Graph-based Methods Applied to the Relevance Ranking Problem in eCommerce Search

By

Isabel Pires Chaves

Master Thesis in Modeling, Data Analysis
and Decision Support Systems

Supervised by:

Professor Dr. João Gama

Dr. Pedro Balage

Faculdade de Economia

Universidade do Porto

2020

"We keep moving forward, opening new doors,
and doing new things, because we're curious
and curiosity keeps leading us down new paths."
– Walt Disney

Biographical note

Isabel Pires Chaves was born on November 21st 1995, in Matosinhos, Portugal.

While working on her Bachelor degree in *Engenharia de Computação e Instrumentação Médica* by *Instituto Superior de Engenharia do Porto (ISEP)*, Isabel started to feel a growing interest in data analysis and processing. During the final work of her Bachelor, Isabel had the opportunity of doing an internship as Signal Processing Engineer. During this internship, she was responsible for designing a data pipeline which filtered and compressed the original signal, producing viable data for posterior analysis of heartbeat signals.

All of the previous experiences led to a growing will of wanting to know more about the world of Data Science. Therefore, after accomplishing her Bachelor in 2017, with the intent of putting herself to the test, Isabel applied for another internship, but this time as a Data Scientist trainee at Farfetch, an e-Commerce Luxury platform. After finishing the internship, she continued to work at Farfetch as a Data Scientist, revolutionizing the company ranking systems. With this challenge, she is continually growing her skills on machine learning and information retrieval.

To keep improving her knowledge in the area, Isabel applied to a Masters program in Modeling, Data Analysis and Decision Support Systems at the *Faculdade de Economia* of the *Universidade do Porto*. During this time, she was able to conduct theoretical analysis and develop new skills in areas such as time series analysis, optimization and multi-agent systems.

Besides that, since 2019, Isabel is also part of the Data Science Portugal (DSPT) community, a group of Data Science enthusiasts that organises events and creates content, promoting the Portuguese data science community engagement and growth.

Acknowledgements

Writing a dissertation can present several challenges and difficulties, but when we surround ourselves with the right people, the hardest tasks can become the simpler ones.

First of all, I must thank with my deepest love to my mother and father, who were capable of showing me that nothing is impossible. You taught me that with hard work and commitment, everything can be accomplished and this work is proof of that. I want to acknowledge my sister and especially my niece, that even though she knew her auntie was full of work, she would have time to see her new drawings. I hope that someday you will also conquer great things and I will be there by your side supporting you.

Also, I would like to thank my *mais-que-tudo* for being my rock, English revisor, rubber duck and everything that I needed during the process of researching and writing this thesis.

Furthermore, I want to express my deep gratitude to my supervisors Prof. Dr João Gama and Dr Pedro Balage, for all of the guidance and support provided during this work. Pedro, thank you for all of the Wednesday morning meetings in which I would bother you with all my doubts and questions. Without the assistance, support and critics from both this work would not be as it is now.

Moreover, I also would like to thank all mentors that were part of my journey, academic and professional. I want to thank all the unique human beings and extraordinary professionals that helped me grow and taught me all I know, which allowed me to became the Data Scientist that I am today.

Finally, I would also like to thank all of my friends, old and new, for all of the support. All of the fun late nights at FEP when we had to study and all of the gaming afternoons/nights only for me to not forget how to be a social human being. I promise you guys, I will not use my thesis as an excuse on the next time.

Abstract

The digital world has been growing, and the demand for information transfer has been increasing with its growth. eCommerce customers are becoming more demanding and eager for improved results when using a search engine. They do not want to spend a meaningful amount of their time to understand how to express their intent. Nonetheless, they want to be shown the best products that match their search query.

Information retrieval is an expertise in constant change, there is novelty every day, and the usage of graph relationship retrieved by click-through data for specific tasks can bring a different perspective to an eCommerce catalogue. With this idea in mind, this works presents a click-through graph-based methodology capable of connecting the products of a given catalogue to search queries by creating better representations that reduce the lexical gap existent between search queries and product descriptions.

The main goal of this work is to extend and improve the Jiang et al. (2016) work, described here as the click graph approach. With that objective, the algorithm was combined with several word representation techniques (TF-IDF, Word2Vec, and BERT) in an attempt to obtain a better and improved network representation learning approach, capable of being applied to eCommerce search engines. Moreover, an extensive experimental analysis was conducted with the intent of understanding the quality of the proposed solution. The evaluation performed took into consideration precision and rankings metrics, which allowed the analysis of not only the accuracy of the retrieved items but also their ranking. The obtained results showed that this type of graph-based approaches needs highly connected data to achieve better performance. By showing the trade-offs of each of the word representation methods used, and also the drawbacks of this combination, it opened paths for future researchers to explore.

Keywords: Knowledge Graphs, Bipartite Graphs, Representation Learning, Search Engines, eCommerce, Information Retrieval

Resumo

O mundo digital tem vindo a evoluir e a procura de transferência de informação tem vindo a aumentar com o seu crescimento. Os clientes de plataformas de comércio eletrónico são cada vez mais exigentes e ávidos por melhores resultados quando utilizam um motor de pesquisa. Estes utilizadores não querem gastar uma quantidade significativa do seu tempo à procura dos termos que melhor expressem a sua intenção relativamente à pesquisa desejada. No entanto, estes clientes esperam sempre que os resultados obtidos correspondam inteiramente ao que eles procuram.

A recuperação de informação é uma área que se encontra em desenvolvimento constante e novos métodos e inovações são apresentados regularmente. O uso de grafos para algumas tarefas em específico são capazes de revolucionar a forma como o catálogo de um sistema de *eCommerce* está conectado. Com o intuito de revolucionar a forma como a pesquisa é realizada num motor de busca, este trabalho apresenta uma metodologia baseada em grafos, capaz de conectar produtos com os termos utlizados para a pesquisa através da informação de cliques previamente armazenada. Com isto é pretendido obter uma redução do campo lexical, pois as representações textuais de ambas as componentes será aproximada.

O principal objetivo desde trabalho é estender e melhorar o trabalho realizado por Jiang et al. (2016), denominado de *click graph* ao longo deste documento. Com o intuito de criar uma melhor abordagem e aprimorar a aprendizagem da representação em rede, este algoritmo foi combinado com diversas técnicas de processamento de linguagem natural (TF-IDF, Word2Vec e BERT), de forma a poder ser utilizado num motor de pesquisa de comércio eletrónico. As metodologias de avaliação tiveram em conta métricas de precisão e ordenação dos resultados, o que permitiu uma análise aprofundada da precisão dos resultados, assim como do *ranking* obtido. Os resultados obtidos permitiram a compreensão de que este tipo de métodos baseados em grafos necessitam de uma extensa coleção de dados altamente conectados, de forma a ser possível obter um melhor desempenho. Ao permitir uma melhor percepção do funcionamento de cada um dos métodos utilizados, assim como clarificar expectativas da combinação dos mesmos, foi possível criar novas oportunidades de investigação.

Palavras-chave: Grafos de Conhecimento, Grafos Bipartidos, Aprendizagem de Representação, Motores de Pesquisa, Comércio Eletrónico, Recuperação de Informação

Contents

List of Figures

List of Tables

Acronyms

AP Average Precision.

