampling negative examples that are naturally far away from the user's preferred items in the embedding space.

For efficient news recommendation, text modeling is the key for understanding news content. Existing news recommendation methods usually model news texts based on traditional NLP [37, 39, 40, 46, 47]. There are multiple examples of complex networks that use news representations, in the form of embeddings, from words and entities present in titles or the corpus of the news. However, these techniques do not capture the semantic relationships between words, which results in a shallow representation of the news content.

The introduction of pre-trained language models (PLM) revolutionized NLP, with great text modeling, performance and versatility. Usually, PLMs are pre-trained on a large unlabeled corpus via self-supervision to encode universal text information, and with the aid of their deeper networks, may have greater ability in modeling the complex contextual information in news text [48].

Regarding decentralized recommendation models, – see, for example, [49–52] – the main focus is on learning collaborative filtering protocols for peer-to-peer networked communities. Algorithms are distributed and exclusively based on information exchanged between peers (users) – *i.e.* without orchestration by a central entity. These proposals focus on collaborative models, that essentially exploit patterns in the user-item interactions. In this sense, our proposal is different, since we use a purely content-based approach. However, we borrow the idea of *personal recommenders* proposed with PocketLens [51], since our motivations are very similar.

2.5 Tackling decentralized computing

We have now arrived in an information-centric age, where computing power is unevenly distributed between provider infrastructure and user devices, where most data is generated [53]. Centralized computing power, where most computation involving the training of RS is done, need to efficiently manage and process these large quantities of data, produced in a widely distributed system, which raises some issues:

- **Cost**: To train models and do inference on centralized computing power requires the transmission of massive amounts of data;
- **Latency**: the delay to access the provider's computing infrastructure power and storage is generally not guaranteed, and might restrain some solutions that are more time-critical.
- **Privacy**: training models requires a lot of private information to be carried, raising privacy issues. Organizations with large amounts of user data heightens the risk of illegitimate data use or hazardous private data leaks.

Under these circumstances, on-device or edge computing offers advantages by hosting some computation tasks close to the data sources and end users. When combined with centralized computing it can: alleviate backbone network, by handling key computation tasks without exchanging data with the central computing cluster; and allowing for more agile service response, by reducing or removing the delay of data transmissions [53]. It also has the potential to provide better privacy guarantees, while simultaneously granting users a finer control over processes involving their personal data.

Since mobile edge computing is closer to the data source and users, on-device computing is expected to solve many of these issues [53]. Artificial Intelligence (AI) and on-device computing are already gradually being combined, giving birth to edge intelligence, to provide higher service throughput and better resource utilization, enabling for distributed, low-latency and reliable intelligent applications and services.

A new form of applications that bring advantages to many aspects of people's lives are ML-based intelligent services, such as the ones associated with NRS. We make use of PLMs to enable the content embedding process. With these embeddings, we propose a negative sampling technique to train small individual neural networks, one for each user. This approach yields a performance close to the latest SoA models using the traditional sampling method, demonstrating a high level of personalization. Moreover, by changing the training framework, this contribution also explores possibilities that this approach opens to train on-device distributed models, with the capabilities to alleviate central computing resources. This would impact delay-sensitive applications, protect users' privacy, and maintain high levels of service personalization and customer satisfaction.

2.6 Building blocks

In this section, the most important tools and methods used in this thesis are detailed.

2.6.1 Artificial Neural Networks

At the core of neural networks there are a combination of simple linear algebra calculations, that combined enables the network to capture non-linear relationships and deal better with both outliers and correlated features [54]. The simplest neural network is called a perceptron

$$f(x) = Wx + b, x \in \mathbf{R}^{d_{in}}, W \in \mathbf{R}^{d_{out} \times d_{in}}, b \in \mathbf{R}^{d_{out}}$$

where W is the weight matrix and b is a bias term. In order to go beyond linear functions, a nonlinear hidden layer is introduced, resulting in the Multi Layer Perceptron with one hidden-layer, which has the form:

$$f_{MLP}(x) = W^{(2)}\phi(W^{(1)}x + b^{(1)}) + b^{(2)},$$

$$x \in \mathbf{R}^{d_{in}}, W^{(1)} \in \mathbf{R}^{d_1 \times d_{in}}, W^{(2)} \in \mathbf{R}^{d_{out} \times d_1}, b^{(1)} \in \mathbf{R}^{d_1, b^{(2)} \in \mathbf{R}^{d_{out}}}$$

Here $W^{(1)}$ and $b^{(1)}$ are a matrix and a bias term for the first linear transformation of the input, ϕ is a nonlinear function that is applied component-wise (also called a nonlinearity or an activation function), and $W^{(2)}$ and $b^{(2)}$ are the matrix and bias term for a second linear transform. To this backbone, additional linear-transformations and nonlinearities can be added, resulting in a more complex neural architecture. For deeper networks, it is perhaps clearer to write using intermediary variables as in Algorithm 1.

Algorithm 1 Neural Network X

```
1: $input \to x$
2: $s^{(1)} = W^{(1)}x + b^{(1)}$
3: $h^{(1)} = \phi^{(1)}(s^{(1)})$
4: $s^{(2)} = W^{(2)}h^{(1)} + b^{(2)}$
5: $h^{(2)} = \phi^{(2)}(s^{(2)})$
6: $output \to y = f_{MLP}(x) = W^{(3)}h^{(2)} + b^{(3)}$
```

Feed-forward Neural Networks with several hidden layers are often referred to as fully connected, or dense. Other types of architectures exist, such as convolutional and pooling layers much used in image recognition. Networks with $d_{out} = k > 1$ can be used for k-class classification, by associating each dimension with a class. If the output vector components are positive and sum to one, the output can be interpreted as a probability distribution over class assignments (such output normalization is typically achieved by applying a softmax transformation on the output layer). Matrices entries and the bias components are the parameters of the network which, together with the input, determine the network's output [54].

The training algorithm is responsible for tuning these parameters such that the network's predictions are (almost) correct. Feedforward Neural networks (FFNNs) are differentiable parameterized functions, and are trained using gradient-based optimization. The objective function for nonlinear neural networks is not convex, and gradient-based methods may get stuck in a local minima. Still, gradient-based methods produce good results in practice [54].

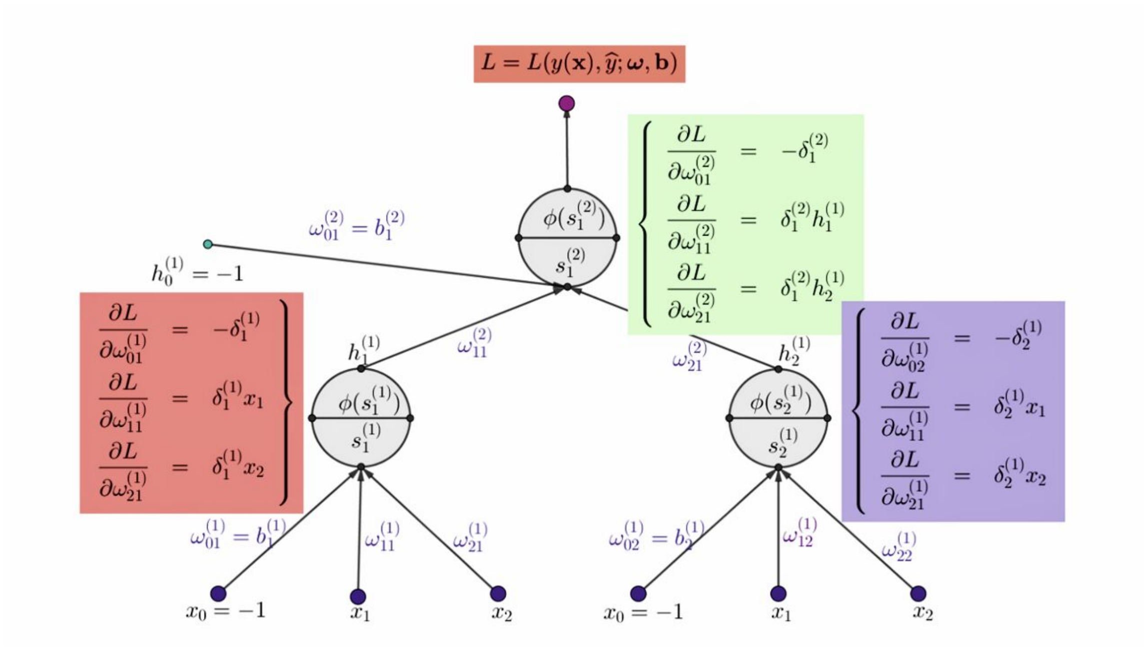
Central to the approach is the gradient calculation, done by following the chain-rule of differentiation. However, for complex networks this process can be laborious and error-prone. Fortunately, gradients can be efficiently and automatically computed using the backpropagation algorithm, which is responsible for methodically computing the derivatives of a complex expression using the chain-rule, while caching intermediary results.

More generally, the backpropagation algorithm is a special case of the reverse-mode automatic differentiation algorithm [54].

Training the weights of the FFNN relies on four main steps:

1. *Forward propagation*: Compute the network output for an input labeled instance x, with label $y(x)$.

2. *Error computation*: Compute the prediction error between the network label prediction $\hat{y}(x)$ and the target label $y(x)$, using a loss function $\mathcal{L}$: $predictionerror(x) = \mathcal{L}(y(x), \hat{y}(x;\theta))$, where $\hat{y}(x;\theta) = \phi(s^{(2)})$ and $\theta = (w_{ij}^{(l)}, b_k^{(l)})$ are the network parameters. Since the loss $\mathcal{L}$ depends on $\hat{y}$, which depends on $w_{ij}^{(l)}$ through $(s^{(l)})$ only, as $\hat{y} = \phi(s^{(l)})$, the application of the chain rule yields Figure 2.2.



$L = L(y(\mathbf{x}), \hat{y}; \boldsymbol{\omega}, \mathbf{b})$
$\frac{\partial L}{\partial \omega_{01}^{(2)}} = -\delta_1^{(2)}$
$\frac{\partial L}{\partial \omega_{11}^{(2)}} = \delta_1^{(2)} h_1^{(1)}$
$\frac{\partial L}{\partial \omega_{21}^{(2)}} = \delta_1^{(2)} h_2^{(1)}$
$\omega_{01}^{(2)} = b_1^{(2)}$
$\phi(s_1^{(2)})$
$s_1^{(2)}$
$h_0^{(1)} = -1$
$\frac{\partial L}{\partial \omega_{01}^{(1)}} = -\delta_1^{(1)}$
$\frac{\partial L}{\partial \omega_{11}^{(1)}} = \delta_1^{(1)} x_1$
$\frac{\partial L}{\partial \omega_{21}^{(1)}} = \delta_1^{(1)} x_2$
$h_1^{(1)}$
$\omega_{11}^{(2)}$
$\omega_{21}^{(2)}$
$h_2^{(1)}$
$\frac{\partial L}{\partial \omega_{02}^{(1)}} = -\delta_2^{(1)}$
$\frac{\partial L}{\partial \omega_{11}^{(1)}} = \delta_2^{(1)} x_1$
$\frac{\partial L}{\partial \omega_{21}^{(1)}} = \delta_2^{(1)} x_2$
$\phi(s_1^{(1)})$
$s_1^{(1)}$
$\phi(s_2^{(1)})$
$s_2^{(1)}$
$\omega_{01}^{(1)} = b_1^{(1)}$
$\omega_{11}^{(1)}$
$\omega_{21}^{(1)}$
$\omega_{02}^{(1)} = b_2^{(1)}$
$\omega_{12}^{(1)}$
$\omega_{22}^{(1)}$
$x_0 = -1$
x_1
x_2
$x_0 = -1$
x_1
x_2

FIGURE 2.2: Neural Networks: Computing the several derivatives of $\mathcal{L}$. Adapted from a Statistical Learning lecture from Professor João Nuno Tavares, FCUP, 2021 [54]

And then the partial derivatives with respect to the weights and biases can be computed, which can be distilled to $\frac{\partial \mathcal{L}}{\partial b_j^{(l)}} = -\delta_j^{(l)}$ for the biases and $\frac{\partial \mathcal{L}}{\partial w_{ij}^{(l)}} = -\delta_j^{(l)} \times x_i^{(l-1)}$ for the weights [54].

3. *Backpropagation*: Compute the gradients in reverse order with respect to the weights. The loss is a smooth function of the parameters (weights/biasses) θ. Its gradient is given by $\nabla \mathcal{L} = \nabla_w \mathcal{L}, \nabla_b \mathcal{L}$. This vector would be nine-dimensional if the input layer had two dimensions and the hidden layer had two neurons, having then six weights and three biases in total. This is needed for the gradient descent method, to improve

the prediction by decreasing the error of the whole network. The computation starts backwards, from the last weights and biases toward the first ones, hence the procedure is called backpropagation. Determining the error for each of the parameters can be done via the chain rule of calculus, to compute the derivatives of each layer (and operation) in the reverse order of forward propagation as seen in Figure 2.3.

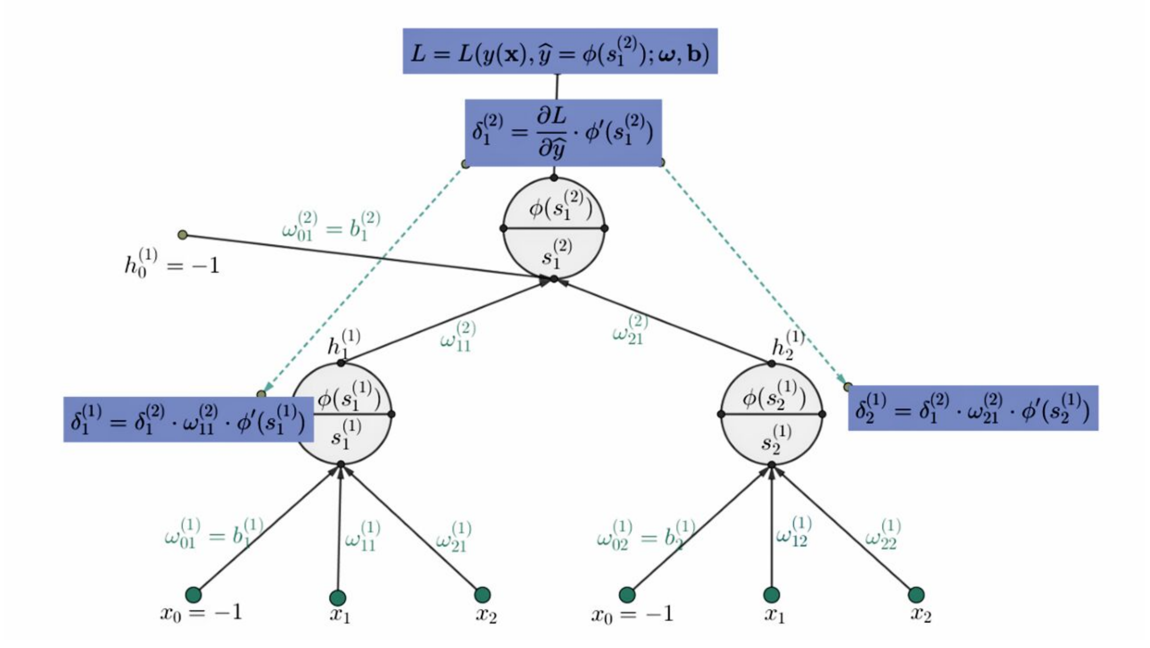

$L = L(y(\mathbf{x}), \hat{y} = \phi(s_1^{(2)}); \boldsymbol{\omega}, \mathbf{b})$
$\delta_1^{(2)} = \frac{\partial L}{\partial \hat{y}} \cdot \phi'(s_1^{(2)})$
$\phi(s_1^{(2)})$
$s_1^{(2)}$
$\omega_{01}^{(2)} = b_1^{(2)}$
$h_0^{(1)} = -1$
$h_1^{(1)}$
$\omega_{11}^{(2)}$
$\omega_{21}^{(2)}$
$h_2^{(1)}$
$\delta_1^{(1)} = \delta_1^{(2)} \cdot \omega_{11}^{(2)} \cdot \phi'(s_1^{(1)})$
$\phi(s_1^{(1)})$
$s_1^{(1)}$
$\phi(s_2^{(1)})$
$s_2^{(1)}$
$\delta_2^{(1)} = \delta_1^{(2)} \cdot \omega_{21}^{(2)} \cdot \phi'(s_2^{(1)})$
$\omega_{01}^{(1)} = b_1^{(1)}$
$\omega_{11}^{(1)}$
$\omega_{21}^{(1)}$
$\omega_{02}^{(1)} = b_2^{(1)}$
$\omega_{12}^{(1)}$
$\omega_{22}^{(1)}$
$x_0 = -1$
x_1
x_2
$x_0 = -1$
x_1
x_2

FIGURE 2.3: Neural Networks: Visualization of backward propagation of the δ's. Adapted from a Statistical Learning lecture from Professor João Nuno Tavares, FCUP, 2021 [54]

4. *Parameter update*: Use stochastic gradient descent to update the weights of the network to reduce the error for that example.

Neural networks might seem to be the obvious choice for the kind of application proposed. Nevertheless, there are strong reasons to why they were chosen. Artificial neural networks offer a number of advantages, from which the most important to this contribution are: i) the ability to implicitly detect complex nonlinear relationships between dependent and independent variables; ii) the statistical flexibility, since many different networks ought to be trained and weights can change from one reader to another, requiring less supervision; and iii) the easiness to parallelize processes (which has great computational advantages). Problems include the proneness to overfitting, chance effects, overtraining and interpretation difficulties [55].

Here, a decisive aspect has been the easiness to parallelize the process, since the computation needs to happen fast so that the system can be scaled to the hundred thousand customers, if needed (which implies training the same amount of networks, without individually supervising each and everyone). Modern neural network packages support GPU parallelization, which drastically reduces computational time, both for research and production scenarios, and are very mature for production environments (see subsubsection 2.6.3.1), which was crucial for the success of this project. Frankly speaking, it is hard to justify any other alternative as the main choice of method and framework for this specific case.

2.6.2 Transformers need Attention

Transformers, introduced in 2017, are a tool for sequence transduction - converting one sequence of symbols to another (being translation tasks its most popular example). This breakthrough has quickly become the gold standard for research and development in natural language processing [38]. This subsection goes through the transformer so that the basic concepts that make it work, and why it works, are clear and understandable.

Firstly, all the words need to be converted to numbers so that matrix operations can be applied to them. One way to convert words to numbers would be to assign each word its own number (as it's the case in one-hot encoding). In one-hot encoding, each symbol or word is represented by an array, the same length of the vocabulary, with only a single element having a value of one. So, a sentence becomes a sequence of vectors. The dot product of any one-hot vector with itself is one (full similarity). And the dot product of any one-hot vector with any other one-hot vector is zero. So, dot products can be used to measure similarity. Matrix multiplication, in this case, can act as a lookup table, using a one-hot vector to pull weights corresponding to the sequence out of a particular row of a matrix, which is a trick used by transformers [38].

One useful way to represent sequences is by using a transition model. If the probabilities for the next word depend only on recent words, it satisfies the Markov property, and we can call it a Markov chain. If it only cares about the single most recent word, it is a first order Markov model. Given a transition matrix, by multiplying a one-hot vector representing one word by the transition matrix, the result is the row that shows the probability distribution of what the next word would be (the transition probabilities). If we look back two words instead of one, it turns into a second order Markov model. The second order matrix has a separate row for every combination of two words, which means that if a vocabulary has size N then the transition matrix has N^2 rows. This results in a matrix with fewer fractions, so looking a the two previous rows gives more context and information to base a next word guess. To improve on the second order language model, third and higher order models can be considered. However, with a significant vocabulary size this would be unreasonable and would need a lot of brute force to execute. Instead, the second order model can consider the combinations of the most recent word with each of the words that came before (that actually appeared). By doing things this way, it is still second order because it is considering only two order at a time, but can reach back further to capture long range dependencies - can be called second order with skips - which moves this paradigm out from the Markov realm. Now each row represents one of many features that may describe the sequence at a particular point. The combination of the most recent word with each one of the words that came before it makes for a collection of applicable rows. Hence, each value in the matrix no longer represents a probability, but rather a vote. Also, in this setting, most of the features don't matter. Most of the words might appear in many sentences, and so the fact that they have been seen is of no help in predicting what comes next [38].

It is possible to further sharpen the prediction by weeding out all the uninformative feature votes, forcing unhelpful features to zero with a mask - a vector full of ones expect the positions to be masked - by multiplying the feature activities with the mask, which will hide a lot of the transition matrix, leaving just the features that matter. This process of selective masking is the attention mentioned in the original paper on transformers [38]. The selective second order model with skips described above is a useful way to think about what transformers do.

In practice, transition probabilities and mask values - model parameters - have to be learned via backpropagation (see subsection 2.6). This means that for any small change in a parameter, it is possible to calculate the corresponding change in the loss. The combination of all the derivatives for all the parameters is the loss gradient. Getting backpropagation to behave well requires gradients that are smooth, that is, the slope does not change very quickly as small steps are taken in any direction (Figure 2.4). They also behave much better when the gradient is well conditioned, that is, it's not radically larger in one direction than another. If the science of architecting neural networks is creating differentiable building blocks, the art of them is stacking the pieces in such a way that the gradient does not change too quickly and is roughly of the same magnitude in every direction [38].

Attention masks are not straightforward to build. If all the mask vectors for every

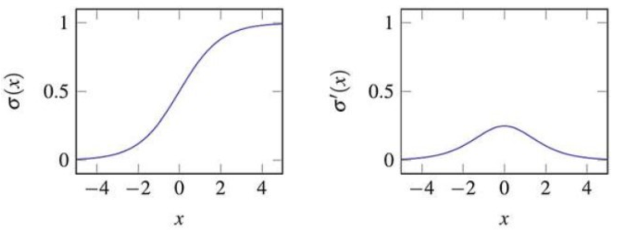


FIGURE 2.4: Sigmoid activation function and its derivative

word are in a matrix (keys: K^T), by using the one-hot representation for the most recent word we can pull out the relevant mask from the matrix (query: Q). This mask lookup is represented by the QK^T term in the attention equation. So, Attention is a matrix multiplication.

$$Attention(Q, K, V) = softmax(\frac{QK^t}{\sqrt{d_k}})$$

Where the query Q represents the feature of interest and the matrix k the collection of masks.

Once the attention step has given its result, a vector that includes the most recent word and a small collection of the words that have preceded it, that has to be translated into features, each being a word pair. For that, a single layer fully connected neural network can be used, and can be calculated by a matrix multiplication with a vector representing the collection of words seen so far - second order sequence model as matrix multiplication. Then, by applying a rectified linear unit (ReLU) nonlinearity, the negative values are replaced with a zero, which cleans up the results so they represent the presence (1) or absence (0) of each word combination feature [38].

All together, the sequence of feature creation matrix multiplication, ReLU nonlinearity and transition matrix multiplication are the Feed Forward processing steps taken after attention is applied. The equation bellow from the original paper ([38]) shows these steps in a mathematical formulation.

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

Where max represents the ReLU, x is the masked word activities, $W_1 + b_1$ is the multi-word feature creation matrix and the $W_2 + b_2$ is the selective second order transition matrix. The architecture diagram from the paper shows these steps lumped together as the ”Feed Forward” block 2.5.

But words are represented as dense embedded vectors, rather than one-hot vectors. The softmax function is used to get results between 0 and 1, but with this the attention mechanism will focus on a single element. To keep several of the preceding words in mind when predicting the next, different instances of attention running at once are used (called *heads*) for the transformer to consider simultaneously several previous words (that is why it is called *Multi-head self-attention*).

Now we have arrived at the part that matters most for our specific use-case, embeddings. For a vocabulary of size, say $N = 50000$, the transition matrix between all pairs of words and all potential next words would have 50000 columns by 50000^2 rows, totaling over 100 billion elements. In a one-hot representation, each word has a vector element, representing a point in space one unit away from the origin along one of the many axes. For a vocabulary of size N the space is N-dimensional. But in an embedding, those word points are all projected into a lower-dimensional space. So, instead of needing N dimensions to specify a word, we would only need, say, 2. Good embeddings group words with similar meaning together. A model that works with embeddings learns patterns in the embedded space, which means that whatever it learns to do with one word is naturally applied to all other words next to it (distance is small between words that behave similarly, semantically), with the added benefit of reducing the amount of training data needed and the number of parameters needed. There is a trade off between computational load and model accuracy since the richness of languages still requires quite a lot of dimensions. Projection matrices can convert the original collection of one-hot vectors into any configuration in a space. As with anything else in the transformer, it can be learned during training. For a sentence what happens is that the embedding part outputs token embeddings for a matrix of N-tokens by M-dimensions from the lower dimensional space (say, 384). Then the matrix is compressed into a single 384-dimensional sentence vector using a pooling function [38].

For positional encoding, a perturbation is added to the words in the embedding space, depending on where it falls in the order of the sequence of words. For each position, the word is moved the same distance but at a different angle, resulting in a circular pattern as it moves through the sequence. Since the embedding space consists of more than two dimensions, the circular wiggle is repeated in all the other pairs of dimensions, which results in a fairly good representation of the absolute position of a word in a sentence.

What we have seen so for is enough to grasp what the embeddings are and how do

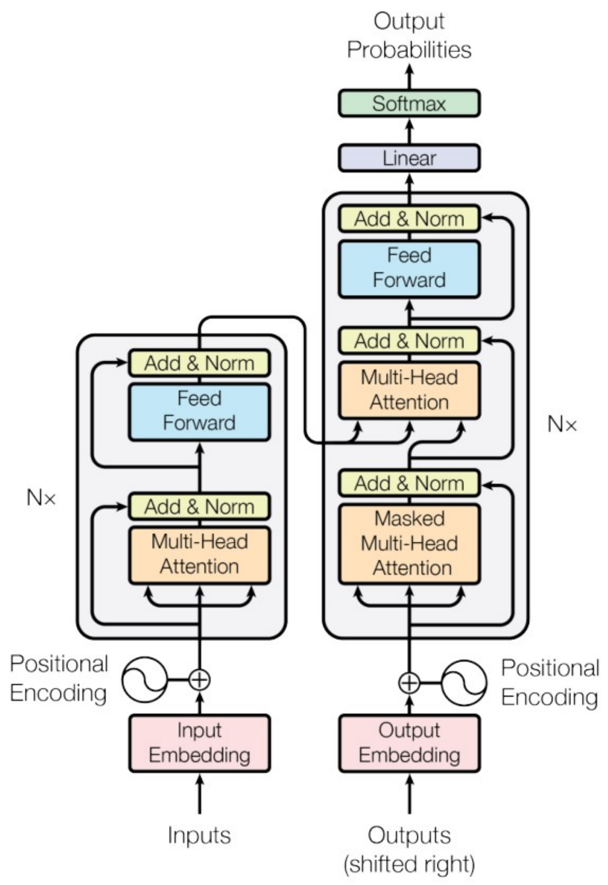


FIGURE 2.5: The transformer: model architecture [38]

transformers create them. The full functioning of transformers might be out of the scope of this thesis, but nevertheless, with just some final remarks we could have a clearer picture of the rest of the process. De-embedding is done to convert embeddings back to words, by a projection from one space to another, that is, a matrix multiplication (the Linear block in the figure from the paper 2.5). The de-embedding matrix is the same shape as the embedding matrix, but with the number of rows and columns flipped (the number of rows is the dimensionality of the space it's converting from and the number of columns is the dimensionality of the space it's converting to - the size of the one-hot representation of the full vocabulary). When an embedded vector representing is multiplied by the de-embedding matrix, the value in the corresponding position is high. Then, the one-hot vector is recreated by choosing the word associated with the highest value. But it might be better to have a "soft" maximum function, because some option might only be infinitessimally larger than other. The softmax of the values in a vector converts the de-embedding results to a probability distribution, making it easier to compare the likelihood of different words. Furthermore, because it is differentiable (calculate how much each element of the results change given a small change in any of the input elements) it can be used with backpropagation to train the transformer [38].

The original input matrix is constructed by getting each of the words from the sentence in their one-hot representation, and stacking them such that each of the one-hot vectors is its own row. The resulting input matrix has n rows (sentence length) and N columns (vocabulary size) ($[n \times N]$). The embedding matrix has N rows and D columns (number of dimensions in the embedding space) ($[N \times D]$). Multiplying the two matrices results in an embedded word sequence matrix a shape of $[n \times D]$. After the initial embedding, the positional encoding is additive, rather than a multiplication, so it doesn't change the shape of things. Then the embedded word sequence goes into the attention layers, and comes out the other end in the same shape. Finally, the de-embedding restores the matrix to its original shape, offering a probability for every word in the vocabulary at every position in the sequence [38].

At this point the reader is owned an apology if this overview about transformers has been confusing to follow. The author of this thesis has read about this several times and is still mazed. Transformers are indeed very complex, but that is part of the reason why they are so fascinating. Hopefully the future holds some time to work on one, but for now we have more than enough information about them to proceed. This is arguably the most difficult piece of the puzzle to understand, but at the same time, one of the easiest to implement in production, given the availability of large PLMs (subsection 2.6.3.2) [56].

2.6.3 General Tools Used

2.6.3.1 Python programming language

Python (available at www.python.org) is a powerful and fast programming language developed under an OSI-approved (Open Source Initiative) open source license, making it freely usable and distributable. Python's license is administered by the Python Software Foundation (www.python.org/psf/). Python's design emphasizes code readability, has a simple syntax that makes it easier to learn and read and enables a clear programming on both small and large scales. Python was used as part of the Anaconda Distribution, a cross-platform free and open source distribution of Python and R for data science applications. It as a simplified package management system, conda, that eliminates the need to install every package independently (available at www.anaconda.com).