BERT Bidirectional Encoder Representations from Transformers.

BFS Breadth-First Search.

BiGIT Bipartite Graph Information Transfer.

BiNE Bipartite Network Embeddings.

CBOW Continuous Bag of Words.

DCG Discounted Cumulative Gain.

DFS Depth-First Search.

ELMo Embeddings from Language Model.

IR Information Retrieval.

LINE Large-scale Information. Network Embedding.

LTR Learning to Rank.

MAP Mean Average Precision.

ML Machine Learning.

MLM Model Language Model.

MRR Mean Reciprocal Rank.

nDCG Normalized Discounted Cumulative Gain.

NLP Natural Language Processing.

NMSLIB Non-Metric Space Library.

NSP Next Sentence Prediction.

RR Reciprocal Rank.

TF-IDF Term Frequency-Inverse Document Frequency.

Chapter 1

Introduction

This chapter has the objective of introducing the reader to this work. It contains the context of the work, the motivation to solve the problem presented, and the objectives defined.

1.1 Context

Human evolution and the digital age revolution have been increasing the demand for information transfer around the globe. The way we communicate and search for information has changed, and there is a sense of urgency to decrease the degree of difficulty of finding a given piece of content within a system, also known as discoverability. In a data-driven era, search engines are the main door to discoverability and are the core of each web interaction (Langville & Meyer, 2011).

Search engines have been growing bigger and more powerful than ever, with more demanding customers, given that it allows the user to express their intents with keywords (Huang et al., 2018). However, not every user can construct an expressive and unambiguous query that translates their exact thoughts, which leads to ambiguity and poor rankings of the retrieved results (R. Li, Li, Wu, Zhou, & Wang, 2019).

The main question in this field is "how can we provide a good output for our customer needs?" given that there is the ultimate goal of retrieving to that person exactly the information that she was looking for when she submitted the query to the search engine (Croft, Metzler, & Strohman, 2009). Taking into consideration that it is not possible to store all the combinations between queries and products since that goes beyond human knowledge, it becomes necessary to create a solution capable of mapping user intents with the desired search output.

Nevertheless, eCommerce products search presents different types of challenges to the main research area, since this type of search engines can generate revenue solely based on the products and how they are shown to the users. One of the challenges is related with the optimization of multiple metrics, since that the list of documents presented to the user should take into consideration metrics that the business is trying to optimize (e.g. revenue, visits) while still maintaining a discovery strategy for the website (Goswami, Zhai, & Mohapatra, 2018).

1.2 Motivation

New challenges for eCommerce search engines appear everyday which motivates scientific advances. Classical methods, such as click modelling and direct text matching, are not robust enough to deal with many of these challenges but the current advances on the state of the art models are leading to novel ways to search interactions.

Direct text matching became an outdated method to deal with search queries. Mostly since document and query language usually present differences from one to another, meaning that there is a semantic gap between the products and the search queries (Müller & Gurevych, 2009). Usually, when users are trying to search for the information they are not careful with the terms used on the query, an example can be the query "How much product X" in which the relevant documents should not contain the term "how much" but rather "price".

Since most of the search queries follow a long tail distribution, click modelling became a method ineffective for these type of challenges since most of the user behavioural information (such as clicks) is not very well represented, due to the skewed distribution.

Another factor is the fact that users started to use search engines as Question and Answering (Q&A) systems that lead to an increasing number of queries done in the natural language style. As a result, new obstacles have appeared that can impact several points of search technology such as query understanding and semantic matching, as stated by D. Yin et al. (2016).

Recent approaches are using technologies based on deep learning networks, such as Brenner, Zhao, Kutiyanawala, and Yan (2018) and Y. Zhang, Wang, and Zhang (2019), but that type of model is still at the beginning, and they are showing a lack of result quality consistency since there is a discrepancy between the obtained metric of frequent queries versus infrequent ones. Therefore, some researchers are still focusing on more straightforward approaches, such as the one presented in Jiang et al. (2016), named here as the click graph algorithm. The authors presented a work capable of modelling queries and products using discrete representations, leading to a very sparse vector space, which possesses many drawbacks for natural language processing (NLP) tasks.

1.3 Objectives

The goal of this work is to study, design, and propose new graph-based methods capable of improving the quality and relevance of the results shown to a user when using a free text search engine. Thus, it is intended to not only use the discrete information of the user query but also to focus on the semantic comprehension solution. To achieve this, a set of experiments will be conducted and compared so that additional value can be delivered to this research field.

With the objective of further investigating graph-based methods, this work intends to look for new ways to represent graph nodes using a continuous representation in a semantic vector space. Consequently, this work has the objective of applying state-of-the-art methods - with focus on representation learning in graphs - to improve the click graph algorithm. The results will then be measured and analysed.

To compare the outcome from this project, some metrics will be used that can demonstrate if the implemented approach can show highly relevant documents to the user or not. Thereby, the normalized Discounted Cumulative Gain (nDCG) (Järvelin & Kekäläinen, 2002),

Mean Reciprocal Rank (MRR) (Craswell, 2009), and others metrics found during the work research will be used with the intent of validating if the obtained results have quality and are relevant for the user and query in question.

Summing up, this work has the following three main objectives:

1. Analyse and implement the method proposed by Jiang et al. (2016), the click graph model.
2. Analyse the behaviour of this method when using non-explored datasets.
3. Analyse the behaviour of this type of model when combined with different word representations approaches.

1.4 Contributions

In summary the contributions of this work are:

1. Bipartite Graph Information Transfer (BiGIT), an open-source implementation of the model proposed by Jiang et al. (2016), the click graph.
2. A comparative analysis of the performance provided by different word representation methods when combined with BiGIT on a search engine task.

1.5 Document Structure

This section has the purpose of providing a small summary of each chapter present in this document. There are six different chapters in total, followed by the references.

1. Introduction - the first chapter introduces the reader to the developed work by presenting the context, motivation, objectives and the document structure.
2. State of the Art - is divided into three components: Background Knowledge, Related Work, and Programming Languages and Tools. During those sections crucial concepts related to this work are described. This includes the most recent research on word representations and network representation learning. The final section contains the reasoning behind the programming language chosen for this work.
3. Data Analysis - contains an exhaustive analysis of the datasets used as input for the set of experiments performed in this work.
4. Design and Implementation - the designed of the performed experiments is defined and justified. Also, the implementation details of every step is described along with the technical description of the problems addressed.
5. Experiment Results - this chapter contains the analysis of the work outputs, when measured against key metrics to evaluate each of the hypotheses defined. The evaluation metrics are also described in this chapter.

6. Conclusions - describes the conclusions collected from the output of this work. In this chapter, its possible to find a description of the achieved objectives along with the challenges overcame during the project. The contributions of this work are also described, along with the future work that can be continued in this topic.

Chapter 2

State of the Art

With the intent of defining the best solution for the problem described in the previous section, it is essential to understand the previous works, the essential concepts of those implementations, and what has not been addressed yet. This chapter contains this type of information and is divided into two major sections: background knowledge and related work.

2.1 Background Knowledge

This first section, includes all the information related with background concepts which are important for the reader to understand.