2.6.3.2 Pytorch

PyTorch (Figure 2.6) is a Python package that provides two high-level features: Tensor (vector) computation (like NumPy) with strong GPU acceleration; Deep neural networks built on a tape-based autograd system. Usually, PyTorch is used either as a replacement

PyTorch

FIGURE 2.6: Pytorch logo

for NumPy to use the power of GPUs, providing a deep learning research platform that provides maximum flexibility and speed. PyTorch Tensors can live either on the CPU or the GPU, which accelerates the computation by a huge amount. It has a wide variety of tensor routines to accelerate and fit the scientific computation needs such as slicing, indexing, math operations, linear algebra, reductions. PyTorch has minimal framework overhead, integrating acceleration libraries such as Intel MKL and NVIDIA (cuDNN, NCCL) to maximize speed. At the core, its CPU and GPU Tensor and neural network backends are mature and have been tested for years. Hence, PyTorch is quite fast – whether running small or large neural networks (available at pytorch.org). And being fast was a prerequisite for this project from the start.

2.6.3.3 Hugging Face Transformers

Hugging Face Transformers (Figure 2.7) supports an open-source library with the goal of opening up these advances to the wider machine learning community, with carefully engineered SoA Transformer architectures available under a unified API. Backing this library is a curated collection of pretrained models made by and available for the community. Hugging Face Transformers is designed to be extensible by researchers, simple for practitioners, and fast and robust in industrial deployments [56].

2.6.3.4 FastAPI

FastAPI is a modern, fast (high-performance), web framework for building APIs with Python 3.6+ based on standard Python type hints. It has a very high performance, on

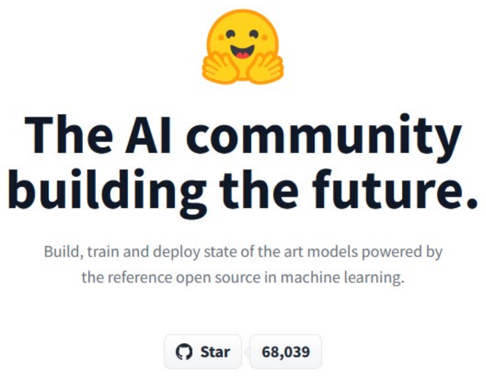
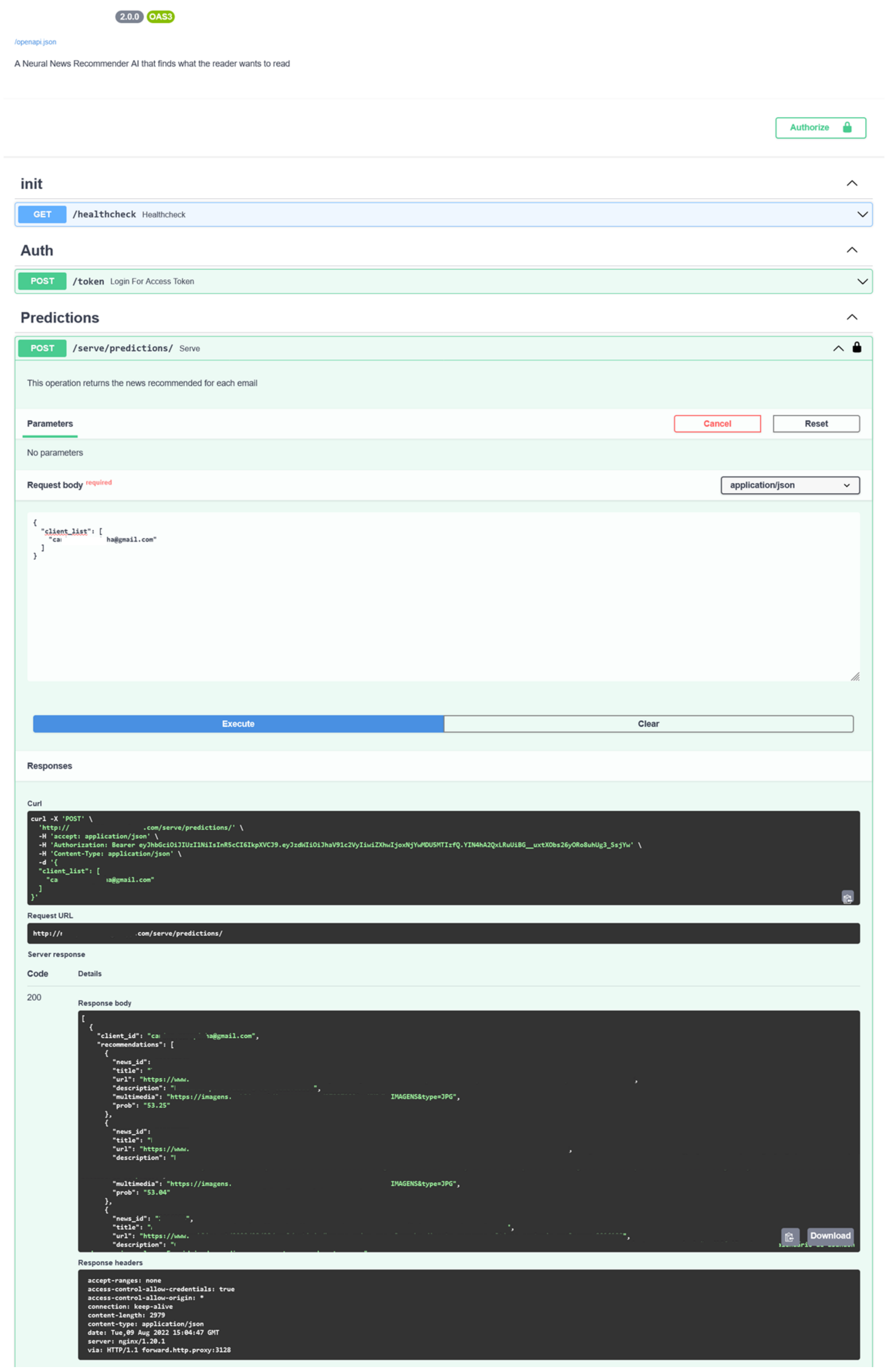

The AI community
building the future.
Build, train and deploy state of the art models powered by
the reference open source in machine learning.
Star
68,039

FIGURE 2.7: Hugging Face logo

par with NodeJS and Go (thanks to Starlette and Pydantic), being one of the fastest Python frameworks available. It is easy to pick-up and intuitive, increasing the development speed. It is robust to get production-ready code, with automatic interactive documentation, and based on (and fully compatible with) the open standards for APIs: OpenAPI (previously known as Swagger) and JSON Schema. Hence, FastAPI was the chosen framework for providing real-time predictions from the underlying model (available at fastapi.tiangolo.com). The Swagger UI for the system in production that was derived from this thesis can be seen in Figure 2.8 (confidential information has been redacted).



A Neural News Recommender AI that finds what the reader wants to read

Authorize
init
GET /healthcheck Healthcheck
Auth
POST /token Login For Access Token
Predictions
POST /serve/predictions/ Serve

This operation returns the news recommended for each email

No parameters

Request body required

application/json

```
{
  "client_list": [
    "ca                ha@gmail.com"
  ]
}
```

Curl

```
curl -X 'POST' \
  'http://                    .com/serve/predictions/' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJhaV91c2VyIiwiZXhwIjoxNjYwMDU5MTIzfQ.YIN4hA2QxLRuUiBG__uxtXObs26yORo8uhUg3_SsjYw' \
  -H 'Content-Type: application/json' \
  -d '{
  "client_list": [
    "ca              a@gmail.com"
  ]
}'
```

Request URL

```
http://                    .com/serve/predictions/
```

Server response

Code Details

200

Response body

```
[
  {
    "client_id": "ca              a@gmail.com",
    "recommendations": [
      {
        "news_id":
        "title": "
        "url": "https://www.                                                              ,
        "description": "                          ",
        "multimedia": "https://imagens.                                    IMAGENS&type=JPG",
        "prob": "53.25"
      },
      {
        "news_id":
        "title": "
        "url": "https://www.                                                    ,
        "description": "

        "multimedia": "https://imagens.                                    IMAGENS&type=JPG",
        "prob": "53.04"
      },
      {
        "news_id": "       ",
        "title": "                                                   ",
        "url": "https://www.                                                           ",
        "description": "
```

Response headers

```
accept-ranges: none
access-control-allow-credentials: true
access-control-allow-origin: *
connection: keep-alive
content-length: 2979
content-type: application/json
date: Tue,09 Aug 2022 15:04:47 GMT
server: nginx/1.20.1
via: HTTP/1.1 forward.http.proxy:3128
```

FIGURE 2.8: Swagger UI for the production system, with a request example

Chapter 3

Methods

3.1 Multi-network training framework

Since the main contributions of this work are two-fold – the distributed model architecture and the negative sampling technique that supports it –, the general training framework is explained before its individual components, *i.e.* the neural network design, the embedding process using PLMs and the negative sampling technique. These components combined create the foundations for the proposed training framework, which changes the way these processes can be approached to tackle issues related to scalability, latency, cost, and most importantly, privacy.

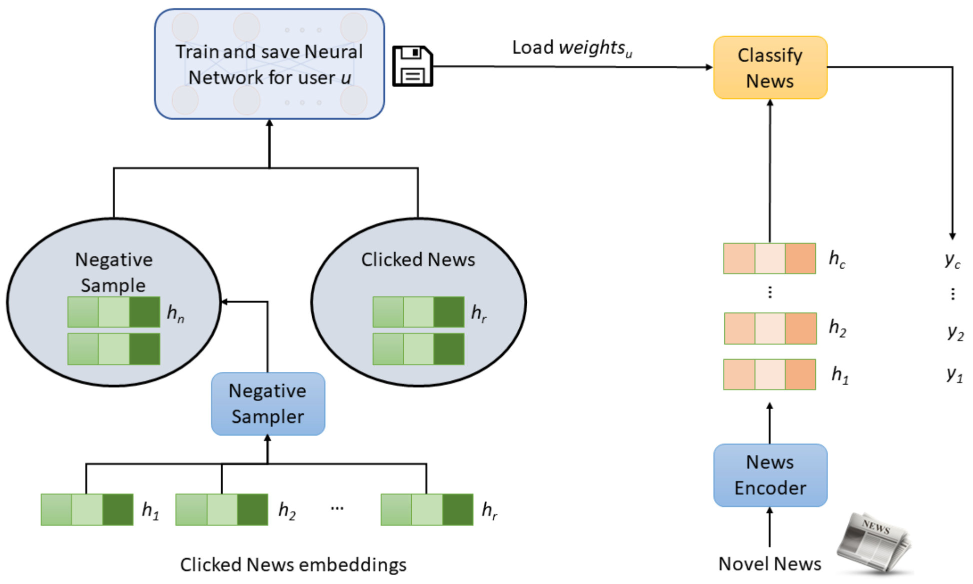


FIGURE 3.1: Training with negative samples Framework

3.1.1 News Recommendation Training Framework

The main components in this training framework include a news encoder to transform news variables into fixed-size tensor embeddings, a negative sampler to generate synthetic negative samples based on the user's history and the classification module. The news encoder processes r read (or clicked) news for each user and creates its embeddings denoted as $[h_1, h_2, ..., h_r]$. The news encoder also creates embeddings for the c candidate news: $[h_1, h_2, ..., h_c]$. For each user u, the negative sampler creates a synthetic negative sample of size n $[h_1, h_2, ..., h_n]$ based on the r read news, with $n = r$ to achieve a balanced sampling every time. A small neural network for each user is then trained with the *Synthetic Pool* (the users' history coupled with the synthetic negative feedback - explained in more detail in subsection 3.1.4) and saved for later, when novel candidate news will need to be scored. Figure 3.1 illustrates the training framework with all of its components.

3.1.2 Neural-network design

To achieve fast training and prediction in a decentralized setting, neural networks need to be lightweight. Network layers were kept to 10 for quick *epochs*. From the total number of n-features, the first dimensions corresponding to the PLM-embedded titles run through 4 initial layers to reduce its dimensionality from 384 to 64. Then, this 64-dimensional vector is concatenated to the rest of the original vector to continue through the feed-forward network. All layers have rectified linear unit (ReLU) activation functions, excepting the forth layer, which uses a Hyperbolic Tangent (Tanh) activation function, before the concatenation.

After concatenation, the activations of the previous layer for each given example are normalized by passing through a normalization layer. Two Dropout points are added, randomly zeroing 0.2 of the elements of the input tensor, which has proven to be an effective technique for regularization and preventing the co-adaptation of neurons as described in [57]. It outputs to a 2-dimensional vector that can be interpreted as the inverse-sigmoid of the threshold. Hence, a sigmoid function is applied to the output for class 1, which corresponds to the *read* class, to get the read probability $Read\ Probability(output[,1]) = Sigmoid(x) = \frac{1}{1+exp(x)}$. This transforms the problem into a regression, which is important to order the recommendations based on the clicking probability. That is why the loss function used is mean squared error (mse) instead of binary cross-entropy. Figure 3.2 displays a graphic representation of these small neural networks.

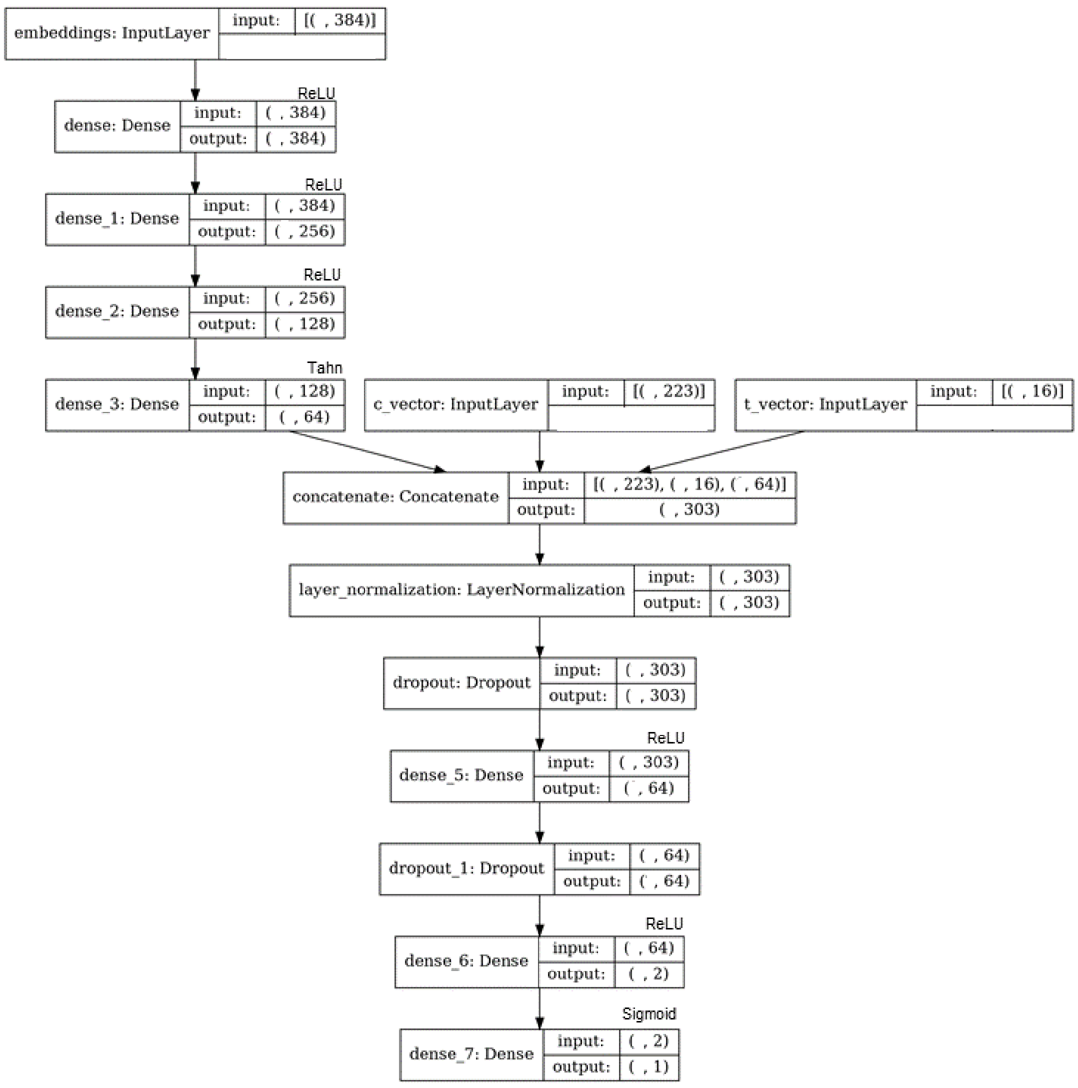


FIGURE 3.2: Feed-forward Neural Network Architecture

3.1.3 PLM-powered Embeddings

Pre-trained Language Models (PLM) were used to empower the content embedding process. In this case, a deep self-attention distillation of a multilingual pre-trained model [58] was used due to its speed to size relationship. Figure 3.3 illustrates the news encoder module, which embeds news titles to a 384-dimensional vector using a PLM, and one-hot encodes the news category and type into fixed-size vectors. These embeddings have a crucial role in the whole process:

- they are required for the negative sampling method proposed.
- the three fixed-size vectors are concatenated and fed to the neural networks to train.

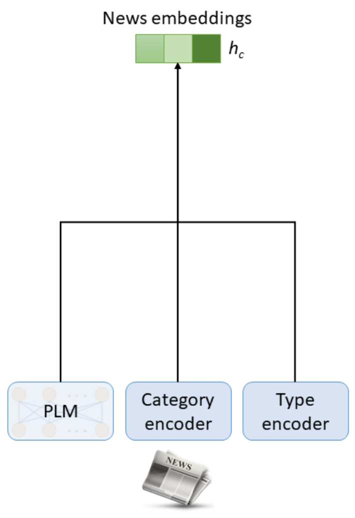


FIGURE 3.3: PLM powered News encoder

3.1.4 Negative Sampling Technique and Model Training

The creation of a synthetic negative sample for each user was approached as essential to the success of the proposed distributed approach, using the history for each user as the reference. To achieve this, the news title embeddings are indexed to search in the L2 space the farthest news from the reference centroid of the user. The reference centroid is computed as the averaged embeddings of all titles from the user's history. The squared Euclidean (L2) distance is monotonic as the Euclidean distance, but if exact distances are needed, an additional square root of the result is needed. The inner product was used for maximum inner product search. It is not by itself cosine similarity, unless the vectors are normalized (lie on the surface of a unit hypersphere). For l^2-normalized vectors x, y, $||x||_2 = ||y||_2 = 1$, we have that the squared Euclidean distance is proportional to the cosine distance (equation 3.1).

$$||x-y||_2^2 = (x-y)^T(x-y) = x^Tx - 2x^Ty + y^Ty = 2 - 2x^Ty = 2 - 2cos(x,y) \qquad (3.1)$$

Then we take the positive sample, which is simply the users' history and feed the pooled sample with positive and synthetic negative feedback to the model. We refer to this pooled sample as *Synthetic Pool*. For a perfectly balanced dataset, the length of the negative sample matches the length of the positive one, which facilitates the learning process. A limit of 60 more recent samples was introduced, since it was enough to capture

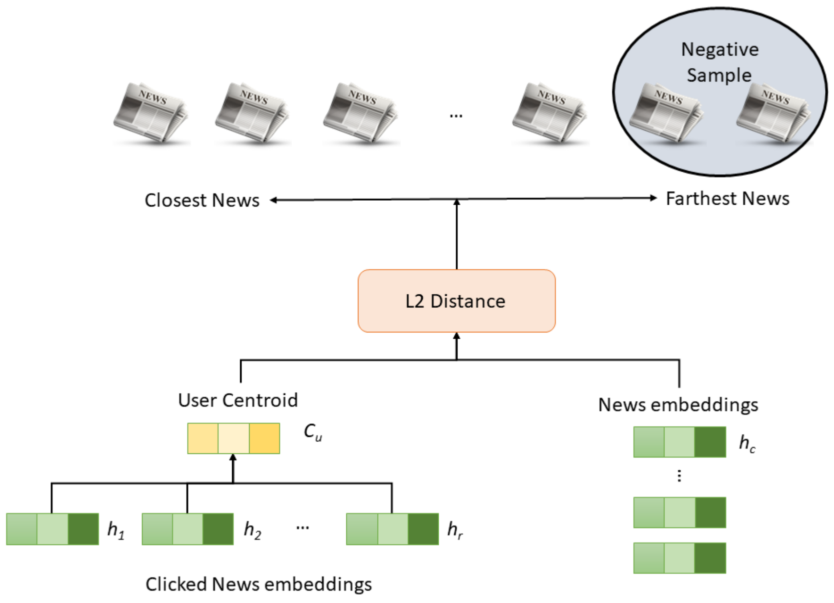


FIGURE 3.4: Synthetic Negative sampling method

user profiles while cutting on training time. The process can be condensed in a few lines (algorithm 2), and by its graphical illustration, in figure 3.4. To obtain a reliable evaluation of this technique's performance, it was compared against two other variants: (i) a model trained only on news impressions – which contain news that were presented to the user, as well as an indicator to whether the user clicked or not [59] –, taking the non-clicked news as negative feedback; and (ii) a model trained on the data with random news taken from the pool of unread news as negative feedback. In other words, the comparison was made by testing the models trained on three different samples - Synthetic Pools, news impressions, and random sampling - against the news impressions.

Algorithm 2 Create Synthetic Negative Samples

```
 1: procedure ALGORITHM
 2: $embeddings \leftarrow$ embeddings for news read by$users$
 3: $embeddings =$ L2-norm(embeddings)
 4: For each user $u$:
 5:     $embedding\ vectors\ for\ user\ u \leftarrow embeddings(u)$
 6:     $User\ centroid = embedding\ vectors\ for\ user\ u$
 7:     $inner\ product\ indices \leftarrow Sort(\langle centroid\ ,\ embeddings\rangle)[indices]$
 8:     $Sample\ Length \leftarrow Length(embedding\ vectors\ for\ user\ u)$
 9:     $Synthetic\ Pool\ for\ user\ u \leftarrow inner\ product\ indices[Sample\ Length]$
10:     return $Synthetic\ Pool\ for\ user\ u$
```

Chapter 4

Results

4.1 Experiments and results

The experiments were conducted on a real-world dataset from Microsoft, the famous MIND dataset [59]. This dataset is mono-lingual (english) and has data from a news aggregator – *i.e.* it includes multiple news sources –, having high content diversity. The MIND-small dataset has anonymized behavior logs from the Microsoft News website. It contains click histories and impressions logs of 50,000 randomly sampled users who had at least 5 news clicks during 6 weeks from October 12 to November 22, 2019. Table 4.1 contains the MIND dataset statistics. When it comes to the content of this dataset, table 4.2 shows an example line from the behaviors file, from which the history and impressions for each user were extracted.

TABLE 4.1: MIND Dataset Statistics

Parameters	Value
# Users	50,000
# News	51,282
# Unique Interactions	926,058
# Impressions	1,804,520

Due to a limit of GPU memory, batch sizes were kept to 64 (60 in the case of DNNR) which is size used in [40]. The number of epochs per user were set to 15, since loss values stabilize at around 15 iterations (Figure 4.1).

TABLE 4.2: MIND Dataset Example Content

Column	Content
Impression ID	91
User ID	U397059
Time	11/15/2019 10:22:32 AM
History	N106403 N71977 N97080 N102132 N97212 N121652
Impressions	N129416-0 N26703-1 N120089-1 N53018-0 N89764-0 N91737-0 N29160-0

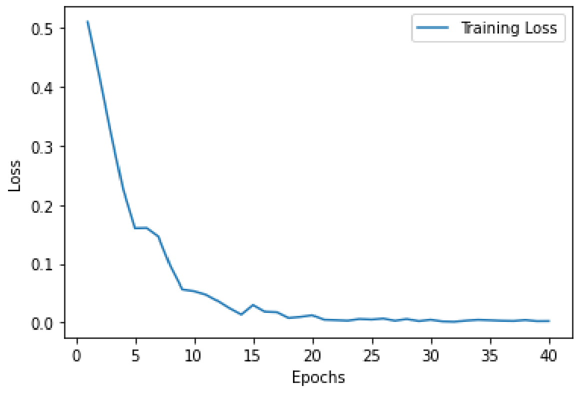


FIGURE 4.1: Loss per epoch during FFNN training

4.1.1 Getting in touch with the data

To get to know the dataset at hand, firstly we looked at the amount of different news types there were and the amount of different news categories. There are 16 news types that accommodate 212 news categories. This diversity is great, providing a good starting point for the construction of the feature vector. Figure 4.2 shows the barplot for the top 15 read news categories from the MIND dataset (a) and the news types from the MIND dataset (b). The most read news category is the ’newsus’, News related to the United States, followed by NFL football, politics and crime, the expected subjects. As for the types, the predominant one is ’news’, followed by sports and lifestyle. Although the distribution across different categories and types is not balanced (as expected), every category (in types and categories) are fairly well represented (except for the types ’kids’ and ’middleeast’, that apparently don’t get that much traffic.

User activity was further characterized with some descriptive statistics applied to the amount of items each user read, the amount of different types of news consumed and the amount of different news categories (Table 4.3).

It is possible to observe that most users read just 19 news during the specified timeframe, which is not a lot. The distribution is fairly left-skewed (Figure 4.3), with users that

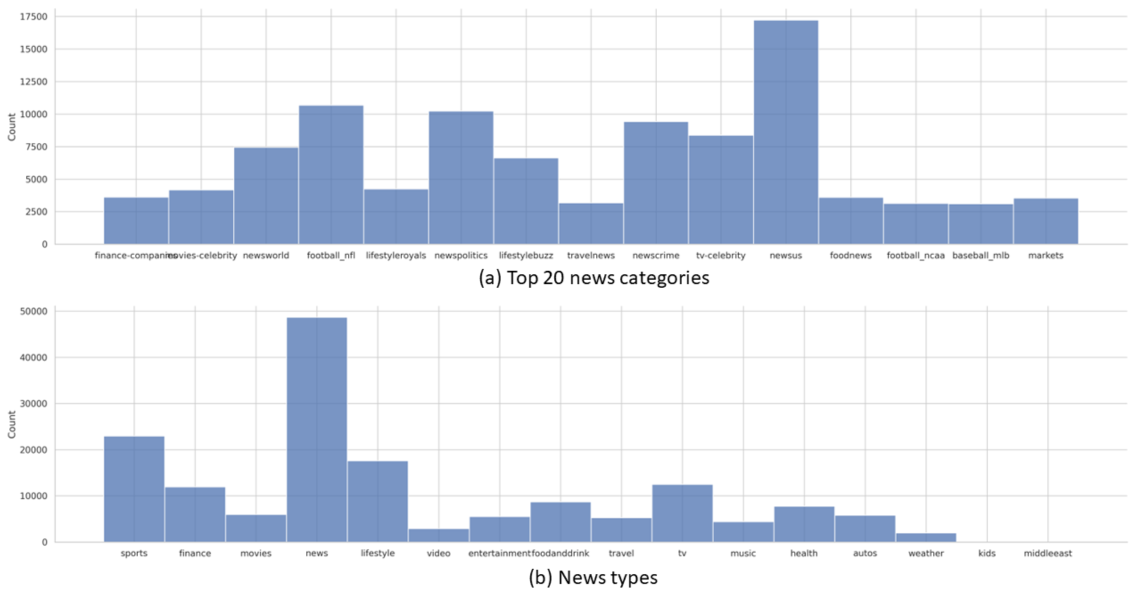


FIGURE 4.2: (a) Top 15 read news categories from the MIND dataset. (b) News types from the MIND dataset

TABLE 4.3: Descriptive statistics on user activity.

Metric	Items read	Types read	Categories read
mean	28.10	6.97	13.68
std	30.26	3.23	9.53
min	1	1	1
25%	8	4	6
50%	19	7	12
75%	37	9	19
max	343	14	73

read more than 80 news items $(37 + 1.5 * (37 - 8))$ being considered outliers. The most avid reader in this sample reached 343 readings, which is more history than we will ever need. However, there are many users with low item counts, which is not ideal to model their preferences.

News articles are grouped within 16 news types, which is good because it gives us another variable to infer user preferences other than just the plain text contained in the title. Although there are 16 news types to chose from, the max number of different news types a client consumed were 14 with most of them reading around 7. The distribution (Figure 4.4) is close to symmetrical, with most of the clients reading less than half of the available news types, which is compatible with the varying preferences among clients.