2.1.1 Knowledge Representation

Y. Zhang et al. (2019) states that it is almost impossible for consumers to find the most relevant items for them without the usage of capable search engines. Semantic networks (Pirnay-Dummer, Ifenthaler, & Seel, 2012) were thought to start coping with the need for knowledge representation frameworks that should be able to capture a wide range of entities, from real-world objects to events or needs. Semantic networks are built from two sets of different kinds of objects: vertices (or nodes) that represent concepts; and edges (directed and labelled) that represent the semantic relations between the concepts. Because of these connections, this type of network is often associated with the term "associative networks". Figure 2.1 presents a simple example of a semantic network regarding clothing.

However, with the growth of search engines, the semantic component of the queries is not the only concern. There is the need to map those queries to the answers to be shown to the end-user. Fortunately, graphs provide a universal way to represent relationships between the same and different types of entities (X. He, Gao, Kan, & Wang, 2017), those that represent two types of entities are called bipartite graphs.

In the case of relevance ranking, given a query input, a model should retrieve the documents that are the most relevant and rank them according to the degree of relevance. As stated by Wu, Li, and Xu (2013), this relevance should be measured as the similarity between the query and available documents.

It is possible to obtain the relevance between sets of queries and documents by using click-through data, which consists of associations of queries and documents, based on user past clicks in a document for a given query. In the area of information retrieval, features

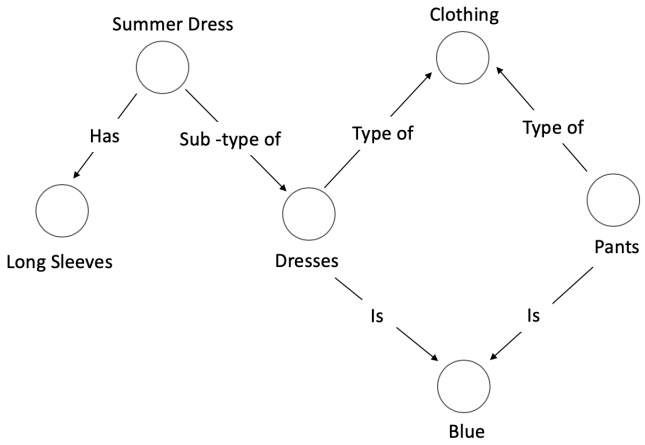

Figure 2.1: An example of a semantic network.

that are based in click information are one of the most important and useful in tasks such as ranking (Jiang et al., 2016).

Nevertheless, features that rely on user behaviour are usually sparse and noisy, especially when considering not popular items, that is why it is essential to also include features capable of describing the items, and that is independent of other resources.

Word vector representations are vectors containing numbers that are capable of representing the meaning of a word and can be of two types: sparse and dense (Mitkov, 2003). Figure 2.2 shows the representation of a single word in a vocabulary of five words.

0	1	0	0	0

-0.084	0.022	-0.119

Figure 2.2: Example of a sparse (left) and dense (right) word vector representation in a vocabulary of five words.

In the sparse representation, it is only possible to retrieve the information that a specific word is present or absent from the observation. In the dense vector representation, each entry of the vector represents a hidden feature inside of the word meaning, which can lead to the identification of semantic or syntactic dependencies.

One way of solving word representation problems is by exploring embedding techniques. Representation learning studies a set of techniques in which individual words are represented as read-values vectors in a pre-defined vector space, leading to the distributed representations being based on the usage of the words. They allow mapping words in a vector space while being able to capture the semantic similarities between them, in order to be able to represent the query-product graph in the same vector space. The significant purpose of this type of frameworks is the capability of grouping vectors of similar words together by detecting mathematical similarities.

In this line of thought, Jiang et al. (2016) presented click graph, a vector propagation algorithm that is capable of learning vector representations for both queries and documents

in the same semantic space. This algorithm is an essential key of this work. The reader will be provided with more information regarding this and other methods of network representation learning on section 2.2.2.

2.1.2 eCommerce Search Engines

A user that relies on eCommerce platforms always wants to find what they are looking for and with the best quality guaranteed. To measure if a result showed to the consumer was the most adequate for their needs, it is necessary to rely on the history of interactions that they performed, such as searches, clicks, transactions, add to bag, as many others (R. Li et al., 2019). According to Pang et al. (2017), even though humans have a different sense of relevance, the process used to find relevant items is similar and have three steps:

1. Detection of relevant locations.
2. Determination of the local relevance.
3. Aggregation of local relevance to output a relevance label.

As previously stated, eCommerce search engines have different objectives, but they are becoming even more popular in everyday life. Those platforms are dynamically increasing and contributing to the most significant percentage of transactions being performed in the world (S. Liu, Xiao, Ou, & Si, 2017; Sondhi, Sharma, Kolari, & Zhai, 2018; Sorokina & Cantú-Paz, 2016). As H. Zhang et al. (2020) stated, three main components guarantee the good behaviour of an eCommerce search engine: query processing, candidate retrieval and ranking. On Figure 2.3 it is possible to observe the macroblocks present on a eCommerce search engine.

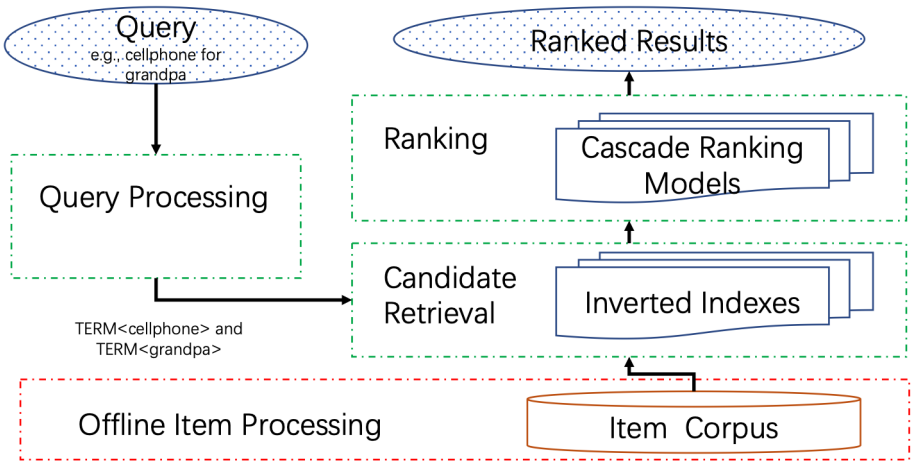

Figure 2.3: Representation of the macroblocks of a eCommerce search engine (H. Zhang et al., 2020).

The main goal of the following subsections is to explain each macroblock, the possible steps and concerns that they address, and how they can be accomplished.

Query Processing

The main goal of query processing is the capability of rewriting the query in terms that can be processed by the downstream tasks. There are some typical components to this step: tokenization, spelling correction, query expansion and rewriting (H. Zhang et al., 2020).

As initially stated by Webster and Kit (1992), the tokenization is the initial phase in natural language processing. This process allows the separation of a piece of text into small segments, called tokens. There are three main types of tokens: words, characters and sub-words. The most common ways to form tokens are based on the space, and the sentences are split by the white spaces that appear in it. Most of the processing being done on natural language processing (NLP) tasks happen at the token level. As already stated, tokenization is the principal task being performed when modelling text data, since the vocabulary can be prepared by considering each unique token present on the *corpus*.