Finally, news articles are distributed across 212 different categories, which is a fairly

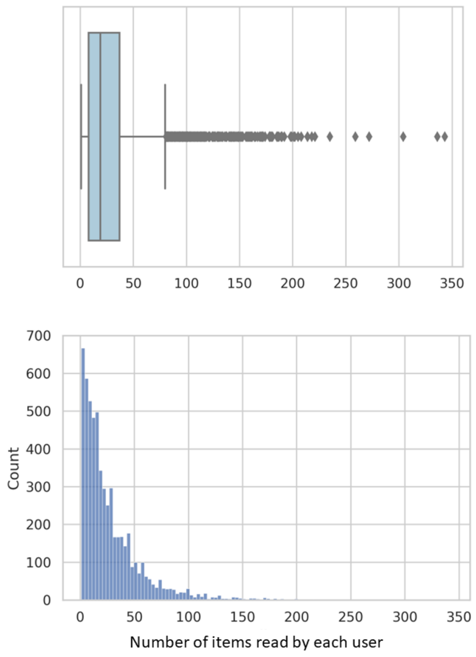


FIGURE 4.3: Distribution analysis of the number of items read by each user

good amount of dimensions to use when modelling reading preferences (Figure 4.5). We can see that most users spread around 12 categories, but the distribution is left-skewed, which means that there are some outliers which read across a much more diverse set of news categories, reaching a maximum of 73, with several outliers reading above 38 $(19 + 1.5 * (19 - 6))$.

4.1.2 Choosing the optimal number of max samples per user

It was already mentioned the decision to take the most recent samples (page views) up to 60 (see Section 3), if ever the user did read that much. However, here lies the explanation for that decision. Although it could be anchored just based on the analysis of the interaction between computational time and performance (area under the receiver operating curve - AUC) alone, picking the subjectively better sample size based on the trade-off between these two metrics, the decision would be better off if the differences in individual AUC between max sample sizes were statistically significant. Figure 4.6 sums up the Kruskal-Wallis test for different medians coupled with the Dunn's pairwise test between samples for 1800 samples. Due to the size of the samples, the visual differences are hard

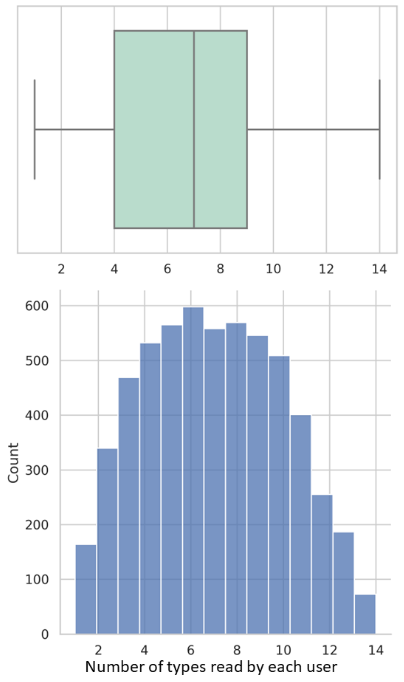


FIGURE 4.4: Distribution analysis of the number of different types of news read by each user

to pinpoint, but the $p_{Bonferroni-adjusted} = 0.02$ indicates that at least one sample has a different median from the rest, it can be said that the four max sample sizes produce different effects.

From the Dunn's test the only statistically significant differences in median values (Bonferroni corrected) are between 120 and 15 max samples and between 15 and 60 samples, with 60 max samples achieving the highest individual AUC median, with no statistically significant difference detected between 60 and 120 max samples. Although the distributions appear to be relatively symmetric with a similar variance, here, due to the amount of samples (which up-eases the argument of statistical power), what interests the most is the median. That is why the Kruskal-Wallis test was chosen. To help decide the better threshold (between 30 and 60), an ANOVA test was performed to evaluate the difference between the group AUC (figure 4.6). For this, six sample runs were performed for each threshold. Figure 4.7 shows the difference in group AUC, and there is at least one mean that is significantly different from the rest. Here the one-way ANOVA seems to be the correct approach, since there are few samples (the mean is more interesting) and the distributions are approximately symmetric and homocedastic. Student's t was performed for pairwise post-hoc testing. All differences are statistically significant except for the one between 15 and 30 ($p_{Bonferroni-adjusted} = 0.10$). Most importantly, the clear choice seems

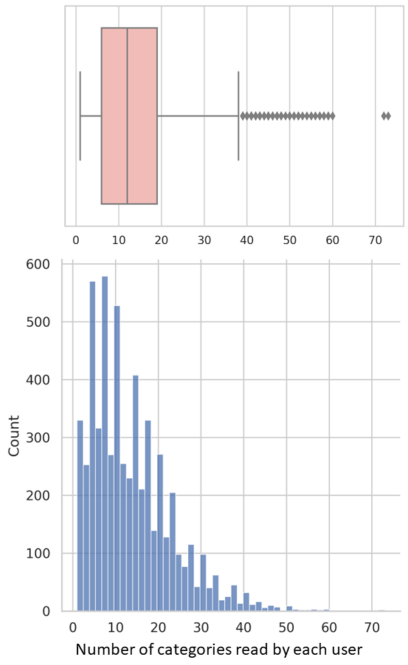


FIGURE 4.5: Distribution analysis of the number of different categories of news read by each user

to be 60 samples, since it has a statistically higher group AUC when compared to both 30 ($p_{Bonferroni-adjusted} = 3.26e-05$) and higher 120 although not statistically significant.

Finally, it was inspected if the number of training samples, within a certain cap (tested for 60 max positive samples per user) had a significant effect on the final AUC score for each user. For that, a simple linear model with a random sample of 500 AUC values and the length of the training sample (s_length) for each user was fitted and the coefficients analysed. Though the $p-value$ appears to be significant (see Table 4.4), the estimate for the sample length indicates a very small negative effect $-8.449e-06$ (more training samples are associated with lower AUC scores). So, if for each additional sample we can expect a decrease of this magnitude in the AUC score for a particular reader, even if a reader were to have 100 more training samples, we should only expect to see a drop in the AUC score of approximately 0.85%. This does not change the fact that the result above stated is counter intuitive. We would have thought that if we had more information about a particular user, we could serve better recommendations and not worse. But avid readers tend to read across a more diverse spectrum of topics, and this diversity could be inserting more uncertainty in the modelling process. This being said, the possibility that readers with higher training sample counts might have bellow average performance, and thus significantly worse recommendations (leading to a worse user experience), is excluded.

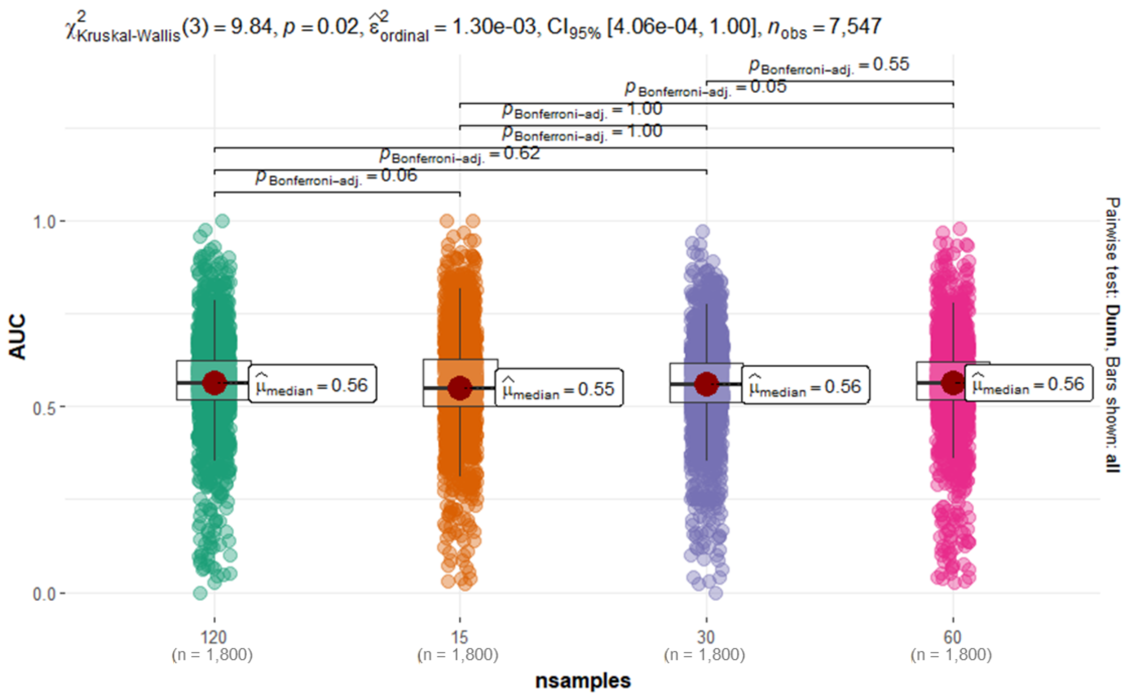


FIGURE 4.6: Max sample size effect on individual AUC

TABLE 4.4: Linear Regression $AUC(s\_length_i) = \beta_0 + \beta_1(s\_length_i)$.

	Estimate	Standar Error	t value	*p-value* (>—*t*—)
(Intercept)	5.879e-01	1.169e-02	50.28	<2e-16 ***
s_length	-8.449e-06	3.985e-06	-2.12	0.0365 *

4.1.3 Impact of the Synthetic Negative Sampling

To better understand the impact of this sampling method, the comparison was made between training the small user networks on three different types of samples and testing them on impressions.

Figure 4.8 shows the difference in AUC when training the models on three different datasets: Impressions, Random Sampling, Synthetic Pools. While the simple neural networks trained on the news impressions cannot hold well, training these networks on the Synthetic Pools manages to model news profiles well enough to perform significantly better. Since the distributions are approximately symmetrical but do not have equal variances, the Welsh test was applied to check if the three population means are equal. Here, we can say that they are clearly not ($p = 2.81e - 06$), so at least one mean is statistically different than the others. Because group variances are unequal, the post-hoc pairwise test applied was the Games-Howell test, as an alternative to Tukey-Kramer. Although the difference between Impressions and Random Sampling is not statistically significant

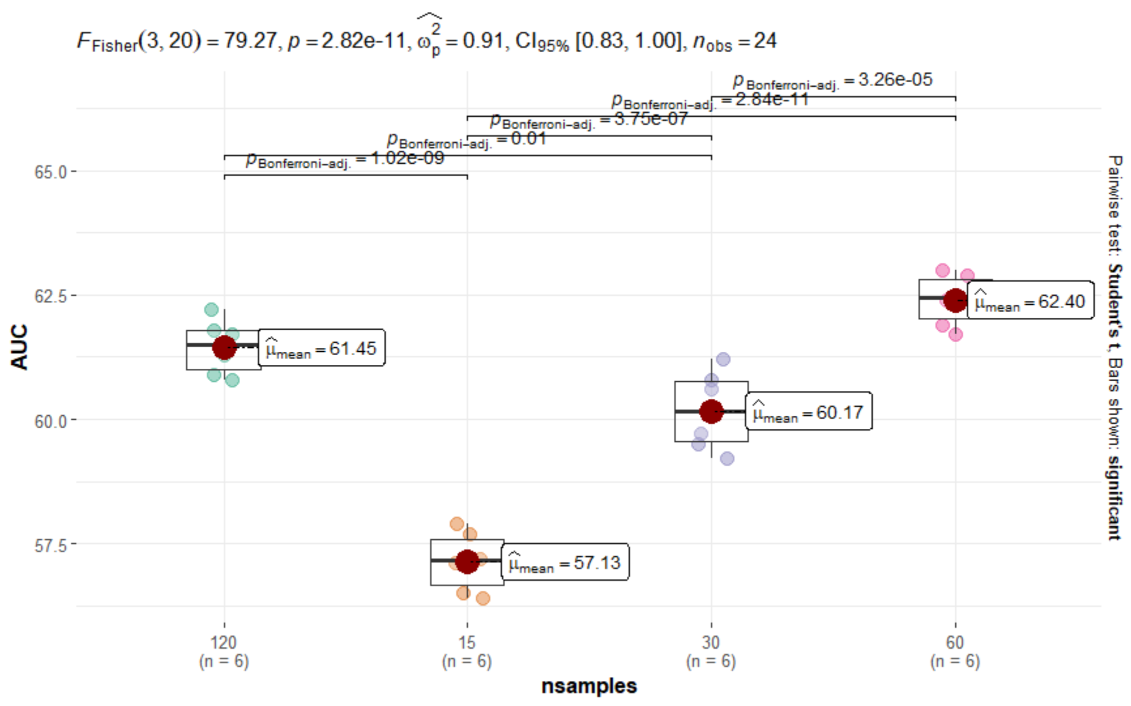


FIGURE 4.7: Max sample size effect on group AUC

($p_{Bonferroni-adjusted} = 0.39$), there is a statistical difference between Synthetic Pools and Impressions or Random Sampling ($p_{Bonferroni-adjusted} = 6.81e-04$, $p_{Bonferroni-adjusted} = 2.54e-05$).

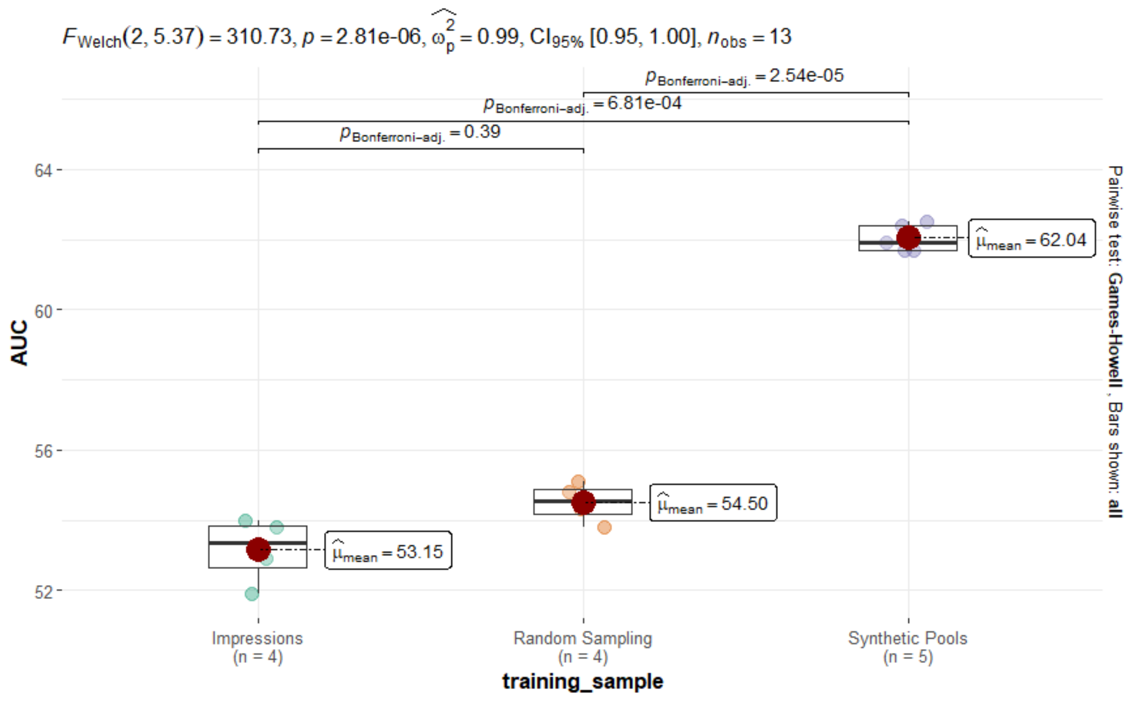


FIGURE 4.8: Influence of the Synthetic Pool: Group AUC comparison

To have a clearer picture of what is going on, the individual AUC was also tested. Figure 4.9 shows the difference in individual AUC for 1800 readers, when training the models on three different datasets (some cases failed using Impressions due to the inability to accurately model preferences, failing to identify even a single true positive). While for the group AUC the difference between Impressions and Random Sampling was not statistically significant, in the individual AUC there is a marked difference. Using the impressions with this method has terrible results, essentially making the prediction random, with a medium AUC of 50%. Using the Random Sampling to obtain negative samples improves a lot, with a statistically significant difference, but we can visually see that the variance is quite high. The Synthetic Pools seem to have two major consequences: significantly raise the individual AUC when compared to the Random Sampling ($p_{Bonferroni-adjusted} = 1.41e - 10$), and shortening the variance, which is desired since the results are more consistent across readers with different reading habits and routines.