Spelling correction can be one of the essential tasks for a modern search engine but is already being studied for a long time (Damerau, 1964; Levenshtein, 1966; Riseman & Hanson, 1974). As stated by Martins and Silva (2004), misspelt queries can lead to incorrect or inconclusive results. There are several reasons why a misspelling might occur, Duan and Hsu (2011) stated five possible reasons: typing quickly, keyboard adjacency, inconsistent rules, ambiguous word breaking and new words. According to the authors, there are two common mechanisms to solve this problem. The first method performs the query corrections after it is sent to the search engine, being this an offline spelling correction. A second approach is an online approach, in which the corrections appear at the same time as the query is being written. When comparing the two approaches, it is possible to state that the offline spelling correction only needs to evaluate what the user already entered and can present some flaws. On the other hand, the online approach needs to address incomplete queries, but also requires lower latency to be effective (Duan & Hsu, 2011).

Query expansion is a process that consists of adding and selecting terms to the user search term with the intent of minimizing the possible mismatches between the query terms and the products. With this, there is a possible improvement in the retrieval performance (Vechtomova, 2009). The terms used for query expansion are typically abbreviations or synonyms. The abbreviations method consists of recognizing those words in the queries and documents, and then substitute them with the same meaning abbreviated term. Similarly to the abbreviation method, when resort to the use of the synonym method, it is also necessary to take into consideration the inherent ambiguity of language. For instance, if there is a semantic relationship between the synonym and the original term, it is possible to take this information into account and use an analogue word that is more specific than the original term (Azad & Deepak, 2019).

As stated by Mandal, Khan, and Kumar (2019), query rewriting is also considered to be a critical component in modern search engines, like the methods previously explained. Tasks like this are usually used to serve two purposes: increase recall and precision. Query rewriting can sometimes be seen as part of the query expansion or correction task since its primary goal is also to remove the gap between queries and documents. On the context of eCommerce search engines, user queries tend to be short and full of colloquial terms when comparing those with the listed products. Several authors, such as Cui, Wen, Nie, and Ma (2002), R. Baeza-Yates and Tiberi (2007) and Y. He et al. (2016), had already studied and proposed automatic methods that allow identifying how a query can be rephrased in a way that will provide better and more accurate results for the user.

All of the methods stated above have one ultimate final goal that is improve the results shown to the user. Only with these, the search engines can perform well and get a clear view of what is the user intention when he was typing the query.

Candidate Retrieval

The candidate retrieval is responsible for retrieving candidate items based on term matching efficiently. This step is built with the intent of reducing the number of items being retrieved, to make the fine ranking feasible (H. Zhang et al., 2020).

Previously it was common to represent the items with a shared group of features, with keywords or certain tags. With this information, the relevant candidates can be retrieved based on features similarity (J. Wang & Li, 2012). Conventional methods that are based on feature similarity are excellent performers when considering real-time retrieval. However, their quality will be limited if there is the desire to consider the user information when retrieving the products (Z. Liu et al., 2019).

According to H. Zhang et al. (2020), this step is considered to be of greater importance, since by reducing the number of possible items from billions to thousands, it will be possible to retrieve feasible rankings, that are considered to be more elegant and accurate.

Ranking

On this step, the main challenge is to order the already chosen candidates according to some crucial factors, such as relevance or conversion rate. On Figure 2.4 it is possible to see a typical Learning to Rank (LTR) flow. From the figure, it is possible to state that this is a supervised task since a training set is needed. This training set is composed of a set of queries containing the documents associated with them and the corresponding relevance judgments (the ground truth). With this data, it is possible to build a specific learning algorithm that will be responsible for learning the ranking model. In this step, the model being built must have the capacity to understand how to combine the different features that allow it to retrieve the best results. These combinations must be performed since the output will be measured by how accurate it was compared with the truth labels. In the test set, a new query is introduced to the system. The model learned during the train phased is used to sort the documents associated with the query (that were assigned to it during the candidate retrieval phase) and output the corresponding ranked list.

As stated by H. Li and Xu (2014), there are different approaches capable of modelling the learning process differently from each other. All of them have different inputs and output spaces, and even their hypothesis differ since they are built on top of different loss functions. The approaches used to perform the task of learning to rank are:

- Pointwise
- Pairwise
- Listwise

The pointwise approach takes a single document and trains a regressor or classifier with the intent to predict how relevant that product is to the current query. On this approach, the score of each product is independent of the other products present in the final result list. The final ranked list is obtained by sorting the result list by the product scores. It is necessary

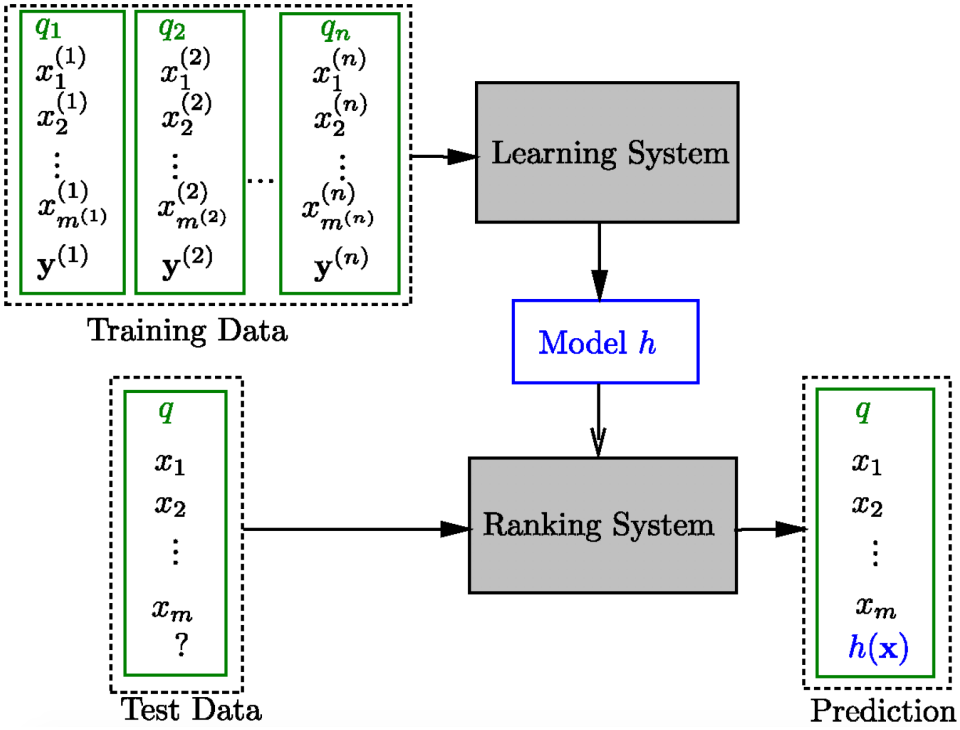

Figure 2.4: Scheme of a general learning to rank framework (T.-Y. Liu, 2009).

to state that this approach has some limitations. For instance, it does not consider possible interdependency among documents, and it is entirely irrelevant to it if the products are associated with the same query (H. Li & Xu, 2014). All standard regression and classification algorithms can be used to solve this problem. Some examples can be found on Crammer and Singer (2002); P. Li, Burges, and Wu (2007) and Chen and Lan (2009).