These results substantiate this technique's ability to efficiently model user preferences, provided that the Synthetic Pools assure a clear distinction between read news and putatively uninteresting news (possible by the synthetic negative feedback sampling method). It is noticeable that even a random sampling of negative samples from the pool of unread news can improve the training of these networks, achieving better mean performance than the networks trained on news impressions alone.

Since the synthetic samples are gathered based on the dissimilarity between them and the user's historical news, it is easier to get better results when testing on this set. Conversely, the news impression set may not have so much dissimilarity between them, since they are based already on some kind of recommendation system, making it a harder set to capture user profiles and testing predictions. News impressions are collected from a pool of news that were presented to the user, with the information if the user clicked or not, it can already be filtered in a way that the user might have done a coin toss between two of them. So, some news marked as 0 (not clicked) might have been of some interest to the user, but for some reason other than disliking the subject they were not clicked. By checking the first three principal components from the PCA analysis (figure 4.10), it is clear that we cannot observe clusters in the impressions, nor the random sampling sets. As for the Synthetic Pools, there is a marked distinction between the clicked historical news for an user, and its synthetic negative feedback. Thereupon, if we consider a classification task,

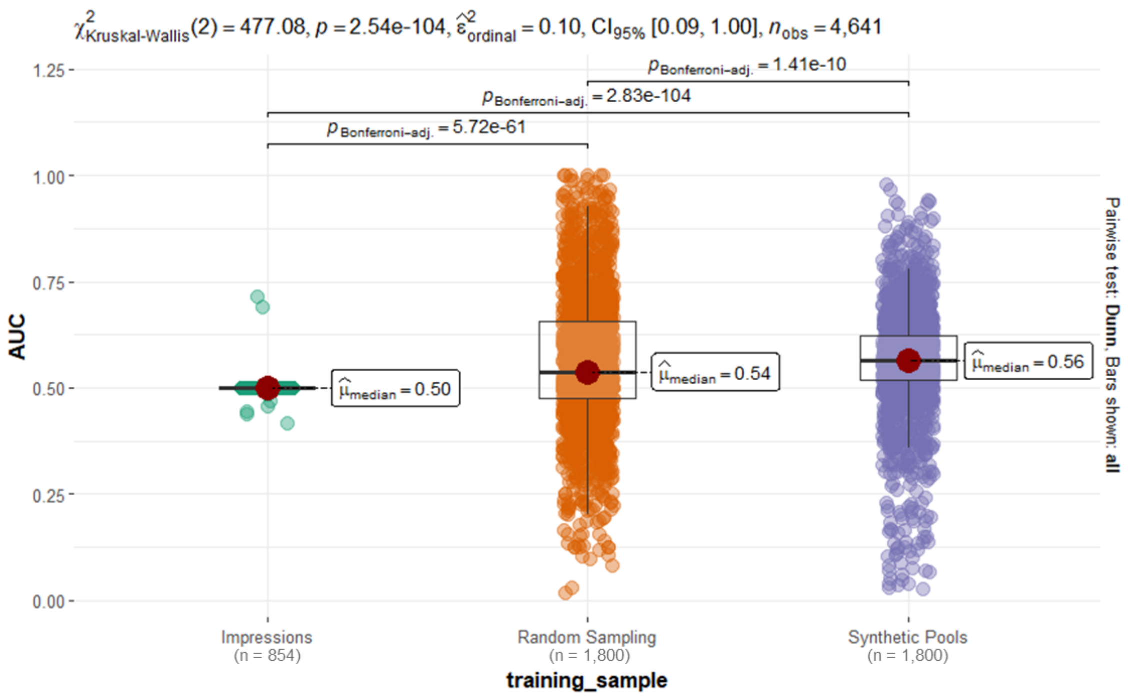


FIGURE 4.9: Influence of the Synthetic Pool: Individual AUC comparison

it is expected that the latter sample would need a simpler model to attain the same accuracy, when compared with the other samples. This was the rationale behind the synthetic negative sampling technique.

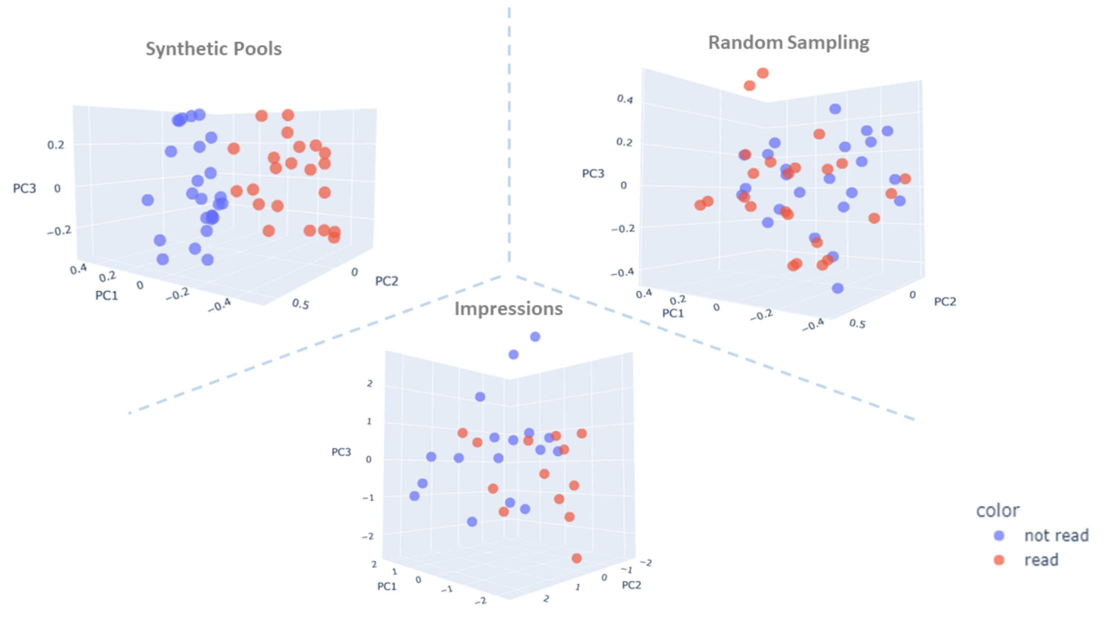


FIGURE 4.10: PCA of the Synthetic Pool VS Random negative sampling VS Impressions

These results expose how data quality imposes limits on ML models, since the only thing that changes is the characteristics of the training sets.

4.1.4 Offline Performance Evaluation

To measure the effectiveness of the proposed approach, the small networks were trained on the Synthetic Pools and tested on samples of impressions. This was compared to some of the most recent SoA news recommendation methods, the NRMS [40], the DKN [39], and the NPA [37]. As seen before, the Synthetic Pools have a great impact when modelling small lightweight networks to capture user reading profiles, but the next question is whether or not it can compete with recent SoA models. Figure 4.11 contains the mean values for AUC for each approach [60]. The results show that this DNNR approach can rival the best models to date, achieving a very similar performance.

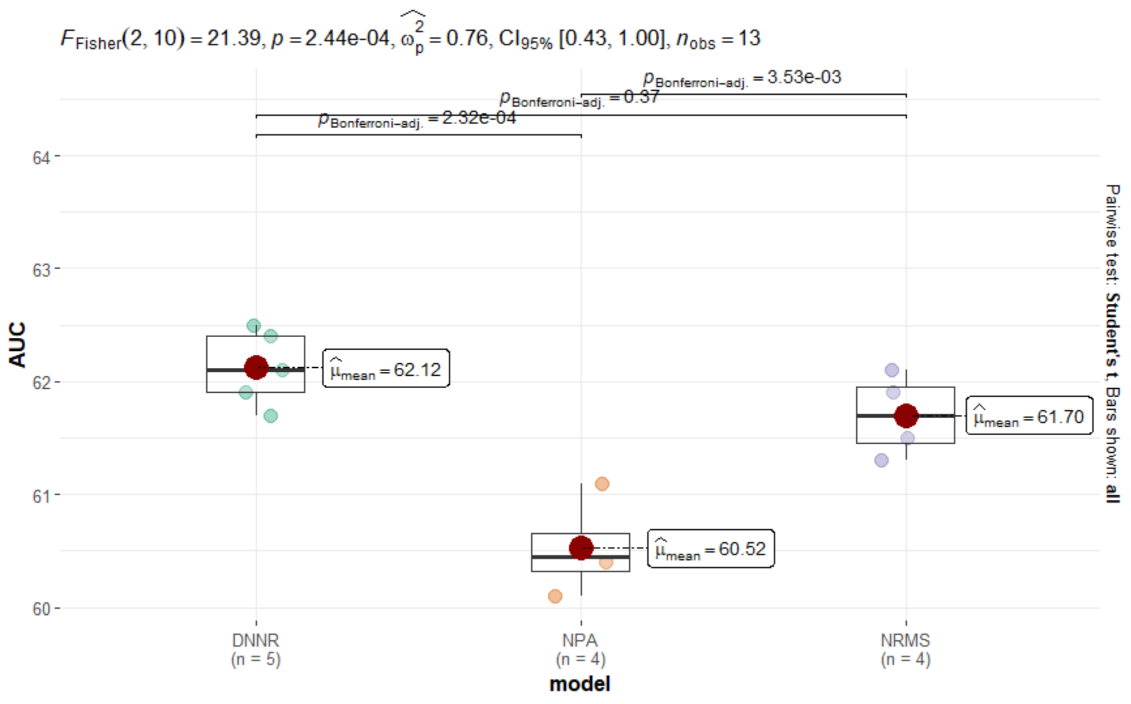


FIGURE 4.11: Comparison of AUC scores between the proposed approach against SoA Models

These results are explained by the singular characteristics of the proposed approach. First of all, PLMs are stronger than the shallow models learned from scratch at text modeling, capturing semantic relationships [41, 61]. We've chosen a multilingual PLM, the Multilingual MiniLM, which uses the same tokenizer as XLM-R but the transformer architecture is the same as BERT, which maps sentences to a 384 dimensional dense vector space for tasks like clustering or semantic search [58, 62]. This was chosen due to having one of the best performance-speed relationship, and because multilingual models tend to outperform monolingual ones, which is a major advantage if we want to jointly train models to serve users in different languages [41]. Second, since each user has its own network, all of the optimized weights and biases in the network were trained only on data directly related to a single user. Finally, the Synthetic Pools used to train the small individual networks make it easy to model each user's profile, requiring less from the model's architecture, since the quality of the data that is fed into the model is better (see figure 4.10).

4.1.5 Speed-Accuracy trade-off

One key aspect to think about when designing a high throughput system that can infer on large amounts of data within a realistic amount of time is the computational time and its trade-off relationship with accuracy (here the word 'accuracy' is being used in the broader spectrum of the term). Although the design of the system allows for more clever implementations to achieve better computational times (see section 4.2), it should also be able to predict in bulk efficiently, if needed. As such, we need to consider the impact of the number of max positive samples per client on the computational times and the final AUC scores. As for the computational times, to get a more precise look at what is happening, we present the time it took to generate the Synthetic Pools (Synthetic Pooling time) and the time it took to generate the predictions.

Figure 4.12 showcases these metrics for four different max sample sizes - 15, 30, 60 and 120 - and the resulting times and group AUC. The line at the bottom is the worst contender, 15 max samples per client. Although it is quicker to compute, we've already seen that raising the number to just 30 can significantly improve the resulting group AUC. Looking at the next contenders, it becomes clear that the choice must be made between 30 and 60, since when choosing 120 as the max number of samples per client, the computational times are raised but with a decrease in final group AUC (although not significant).

Earlier it was argued that "the clear choice seems to be 60 samples, since it has a statistically higher group AUC when compared to both 30 ($p_{Bonferroni-adjusted} = 3.26e-05$) and 120 max positive samples although not statistically significant". Setting aside the correctness of the numbers, it does not mean that the choice is clear. Although we can argue that there is a statistically significant difference, it is not that pronounced, and when looking

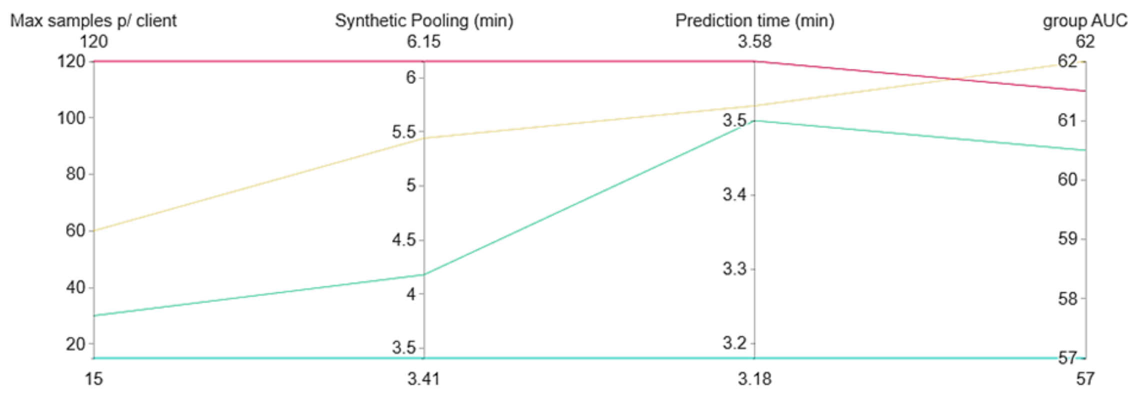


FIGURE 4.12: Comparison of AUC scores resulting from different max number of samples considered per client, and the corresponding Synthetic Pooling and prediction times (in minutes per 4000 clients). Starting with the max sample size per reader, this shows its effect on the time it takes to compute the Synthetic Pools as well as the toll it takes in the time needed to predict, and finally its effect on group AUC. Each max sample size cap has its own color.

at the computational times, depending on the scenario (number of users and the computational power available), one could argue that using 30 max positive samples per client would be the wisest choice. It is noticeable that the prediction time is not very much affected, but the biggest tole is observed in the Synthetic Pooling process. Indeed, this is the most challenging part of the data preparation. On the other hand, due to the architecture of the hole system, these synthetic Pools can be pre-processed before any prediction might be needed, since it only looks at past activities. As such, the Synthetic Pooling routine can be scheduled to run in the previous day, and then we only need to account for the prediction time required from the moment we have novel news to score and the moment we are ready to send recommendations. In this case, maitaining the computational power used during research, it would be need around 88 minutes to predict for 100000 clients when considering a max number of 60 positive samples per user and around 87 minutes to predict for 100000 clients when considering a max number of 30 positive samples per user. So, for 100000 clients, the difference in computational time would be around 1 minute, not a lot.