During a pairwise approach, the documents are combined in pairs, and the trained model has the intent of coming up with the optimal order of the given pair. The main goal of this approach is to minimize the number of inversions in the ranking, minimizing the cases in which the pairs of results are in the wrong order when comparing with the ground truth. This approach is better than the last one because predicting the relative order is closer to the true nature of ranking, instead of predicting a relevance score. Some of the most popular LTR algorithms are pairwise approaches, such as the ones presented by C. Burges et al. (2005); Y. Cao et al. (2006); Freund, Iyer, Schapire, and Singer (2003); Tsai, Liu, Qin, Chen, and Ma (2007) and C. J. C. Burges (2010).

The listwise approach is the most robust and sophisticated approach, in which it takes the entire list of documents, and the model is responsible for learning the optimal order. There are two main sub-techniques for this approach, meaning that there are two ways of calculating the loss function. One of the methods, presented by Taylor, Guiver, Robertson, and Minka (2008); Xu and Li (2007), is accomplished by directly optimizing information retrieval metrics, such as normalized discounted cumulative gain (NDCG). The main goal of the other method is to minimize the loss function that is defined based on the understanding of the properties of the optimal rank that the user is trying to achieve. Some of the popular algorithms that use this approach were presented by Z. Cao, Qin, Liu, Tsai, and Li (2007); Lan, Zhu, Guo, Niu, and Cheng (2014).

Challenges of eCommerce search engines

One of the main challenges of this type of search engines is the idea of retrieving more personalized and semantically relevant items, bringing two new concepts: *Personalized Retrieval Problem* and *Semantic Retrieval Problem* (H. Zhang et al., 2020).

The *Personalized Retrieval Problem* is related to the fact that the traditional methods are not able to retrieve different items according to user preferences or characteristics. Information that can be powerful and indicative of the users' preferences (such as gender, age or purchase power) are not being used to perform better and richer listing pages for the users.

The *Semantic Retrieval Problem* will be studied during the length of this document. This problem exists because the items can be semantically relevant, but since they do not contain the exact terms of the query, they will not be retrieved by the traditional methods. As stated in H. Li and Xu (2014), the most common and critical challenge of search engines is the mismatch between the terms used on the queries and items. When considered eCommerce search engines, those problems are exacerbated since item titles can be relatively short and the search terms can contain the intent of the user (e.g. "I want to buy red shoes for tonight's party"). Strategies to mitigate this issue can be taken into place on two of the macro-blocks introduced. For instance, on the query processing, a possibility is to use methods that allow a query rewriting, in its essence the original query is being transformed into a better representative of the search needs (H. Zhang et al., 2020). As already stated during this document, one of the main goals of this work is to study how the vector propagation methods behave as a candidate retrieval, and how does it perform with different word representations.

2.2 Related Work

An eCommerce search engine drives to present relevant results by taking the needs and expectation of the user into account. Early work from Manning, Raghavan, and Schütze (2008), and Robertson and Zaragoza (2009), looked into query-document matching algorithm solutions that presented to be useful but do not take the context into account. Recent advances in the literature are leading the way to the use of deep learning networks, as the ones presented by Brenner et al. (2018) and Y. Zhang et al. (2019). However, this type of research is still in their early stages, and there is a lack of quality when considering the results obtained. Therefore, some researchers are still focusing on more straightforward approaches, such as the one presented in Jiang et al. (2016).

The next sections will show in detail some of the existent studies conducted regarding word and network representation learning which will be the starting point of the improvements proposed by this work.

2.2.1 Word Representations

In 1988, Rumelhart, Hinton, and Williams (1988) showed that a distributed representation of words in a vector space improved the algorithms accuracy in natural language tasks since they were able to group similar words. The usage of dense and low-dimensional vector have a computational benefit since the predominance of neural network toolkits are not able to cope with very high-dimensional and sparse vectors, as stated by Goldberg (2018).

Term Frequency-Inverse Document Frequency, TF-IDF

One of the most used techniques is the Term Frequency-Inverse Document Frequency (TF-IDF) is used in the context of document search and information retrieval (R. A. Baeza-Yates & Ribeiro-Neto, 1999; Jones, 1972). In these methods, a weight is calculated to measure how important is a word in the given document taking into consideration the corpus. Word importance proportionally increases with the number of times that words appear in the document but is offset by the number of documents that contain the word. Meaning that common words, such as *what*, *if*, *that*, may appear many times in a document but are ranked lower since their representation for the document is also low. The calculation of this weight can be represented by equation 2.1.

$$w_{i,j} = tf_{i,j} \times \log(\frac{N}{df_i}) \tag{2.1}$$

In which, $tf_{i,j}$ represent the number of occurrences of i (a word) in j (a collection or corpus), $df_{i,j}$ is the number of documents that contains i, and N is the total number of documents.

Nonetheless, this approach is not able to capture the semantic representation and similarities of the words, but rather the frequency that of the exact match of a word appears in a set of documents, which lead to the appearance of dense word representations.

Word2Vec

Later, Bengio, Ducharme, Vincent, and Janvin (2003) presented techniques capable of reducing the dimensionality of word representations while, at the same time, preserving the context. This type of representation has an improvement when compared with the TF-IDF approach. When words are used in similar ways, they end up having an equal representation, which means that their meaning was naturally captured.

In 2013, Mikolov, Yih, and Zweig (2013) presented a framework that contained both of these methods, the Word2Vec. This technique is capable of creating a vector representative of distributed numerical representations of word features. With enough data, usage, and contexts, it is able to predict the semantic of a word based on its past appearances. These predictions are representations of the linear relations between word pairs. For instance, considering the word vector "boy" and calculate its distance to another word such as "girl". If a different vector is applied in other words, such as "man", the output will have the representation of the word "woman". Figure 2.5 shows visually how this word analogy can be represented.

This is possible since this framework is based on a feed-forward neural network. By using non-supervised tools, it is possible to do two types of predictions:

1. Predict the target word given the context by using a Continuous Bag of Words (CBOW) approach.

2. Predict the context given the target word with the help of a Skip-gram approach.

Either the input or the output of this method is one-hot encoded vectors with the length of the vocabulary size.

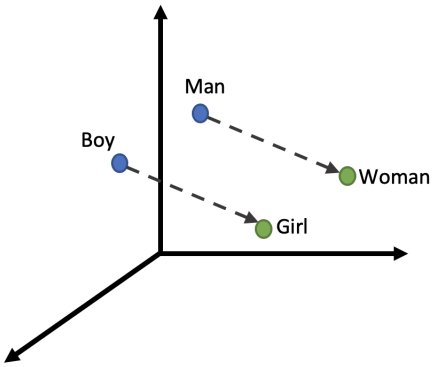

Figure 2.5: Word analogy visually represented.

As already stated, the CBOW models tend to predict the probability of a word given a context, that can be a single word or a group of words, in Figure 2.6 it is possible to see a high-level architecture for this model. A shallow neural network composes this framework with three layers: an input layer, a hidden layer and an output layer. The output layer is composed by a softmax layer responsible for the sum of the probabilities obtained in the output layer.

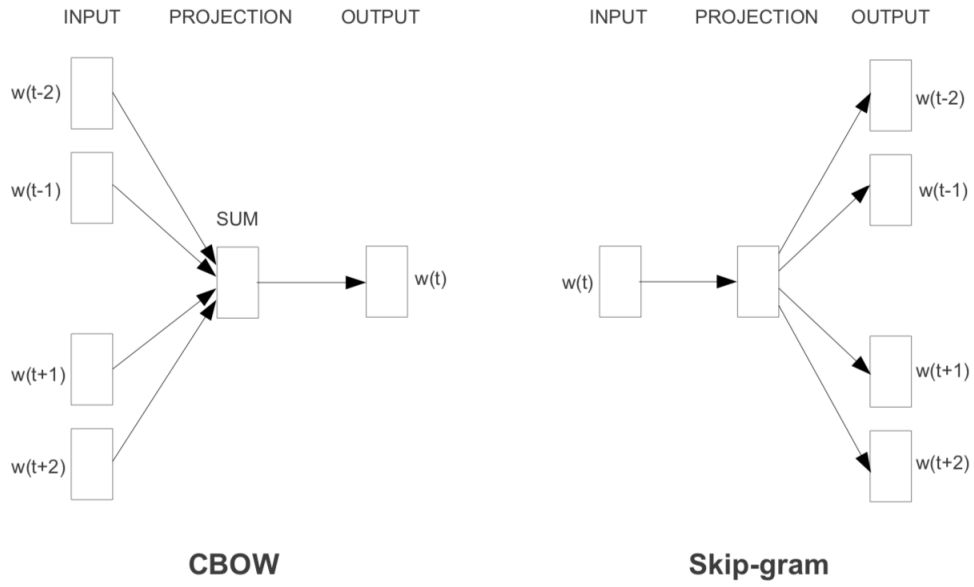

Figure 2.6: CBOW and Skip-gram model architectures presented by Mikolov, Chen, et al. (2013).

The Skip-gram approach, presented by Mikolov, Sutskever, Chen, Corrado, and Dean (2013a), is a model capable of learning vector representation from significant amounts of unstructured data. In this model (high-level architecture diagram in Figure 2.6), the train has the objective of learning word vector representations that can predict the nearby words in a sentence or a document.

In summary, both models have a focus on learning words taking into consideration their

context, defined by a model configured window of neighbouring words. Mikolov, Chen, et al. (2013) stated that the Skip-gram model has a better performance with smaller amounts of data and is capable of correctly represent rare words. However, the CBOW is faster and is best suitable for more frequent words.

GloVe

Word2Vec is not the only framework that has good results when it comes to word embeddings. In 2014, Pennington, Socher, and Manning (2014) presented another word embedding solution named GloVe, a method able to collect the word co-occurrence statistics in the form of word co-occurrence matrix. The main objective of this framework is to learn word representations that their dot product equals the logarithm of the words' probability of co-occurrence, meaning that during the learning component it takes the global information into account.

GloVe is a method capable of creating a global co-occurrence matrix with the intent of estimating the probability of a word co-occurring with other words. As stated by the authors (Pennington et al., 2014), this method is an unsupervised learning method for word representations, a log-bilinear regression model that can be used for tasks such as word similarity, and word analogy.

Embeddings from Language Model, ELMo

More recently, Peters et al. (2018) presented a new model called Embeddings from Language Models (ELMo) that can capture aspects that are dependent on the word meaning context. Three layers compose ELMo:

1. Input layer: a convolutional neural network (CNN) that is feed with raw characters and produces compact embeddings.

2. Hidden layer: composed by a multi-layer bi-directional long short-term memory (LSTM). This layer that has as input the output sequence from the previous ones except for the first LSTM that receives the character embedding from the input layer.

3. Output layer: an embedding layer that concatenates the hidden states of the LSTMs and produces context-dependent embeddings.

ELMo can build word embedding dynamically, taking into considerations a sequence of characters. Besides that, it relies on the current surrounding words in the sentence, which means that the same word has different embeddings when is used in distinct contexts.

Bidirectional Encoder Representations from Transformers, BERT

Devlin, Chang, Lee, and Toutanova (2018) introduced language representation model, the Bidirectional Encoder Representations from Transformers (BERT). Previous works have a deep focus on training models in an ordered sequence of words (left-to-right or by combining left-to-right with right-to-left), but BERT brought the ability of training language models with all words in a sentence or query (named bidirectional training). With this new way of learning, the authors presented a model capable of learning word context based on all surrounding words.

The framework contains two steps: pre-training and fine-tuning. Two unsupervised tasks accomplish the pre-training:

1. Masked Language Model (MLM): input tokens are masked at random to predict the original vocabulary of the masked word taking into consideration its context.

2. Next Sentence Prediction (NSP): with the intent of understanding the relationship between sentences, the authors trained a binarized next sentence prediction task that is possible to be generated from a monolingual corpus.

During the fine-tuning task, the self-attention mechanism present in the transformer enables the model presented by Devlin et al. (2018) to prototype the downstream tasks and allow the swap between the right input and output.

BERT boosted a new era in the field of NLP by applying the transformer in these types of tasks, which lead to several types of research considering different aspers of pre-training, transformers and fine-tuning. From that, several variations of the models proposed by Devlin et al. (2018) came along, such as RoBERTa presented by Facebook AI researchers (Y. Liu et al., 2019), and DistilBERT (Sanh, Debut, Chaumond, & Wolf, 2019) a compact and faster version of BERT.

XLNet

A recent breakthrough in the language modelling tasks was presented by Yang et al. (2019), the XLNet. Being a generalized autoregressive method, XLNet introduces a new concept named permutations of the factorization order. With the usage of permutation, the retrieved context is a combination of tokens from both left and right positions which allows not to use masked token (used by BERT (Devlin et al., 2018)).

XLNet does not alter the text sequence but relies on the position encoding and the attention mask to achieve the permutation. The first guarantees the position information of the text and the second makes sure that is given different attention to text (different weights according to its relevance) when creating the input embedding.

When comparing XLNet with BERT, it is possible to state that the first does not make independence assumptions since it does not mask the tokens, meaning that there is no disparity between the pre-training and fine-tuning. Besides that, XLNet can capture dependencies in longer sequences by propagating states across segments.

Table 2.1 compiles the frameworks presented in this section followed by their advantages and disadvantages.

Table 2.1: Comparison between the different types of word representations frameworks.

Name	Advantages	Disadvantages
TF-IDF	• Easier to comprehend. • Each term is considered as a dimension of the vector. • Good for exact word matching tasks, rather than phrases.	• Needs a big amount of data in order to have good results. • Does not capture position in text, semantics and co-occurrences in different documents.
Word2Vec	• Able to transform the unlabeled raw corpus into labeled data. • Capable of doing online preprocessing with the need of a big amount of resources. • Relationships between words can be retrieved from the embeddings.	• Sub-linear relationships are not explicitly defined.
GloVe	• Good performance in word analogy tasks. • Considers the relationships between word pairs. • Gives lower weight for highly frequent word pairs, such as 'like' and 'the'.	• The training set takes a big amount of storage memory.
ELMo	• Take into account word order. • Generate embeddings that are context sensitive. • Feature-based approach.	• Inference cost due to run-time predictions.
BERT	• Pre-trained language model. • Capable of learn the representations of out-of-vocabulary words. • Generate embeddings that are context sensitive.	• Hyper sensitive to hyperparameters. • Treats masked tokens independently which can lead to some limitations if they are dependent.