4.1.6 Variable importance

Since we are defining a custom feed-forward neural network class and the whole process is done using tensors, with custom tensor loaders for faster iterations, not dataframes, most packages that analyse sample input and result output to inform about variable weights do not work here. Instead, we have opted for a more 'statistical' approach, by measuring the individual AUC scores when using different variables available. These are: the news type, the news category and the news title embeddings. Some relevant variable combinations were tested, namely: the embeddings only (Emb), the type and category without the embeddings (T&C), the embeddings with category (Emb&C), the embeddings with type (Emb&T), and all the variables combined (Emb&T&C).

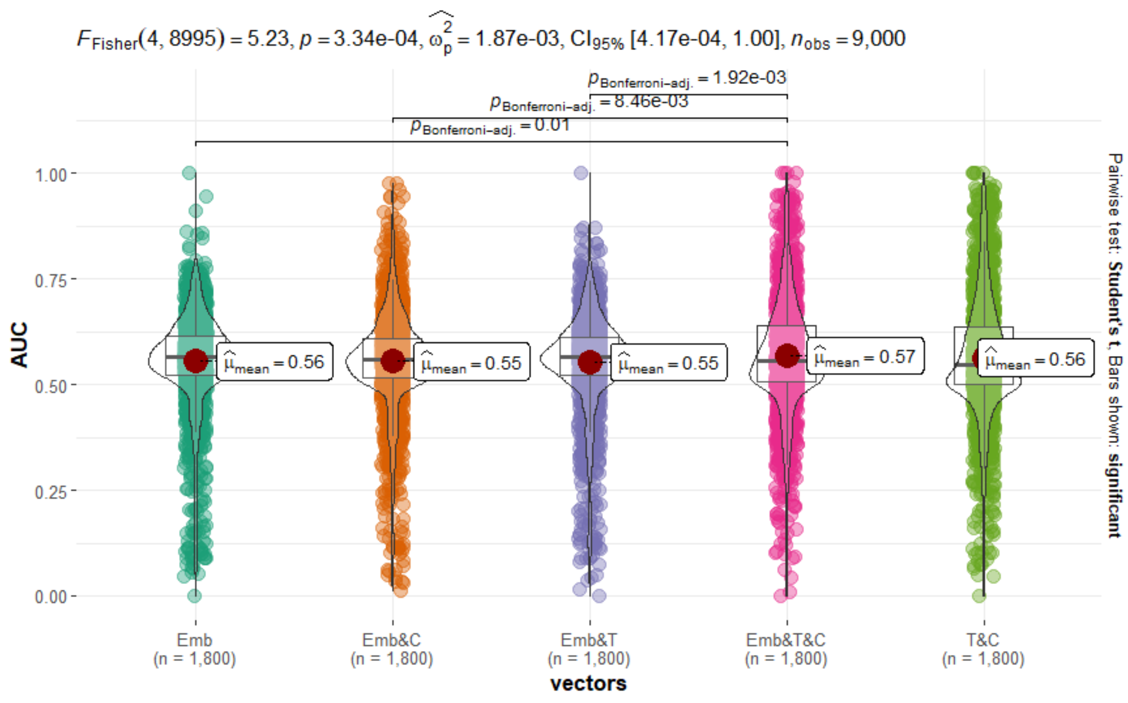


FIGURE 4.13: Variable effect on individual AUC

Some eye-opening results can be seen in Figure 4.13, where the distributions are displayed along the corresponding box-plot and violin-plot. Student's *t* was performed for pairwise *post-hoc* testing. Most differences are not statistically significant except for three: Emb and Emb&T&C ($p_{Bonferroni-adjusted} = 0.01$), Emb&C and Emb&T&C ($p_{Bonferroni-adjusted} = 8.46e-03$), and Emb&T and Emb&T&C ($p_{Bonferroni-adjusted} = 1.92e-03$). All the differences detected were between using all of the variables combined against using only the title embeddings or these combined with either news category or news type. What is more surprising is that by only using the type and the category information from news, the results come so close to the model using all the variables that there is no statistical significant difference between them. But since using all the variables achieves higher mean

AUC, the variance is lower (0.0176 vs 0.0209) and the fact that the title embeddings were already computed for the synthetic negative sampling technique, the choice became clear.

4.2 Discussion

Our results can be interpreted from two perspectives, bringing different insights. The first comes directly from the results. The second considers the implications of using a decentralized model.

4.2.1 Predictive ability

In terms of predictive ability, our proposal achieves competitive results when compared to other SoA news recommenders. Another important observation, but tightly connected to the MIND dataset, is that the negative item sampling strategy is key to achieve better performance. We have used a strategy that maximizes the distance of negative items to the observed users' preferences. Changing this strategy to random sampling drastically reduces performance. Moreover, using the negative indicators – the non-clicked items in the impressions list – from the dataset has the poorest performance, possibly because these new items already come from a recommended list, so they are likely relevant to the users, just somewhat less relevant than the clicked ones.

4.2.2 From a big monolithic networks to small user-based networks

The second perspective is related to the decentralized nature of the proposed method. According to Cisco's Annual Internet Report [63], networked devices around the globe will total 29.3 thousand million in 2023, outnumbering humans by more than three to one, when in 2018 there were 18.4 thousand million. This while global mobile data traffic is expected to continue to grow by nearly 11-fold in the next four years [64]. To ensure higher quality of services, the development is focused on intelligence, personalization and integrations. Along side this, a novel computing paradigm emerges, Mobile edge computing (MEC), that ought to push computation resources from the remote provider servers to the network user devices [65]. This does not mean that MEC is independent from centralized computing power, but rather becomes a supplement [53]. Machine learning models have already been deployed at user devices to be more agile, analyse heterogeneous data, and follow the privacy-policy strategies of computation tasks [66–68]. Therefore, this paradigm is expected to solve the issues of current RS [65]. Under this decentralized MEC network, partial recommendation algorithms from RS can be deployed at user devices, where they can perform data sources caching, aggregation and lightweight processing to remove pressure from the company servers [69], and if recommendation results are outputs from the partial algorithm, this eliminates the latency of data transmission between cloud servers and devices, having a direct impact on customer experience as well.

Recommendation models usually collect all information from users and items into cloud servers to then process them and meet users' demands and interests. However, this conventional approach cannot keep up with the scale and requirements of IoT services. Plus, it does not allow for privacy centered solutions and services. Hence, by deploying decentralized recommendation models to the mobile edge devices to perform some lightweight processing can dramatically improve the quality of service [65]. RPME seeks to relieve the pressure of cloud computing platforms by dynamically placing replicas of movie recommendations [70]. An on-device intelligent cache was also proposed by Weisheng He et al.,[71], where the RS could predict which media files should be cached on-device to reduce latency.

Another problem that RS face is that, to further improve the precision/recall, they become extremely complex and require dramatic computation costs [65]. This cannot cope with the case of IoT, neither can they be deployed to the resource-constrained user devices. Although the deployment of RS on the mobile edge devices is the trend to provide context-aware services, how to offload the partial distributed RS to devices is an open issue.

Our method consists of two stages. The first stage is the computation of embeddings from news titles and keywords, which is done centrally. Note that this computation is purely content-based, so it does not require any kind of user data. Then a simple and fast neural network is trained *for each user*, which, again, does not require data to leave the user device(s). Instead, it brings the algorithmic decisions into the user realm. The advantages are three-fold. First, privacy is improved, since no user data is exchanged or stored centrally. Second, user agency is augmented, since users can decide which algorithms to use and how to tune them, at least in theory. Since personal models have lightweight training and inference pipelines, they can run on low-capacity platforms such as mobile phones or web browsers. This may open interesting possibilities in terms of the actual business model involving recommendations. Finally, the decentralized paradigm also brings interesting advantages from the computational perspective, simply because a large share of computation is offloaded to user devices, reducing computational costs and latency, and reducing the dependency on network connectivity.

4.2.3 Limitations

Although our proposal does not strictly require centralized training, we recognize that it is impractical, in most real-world scenarios, to rely exclusively in decentralized computation. The computation of item embeddings, for example, requires access to a very large number of content items, and is too resource-demanding to run in user devices. We note that in the scenario of central computation of embeddings, privacy benefits are maintained, since users are not required to share their history with the provider. It does, however, reduce user agency, since users cannot compute their own item representations. Also regarding the decentralized approach, we note that this is highly facilitated when using a purely content-based method. Collaborative approaches are much harder to decentralize, because they fundamentally rely on personal data from multiple users.

Another thing to keep in mind is that, although this approach can drastically simplify the neural network architectures typically used for these applications, it cannot beat them and it is not perfect. More complex neural architectures should be explored in the future to push the limits of what this training framework can achieve, without compromising its flexibility and easiness to read, implement, maintain and interpret. Indeed, the balance between complexity and performance is a fine line.

At last, despite the customization power of the proposed approach, people without consolidated reading habits will still receive personalized recommendations based on sparse and seldom past interactions. These interactions can be very old, few and far between. So, even though there is enough information to base the recommendations upon, there is less certainty to the modeled preferences. This might not be ideal if the purpose is to capture the attention of disloyal readers or newcomers. For that, a trending algorithm would probably be more effective than any other alternative, before we could start serving personalized communications.

4.2.4 On model deployment

To complete this project, the final model was packaged into a high performance API, designed for speed, flexibility and reliability. The API was constructed using FastAPI, a modern web framework for building APIs with Python, which is production-ready and automatically generates interactive documentation (see Figure 2.8). Internal testing showed that the final underlying model manages to complete the full recommendation task in $1m15s$ for 1000 users ($0.075s$ per reader), and the API takes just 0.16 seconds to respond to a user query. Even though the final implementation does not leverage the full potential of the approach, specifically when it comes to the decentralization, its flexibility allowed us to break the computational time into smaller pieces, precomputing some parts to shave some time for when the magic needs to happen really fast. And it does!

Chapter 5

Conclusion

In this thesis, we propose a decentralized neural NRS approach that explores other less prominent facets of the news recommendation environment. Namely, using the MIND dataset, we argue that a different approach to negative sampling can change the way model architectures are designed and improve model training without the need of adding additional layers of complexity to already highly complex neural architectures. In fact, if even random sampling is better than relying solely on impressions alone, the negative sampling technique creates an easier environment for models to efficiently learn user profiles. By having lightweight individual models for each user, this also opens up the possibility to take the recommendation process from the provider's central computing power on to the user's devices, enabling on-device learning. This brings three major benefits: i) reducing the computational cost associated with training pipelines of models on massive amounts of data; ii) the possibility to train profiles without having to transfer private information between users devices and the central computing infrastructure; and iii) the offloading of computation from a centralized infrastructure to user devices in a manageable amount, reducing costs, network dependability and latency. In future efforts we will seek to improve our approach in several important directions. First, we would like to study the trade-off between model complexity and computation time, taking into account the typical computational constraints of user devices. Second, we will study the integration with a session-based collaborative filtering component, without compromising the benefits of the decentralized approach. Third, a trending algorithm would be a good alternative for clients whose interactions are few and far between. Finally, an online test would be key to measure the performance of these techniques in a real news recommendation scenario.

Bibliography

[1] N. Newman, D. A. L. Levy, and R. K. Nielsen, "Reuters Institute Digital News Report 2015," *SSRN Electronic Journal*, 2015. [Cited on page 5.]

[2] C. C. Aggarwal, *Recommender Systems*. Cham: Springer International Publishing, 2016. [Cited on pages 6, 8, and 10.]

[3] H. Ko, S. Lee, Y. Park, and A. Choi, "A Survey of Recommendation Systems: Recommendation Models, Techniques, and Application Fields," *Electronics*, vol. 11, no. 1, p. 141, Jan. 2022. [Cited on pages 6, 8, 9, and 10.]

[4] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative Filtering for Implicit Feedback Datasets," in *2008 Eighth IEEE International Conference on Data Mining*. Pisa, Italy: IEEE, 2008, pp. 263–272. [Cited on pages 6 and 10.]

[5] K. Balog and F. Radlinski, "Measuring recommendation explanation quality: The conflicting goals of explanations," *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020. [Cited on pages 6 and 7.]

[6] D. M. Fleder and K. Hosanagar, "Recommender systems and their impact on sales diversity," in *Proceedings of the 8th ACM Conference on Electronic Commerce - EC '07*. San Diego, California, USA: ACM Press, 2007, p. 192. [Cited on page 6.]

[7] N. Good, J. B. Schafer, J. A. Konstan, A. Borchers, B. Sarwar, J. Herlocker, and J. Riedl, "Combining collaborative filtering with personal agents for better recommendations," in *Combining Collaborative Filtering with Personal Agents for Better Recommendations*, ser. AAAI '99/IAAI '99. Orlando, Florida, USA: American Association for Artificial Intelligence, 1999, pp. 439–446. [Cited on pages 7 and 8.]

[8] M. Srifi, A. Oussous, A. Ait Lahcen, and S. Mouline, "Recommender Systems Based on Collaborative Filtering Using Review Texts—A Survey," *Information*, vol. 11, no. 6, p. 317, 2020. [Cited on page 7.]

[9] F. Lorenzi and F. Ricci, "Case-Based Recommender Systems: A Unifying View," in *Intelligent Techniques for Web Personalization*, B. Mobasher and S. S. Anand, Eds. Springer Berlin Heidelberg, 2005, pp. 89–113. [Cited on pages 7 and 8.]

[10] P. Lops, D. Jannach, C. Musto, T. Bogers, and M. Koolen, "Trends in content-based recommendation," *User Modeling and User-Adapted Interaction*, vol. 29, pp. 239–249, 2019. [Cited on pages 7 and 8.]

[11] R. Burke, "Hybrid recommender systems: Survey and experiments," *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331–370, 2002. [Cited on pages 7, 8, and 9.]

[12] R. Forsati, H. M. Doustdar, M. Shamsfard, A. Keikha, and M. R. Meybodi, "A fuzzy co-clustering approach for hybrid recommender systems," *Int. J. Hybrid Intell. Syst.*, vol. 10, no. 2, pp. 71–81, Apr. 2013. [Cited on pages 8 and 9.]