Name	Advantages	Disadvantages
XLNet	• Capable of learn the representations of out-of-vocabulary words. • Generate embeddings that are context sensitive.	• Requires high compute power and memory.

2.2.2 Network Representation Learning

As seen in Section 2.1.1, knowledge graphs have been used in different real-world applications, and the interest in applying them to some of the known machine learning (ML) techniques is growing. However, performing mathematical or statistical operations on graphs is limited, and algorithms only perform well until a certain level. Those types of challenges led to the growing interest in using embeddings, which allows transforming some properties of the graphs into a vector or a set of them. Those embeddings can capture graph topology, vertex-to-vertex relationship, and other relevant information about graphs, subgraphs, and vertices. Being able to learn a representation from these highly structured objects can be helpful in several machine learning tasks.

DeepWalk

DeepWalk, presented by Perozzi, Al-Rfou, and Skiena (2014), is one of the first approaches used as a benchmark to compare with other graph learning approaches. The framework uses walks, a concept of graph theory that allows travelling through a graph by moving from one node to another if connected by a common edge. To make predictions, DeepWalk uses a three-step method:

1. *Sampling*: the model takes a target node, picked randomly and "walks" to a neighbour node randomly until it reaches the maximum walk length (Figure 2.7).

2. *Training skip-gram*: the network takes as input a one-hot encoded node retrieved from the random walk, and tries to predict the probability of the neighbour nodes.

3. *Computing embeddings*: the node embeddings are retrieved from the output of a hidden layer of the network.

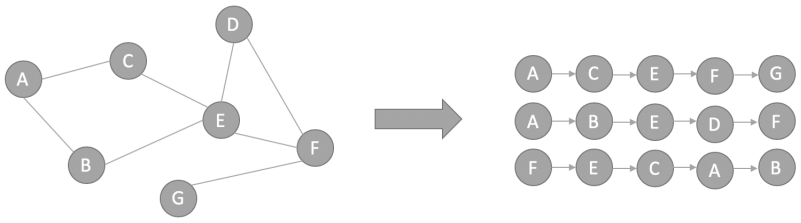

Figure 2.7: Example of random walk sequences (right) generated from the knowledge graph (left).

Since DeepWalk performs walks randomly along with all the graph structure, this means that the embeddings are not capable of preserving some local neighbourhoods correctly, as stated by D. Wang, Cui, and Zhu (2016).

Large-scale Information Network Embedding, LINE

Later, a Large-scale Information Network Embedding (LINE) was present by Tang et al. (2015). The authors proposed a scalable solution capable of dealing with arbitrary types of information when using real-world data, that usually contains millions of nodes and billions of edges.

LINE, as stated by Tang et al. (2015), was developed with the intent to preserve the first-order proximity and second-order proximity separately:

1. First-order proximity: two nodes are similar if they share an edge (also known as pair-wise similarity).

2. Second-order proximity: two nodes are similar if they share neighbours/adjacent nodes.

After that calculation, the authors Tang et al. (2015) presented a simple way of combining the two proximities by concatenating both trained embeddings for each vertex. The final goal is to minimize the Kullback-Leibler divergence (also called relative entropy, it is a measure of the difference between two probability distributions) of these two distributions. Besides that, only nodes that are less than two steps away from a given node are considered as its neighbours.

Node2vec

In 2016, Grover and Leskovec (2016) presented Node2vec, a framework capable of learning continuous feature representations for graph nodes in a network, which allows neighbourhood graph optimization. The authors presented a flexible notion of the network neighbourhood and designed a biased random walk procedure that is capable of exploring the diverse neighbourhoods. Node2vec was an improvement done on top of DeepWalk (Perozzi et al., 2014), an algorithm in which random walks are randomly performed, leading to embeddings that do not preserve local neighbourhoods.

When compared with the DeepWalk method, Node2vec present a flexible approach when it comes to exploring the neighbourhoods present in the graph. The new method is capable of learning the embedded representations of the nodes from the same network community and is capable of learning that nodes with similar roles have similar embeddings. This principle leads to better results on multi-label classification and link prediction. The crucial contribution is in defining a flexible notion of a node's network neighbourhood by developing a family of biased random walks, which efficiently explore diverse neighbourhoods of a given node.

The flexibility of this implementation allows the algorithm to perform a random walk (during the sampling step) that can interpolate between breadth-first search (BFS) and depth-first search (DFS) (Kozen, 1992). In BFS the neighbourhood is restricted to nodes which are immediate neighbours of the main one. Furthermore, in DFS, the neighbourhood is formed by nodes that are sampled sequentially by increasing the distance from them to the primary node. The sampling strategy allows Node2vec to generate "sentences", considered

to be directed subgraphs, that will be used for the embedding step, similar to the technique explained in subsection 2.2.1.

Despite Node2vec presenting good results (Grover & Leskovec, 2016), the method is not prepared to be used in heterogeneous information networks nor in networks that contain explicit domain features for its nodes and edges. The lack of those characteristics is not a problem when taking into consideration a homogeneous network but are crucial for bipartite networks in which there is a need to combine two different types of entities that can have implicit and explicit features.

Click Graph

The click graph approach, presented by Jiang et al. (2016), is capable of joining clicks and content information into one single vector space that can be directly used to improve the ranking of documents and queries already seen. In Figure 2.8 it is possible to see a simple example of a bipartite graph that connects queries with products.

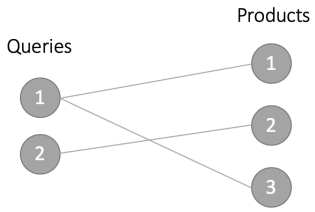

Figure 2.8: Example of a click-through bipartite graph.

The main goal of the click graph vector propagation is to be able to learn, in the same semantic space, the representation of queries and documents. In Figure 2.9 it is possible to see a simple execution of this method in one query of the graph presented in Figure 2.8. At the beginning of the process, a side of the click graph is chosen. The vectors initialization is based on the existent content information on that side (upper scheme of the Figure 2.9). During initialization, the word-level information is kept, each query is represented by the weights (proportional to their frequency and normalized by the distance of the vector coordinate from the vector space origin - also known as L2 norm) of its own words (Jiang et al., 2016). Then, the information is transferred from one entity to the other, and the weighted representations are updated according to the information retrieved. At last, the information is back-propagated, and the same update is done. This process is repeatedly done until convergence is reached in all of the graph structure.

Even though this method presented a significant improvement when compared with multiple baselines, it presents some flaws when considering an eCommerce search engine (Jiang et al., 2016). First of all, query language is usually different than document language, which leads to a word representation problem. Then, search queries follow a long-tail distribution and the majority of queries have a low frequency or are entirely new to the engine. This usually means that user behavioural information (click-through data) is not available, which makes impossible to apply click modelling to these type of queries.