[13] M. Â. Rebelo, D. Coelho, I. Pereira, and F. Fernandes, "A new cascade-hybrid recommender system approach for the retail market," in *Innovations in Bio-Inspired Computing and Applications*, A. Abraham, A. M. Madureira, A. Kaklauskas, N. Gandhi, A. Bajaj, A. K. Muda, D. Kriksciuniene, and J. C. Ferreira, Eds. Cham: Springer International Publishing, 2022, pp. 371–380. [Cited on pages 7 and 9.]

[14] S. D. Raj, "Automated service recommendation with preference awareness: An application of colaborative filtering approach in big data analytics," *2017 2nd International Conference on Communication and Electronics Systems (ICCES)*, pp. 766–769, 2017. [Cited on page 7.]

[15] P. Aditya, I. Budi, and Q. Munajat, "A comparative analysis of memory-based and model-based collaborative filtering on the implementation of recommender system for E-commerce in Indonesia: A case study," in *2016 International Conference on Advanced Computer Science and Information Systems*. Malang, Indonesia: IEEE, 2016, pp. 303–308. [Cited on page 7.]

[16] P. H. Aditya, I. Budi, and Q. Munajat, "A comparative analysis of memory-based and model-based collaborative filtering on the implementation of recommender system for E-commerce in Indonesia: A case study PT X," in *2016 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*. Malang, Indonesia: IEEE, Oct. 2016, pp. 303–308. [Cited on page 7.]

[17] Y. Koren, "Factor in the neighbors: Scalable and accurate collaborative filtering," *ACM Transactions on Knowledge Discovery from Data*, vol. 4, no. 1, pp. 1–24, 2010. [Cited on page 8.]

[18] P. Lops, M. de Gemmis, and G. Semeraro, "Content-based Recommender Systems: State of the Art and Trends," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. Boston, MA: Springer US, 2011, pp. 73–105. [Cited on page 8.]

[19] B. Walek and V. Fojtik, "A hybrid recommender system for recommending relevant movies using an expert system," *Expert Systems With Applications*, vol. 158, p. 113452, 2020. [Cited on page 9.]

[20] E. Çano and M. Morisio, "Hybrid recommender systems: A systematic literature review," *Intelligent Data Analysis*, vol. 21, pp. 1487–1524, 2017. [Cited on pages 8 and 9.]

[21] P. Funk, Ed., *Advances in Case-Based Reasoning: 7th European Conference, ECCBR 2004, Madrid, Spain, August 30 - September 2, 2004; Proceedings*, ser. Lecture Notes in Computer Science Lecture Notes in Artificial Intelligence. Berlin Heidelberg: Springer, 2004, no. 3155. [Cited on page 9.]

[22] K. Tas, E. Onder, and M. S. Aktas, "On the Implicit Feedback Based Data Modeling Approaches for Recommendation Systems," in *2021 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*. Kuala Lumpur, Malaysia: IEEE, Jun. 2021, pp. 1–6. [Cited on page 10.]

[23] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A Constant Time Collaborative Filtering Algorithm," *Information Retrieval*, vol. 4, no. 2, pp. 133–151, Jul. 2001. [Cited on page 10.]

[24] M. A. Hart, "The long tail: Why the future of business is selling less of more by chris anderson," *Journal of Product Innovation Management*, vol. 24, no. 3, pp. 274–276, 2007. [Cited on page 10.]

[25] H. Yin, B. Cui, J. Li, J. Yao, and C. Chen, "Challenging the long tail recommendation," *Proc. VLDB Endow.*, vol. 5, no. 9, pp. 896–907, May 2012. [Cited on page 10.]

[26] Y.-J. Park and A. Tuzhilin, "The long tail of recommender systems and how to leverage it," in *Proceedings of the 2008 ACM Conference on Recommender Systems - RecSys '08*. Lausanne, Switzerland: ACM Press, 2008, p. 11. [Cited on page 10.]

[27] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-n recommendation tasks," in *Proceedings of the Fourth ACM Conference on Recommender Systems*, ser. RecSys '10. Barcelona, Spain: Association for Computing Machinery, 2010, pp. 39–46. [Cited on page 10.]

[28] M. Karimi, D. Jannach, and M. Jugovac, "News recommender systems – Survey and roads ahead," *Information Processing & Management*, vol. 54, no. 6, pp. 1203–1227, Nov. 2018. [Cited on pages 11, 12, 13, and 14.]

[29] D. Jannach, M. Zanker, M. Ge, and M. Gröning, "Recommender Systems in Computer Science and Information Systems – A Landscape of Research," in *E-Commerce and Web Technologies*, W. van der Aalst, J. Mylopoulos, M. Rosemann, M. J. Shaw, C. Szyperski, C. Huemer, and P. Lops, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, vol. 123, pp. 76–87. [Cited on page 12.]

[30] W. IJntema, F. Goossen, F. Frasincar, and F. Hogenboom, "Ontology-based news recommendation," in *Proceedings of the 1st International Workshop on Data Semantics - DataSem '10*. Lausanne, Switzerland: ACM Press, 2010, p. 1. [Cited on page 12.]

[31] J. Rao, A. Jia, Y. Feng, and D. Zhao, "Personalized News Recommendation Using Ontologies Harvested from the Web," in *Web-Age Information Management*, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, J. Wang, H. Xiong, Y. Ishikawa, J. Xu, and J. Zhou, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, vol. 7923, pp. 781–787. [Cited on page 12.]

[32] H. Lee and S. J. Park, "MONERS: A news recommender for the mobile web," *Expert Systems with Applications*, vol. 32, no. 1, pp. 143–150, 2007. [Cited on page 12.]

[33] J.-w. Ahn, P. Brusilovsky, J. Grady, D. He, and S. Y. Syn, "Open user profiles for adaptive news systems: Help or harm?" in *Proceedings of the 16th International Conference on World Wide Web*, ser. WWW '07. New York, NY, USA: Association for Computing Machinery, 2007, pp. 11–20. [Cited on pages 12 and 13.]

[34] A. S. Das, M. Datar, A. Garg, and S. Rajaram, "Google news personalization: Scalable online collaborative filtering," in *Proceedings of the 16th International Conference on World Wide Web*, ser. WWW '07. New York, NY, USA: Association for Computing Machinery, 2007, pp. 271–280. [Cited on page 12.]

[35] J. Liu, P. Dolan, and E. R. Pedersen, "Personalized news recommendation based on click behavior," in *Proceedings of the 15th International Conference on Intelligent User Interfaces - IUI '10*. Hong Kong, China: ACM Press, 2010, p. 31. [Cited on page 12.]

[36] Vietnam National University — University of Engineering and Technology, Hanoi, Vietnam., N. T. Huy, D. T. Chuan, and V. A. Nguyen, "Implicit Feedback Mechanism to Manage User Profile Applied in Vietnamese News Recommender System," *International Journal of Computer and Communication Engineering*, vol. 5, no. 4, pp. 276–285, 2015. [Cited on page 13.]

[37] C. Wu, F. Wu, M. An, J. Huang, Y. Huang, and X. Xie, "NPA: Neural News Recommendation with Personalized Attention," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Anchorage AK USA: ACM, Jul. 2019, pp. 2576–2584. [Cited on pages 13, 14, and 45.]

[38] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6000–6010. [Cited on pages xv, 13, 20, 21, 22, 23, 24, and 25.]

[39] H. Wang, F. Zhang, X. Xie, and M. Guo, "DKN: Deep Knowledge-Aware Network for News Recommendation," in *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18*. Lyon, France: ACM Press, 2018, pp. 1835–1844. [Cited on pages 13, 14, and 45.]

[40] C. Wu, F. Wu, S. Ge, T. Qi, Y. Huang, and X. Xie, "Neural News Recommendation with Multi-Head Self-Attention," in *Proceedings of the 2019 Conference on Empirical*

Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Hong Kong, China: Association for Computational Linguistics, 2019, pp. 6388–6393. [Cited on pages 13, 14, 35, and 45.]

[41] C. Wu, F. Wu, T. Qi, and Y. Huang, “Empowering News Recommendation with Pre-trained Language Models,” *arXiv:2104.07413 [cs]*, Apr. 2021. [Cited on pages 13, 45, and 46.]

[42] C. Wu, F. Wu, Y. Huang, and X. Xie, “Neural news recommendation with negative feedback,” *CCF Transactions on Pervasive Computing and Interaction*, vol. 2, no. 3, pp. 178–188, Oct. 2020. [Cited on pages 13 and 14.]

[43] W.-B. Feng, Z.-S. Zhang, C.-Z. Li, and A.-P. Song, “Negative examples sampling based on factorization machines for OCCF,” in *2019 6th International Conference on Systems and Informatics (ICSAI)*, 2019, pp. 1482–1485. [Cited on page 14.]

[44] J.-c. Zhang and Y. Takama, “Proposal of context-aware music recommender system using negative sampling,” in *Advances in Artificial Intelligence*, Y. Ohsawa, K. Yada, T. Ito, Y. Takama, E. Sato-Shimokawara, A. Abe, J. Mori, and N. Matsumura, Eds. Cham: Springer International Publishing, 2020, pp. 114–125. [Cited on page 14.]

[45] J. a. Vinagre, A. M. Jorge, and J. a. Gama, “Collaborative filtering with recency-based negative feedback,” in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, ser. SAC ’15. New York, NY, USA: Association for Computing Machinery, 2015, p. 963–965. [Online]. Available: https://doi.org/10.1145/2695664.2695998 [Cited on page 14.]

[46] S. Okura, Y. Tagami, S. Ono, and A. Tajima, “Embedding-based News Recommendation for Millions of Users,” in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Halifax NS Canada: ACM, Aug. 2017, pp. 1933–1942. [Cited on page 14.]

[47] H. Wang, F. Wu, Z. Liu, and X. Xie, “Fine-grained interest matching for neural news recommendation,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 836–845. [Cited on page 14.]

[48] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, “Pre-trained Models for Natural Language Processing: A Survey,” *Science China Technological Sciences*, vol. 63, no. 10, pp. 1872–1897, Oct. 2020. [Cited on page 15.]

[49] J. Canny, “Collaborative filtering with privacy via factor analysis,” in *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR ’02. New York, NY, USA: Association for Computing Machinery, 2002, p. 238–245. [Online]. Available: https://doi.org/10.1145/564376.564419 [Cited on page 15.]

[50] P. Han, B. Xie, F. Yang, and R. Shen, “A scalable p2p recommender system based on distributed collaborative filtering,” *Expert Systems with Applications*, vol. 27, no. 2, pp. 203–210, 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417404000065

[51] B. N. Miller, J. A. Konstan, and J. Riedl, “Pocketlens: Toward a personal recommender system,” *ACM Trans. Inf. Syst.*, vol. 22, no. 3, p. 437–476, jul 2004. [Online]. Available: https://doi.org/10.1145/1010614.1010618 [Cited on page 15.]

[52] T. Eichinger, F. Beierle, R. Papke, L. Rebscher, H. C. Tran, and M. Trzeciak, “On gossip-based information dissemination in pervasive recommender systems,” in *Proceedings of the 13th ACM Conference on Recommender Systems*, ser. RecSys ’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 442–446. [Online]. Available: https://doi.org/10.1145/3298689.3347067 [Cited on page 15.]

[53] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, “Convergence of Edge Computing and Deep Learning: A Comprehensive Survey,” *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869–904, 2020. [Cited on pages 15, 16, and 49.]

[54] J. Nuno Tavares, “Module 7: Crash course in neural networks.” in *Statistical Learning*. DM-FCUP: PG-Computational Statistics And Data Analysis, 2021. [Cited on pages xv, 16, 17, 18, and 19.]

[55] D. Livingstone, D. Manallack, and I. Tetko, “Data modelling with neural networks: Advantages and limitations,” *Journal of Computer-Aided Molecular Design*, vol. 11, no. 2, pp. 135–142, Mar. 1997. [Cited on page 19.]

[56] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew, "Huggingface's transformers: State-of-the-art natural language processing," *ArXiv*, vol. abs/1910.03771, 2019. [Cited on pages 25 and 26.]

[57] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *ArXiv*, vol. abs/1207.0580, 2012. [Cited on page 30.]

[58] W. Wang, F. Wei, L. Dong, N. Yang, and M. Zhou, "MINILM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers," p. 13, 2020. [Cited on pages 31 and 46.]

[59] F. Wu, Y. Qiao, J.-H. Chen, C. Wu, T. Qi, J. Lian, D. Liu, X. Xie, J. Gao, W. Wu, and M. Zhou, "MIND: A Large-scale Dataset for News Recommendation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, 2020, pp. 3597–3606. [Cited on pages 33 and 35.]

[60] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, pp. 1145–1159, 1997. [Cited on page 45.]

[61] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Cited on page 45.]

[62] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, 2019, pp. 3980–3990. [Cited on page 46.]

[63] Cisco, "Cisco Annual Internet Report (2018–2023)," 2020. [Cited on page 49.]

[64] A. Tongaonkar, "A look at the mobile app identification landscape," *IEEE Internet Computing*, vol. 20, no. 4, pp. 9–15, 2016. [Cited on page 49.]

[65] C. Sun, H. Li, X. Li, J. Wen, Q. Xiong, and W. Zhou, "Convergence of recommender systems and edge computing: A comprehensive survey," *IEEE access : practical innovations, open solutions*, vol. 8, pp. 47 118–47 132, 2020. [Cited on pages 49 and 50.]

[66] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-Edge AI: Intelligentizing Mobile Edge Computing, Caching and Communication by Federated Learning," *IEEE Network*, vol. 33, no. 5, pp. 156–165, Sep. 2019. [Cited on page 49.]

[67] B. Chen, J. Wan, A. Celesti, D. Li, H. Abbas, and Q. Zhang, "Edge Computing in IoT-Based Manufacturing," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 103–109, Sep. 2018.

[68] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge AI: On-Demand Accelerating Deep Neural Network Inference via Edge Computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 447–457, Jan. 2020. [Cited on page 49.]

[69] Y. Cao, O. Kaiwartya, Y. Zhuang, N. Ahmad, Y. Sun, and J. Lloret, "A decentralized deadline-driven electric vehicle charging recommendation," *IEEE Systems Journal*, vol. 13, no. 3, pp. 3410–3421, 2019. [Cited on page 50.]

[70] Y. Tang, H. Wang, K. Guo, T. Luo, and T. Chi, "A new replica placement mechanism for mobile media streaming in edge computing," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 7, p. e5361, 2021. [Cited on page 50.]

[71] W. He, Y. Su, X. Xu, Z. Luo, L. Huang, and X. Du, "Cooperative content caching for mobile edge computing with network coding," *IEEE access : practical innovations, open solutions*, vol. 7, pp. 67 695–67 707, 2019. [Cited on page 50.]