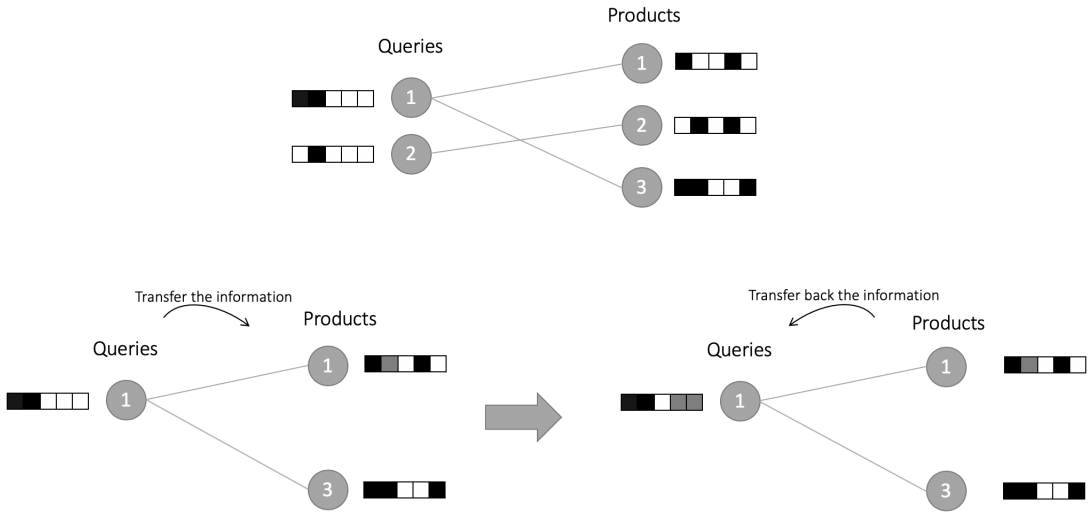

Figure 2.9: Example of vector propagation on a click graph.

Bipartite Network Embeddings, BiNE

More recently, Gao, Chen, He, and Zhou (2018) presented a new framework named Bipartite Network Embeddings (BiNE). BiNE presented a focus on learning embeddings on bipartite graphs (for tasks such as recommendations and search logs), besides the fact that the main goal is to be capable of modelling both explicit (observed links) and implicit (unobserved but transitive links) information. Besides that, the authors stated that both types of relationships typically have different semantics, so BiNE is capable of treating them differently when it performs vertex embeddings learning.

The ability to learn each relation comes from a dedicated objective function, leading to the reinforcement of both types of information that allows better vertex embeddings. When modelling explicit relations, the authors aim to reconstruct the network by focusing on the observed links. Implicit relations aim to find high-order correlation in the bipartite graph. In order to keep the network properties, BiNE was built on top of a biased and self-adaptive random walk generator. This method allowed the authors to preserve properties such as the importance and distribution of vertices.

Nevertheless, BiNE also presented some flaws in its performance, since Gao et al. (2018) only considered the information that can be seen in the observed edges, this can lead to bad representations in vertices that have few or none edges. Taking into consideration that when using real datasets, having missing or noisy data is a common problem, it is possible to see those type of errors happening. However, the issue can be bypassed by using auxiliary information such as textual descriptions (T. M. Le & Lauw, 2014) and other attributes (Liao, He, Zhang, & Chua, 2018).

Table 2.2 compiles the frameworks presented in this section followed by their advantages and disadvantages.

Table 2.2: Comparison between the different types of network representations learning frameworks.

Name	Advantages	Disadvantages
DeepWalk	• Simple and effective on homogeneous networks.	• Needs to be re-trained when a new node is added. • It is not possible to control the path through the graph. • Incapable of preserving the network properties.
LINE	• Capable of learning 1st-order and 2nd-order relations.	• Bad performance when taken into consideration the node community structure.
Node2vec	• Preserves the network properties.	• Searches the graph in a weighted random way.
Click Graph	• Focus on bipartite networks. • Models clicks and content information. • Expands the vector representations with relevant new words and smoothes the terms with the clicks weights.	• Needs data with high connectivity to generate accurate vector representations.
BiNE	• Focus on bipartite networks. • Preserves the network properties. • Model implicit and explicit information.	• Only considers the information in the observed edges.

2.3 Programming Languages and Tools

Python[1] has been growing its popularity, and according to studies performed by Srinath (2017) it was the 5^{th} largest StackOverflow Community and the 4^{th} most used at GitHub. Accordingly to a StackOverflow analysis, Python is the fastest-growing language for data science, machine learning and academic research (Robinson, 2017). This can be further cor-

[1] https://www.python.org

roborated by analysing the results of the StackOverflow 2020 survey[2] in which Python is the most wanted language.

As well as being an easy to use and versatile language, it also contains a vast repository of open-source libraries that can be used for a variety of tasks and problems, alongside with built-in support for scientific computing (Rao, 2018).

The fact that Python is an object-oriented language is a big advantage when looking for stability, modularity and code readability (Srinath, 2017).

Besides all advantages enumerated, all of the methods described in this chapter only have their implementations in Python. For this reason, Python was the language of choice to develop this work.

[2]https://insights.stackoverflow.com/survey/2020#overview

Chapter 3

Datasets

With the evolution of machine learning techniques, the number of public datasets for research has been rising every day. However, specific datasets with click-through data are challenging to find since they end up giving more information regarding the company intellectual properties, such as its popularity and possible profits - that can be inferred by the value of products transactions. Nonetheless, to conduct the experiments and validations for this work, it is possible to transform relevance datasets into click-through logs. There is some limitation to this transformations since it is not possible to simulate human behaviour, but there are some conditionals that can be taken into consideration. Chapter 4 explains this process in detail.

On the following sections, it is possible to find a detailed description of the two datasets that will be used to evaluate the implementations, its features and structure.

3.1 Home Depot

The Home Depot public dataset is shared in a public kaggle competition[1]. It contains several products and real customer search terms from Home Depot's website. All of the items present in this dataset are related to home improvements, gardening and do-it-yourself projects.

The data is divided into four files: *train*, *test*, *attributes* and *product_descriptions*. The main product features present in the entire dataset are the following:

- Search query: the actual search term provided by the user.
- Product title: the title of the product returned for the search query.
- Product descriptions: text description of each product.
- Product attributes: extended information about a subset of the products, typically are technical specifications of the products in questions. Not every product will have attributes.

[1] https://www.kaggle.com/c/home-depot-product-search-relevance/overview

Since the kaggle competition has already ended, the ground truth is present on both *train* and *test*. In these types of datasets, the ground truth is the relevance label. Each query-document pair contains a human-annotated relevance label, which is the average of five ratings assigned by different human annotators, and explains how relevant the product is for the matched query, with three levels (The Home Depot, 2016):

1. Irrelevant:
    - The result to not relate to the intent of the search term;
    - The result has a completely different meaning;
    - The result is an item that can be used with the search term, such as tools and accessories;
    - The result is a collection of items, and the searched term does not appear in the said set.
2. Partially or somewhat relevant:
    - The result matched the query, but there is some ambiguity on the search terms or product details;
    - The result is the actual product but with different characteristics from the search query (such as the brand, dimensions and colour);
    - The product specifications are unclear, or the product is an ambiguous synonym of the search term;
    - The search result only presents one type of searched products (example: when searching for "paint & brushes" only paint is being shown).
3. Perfect match:
    - The result is a perfect match of the query intent;
    - The product being searched is part of a kit or an item of the product shown;
    - The product is what the customer was most likely trying to find;
    - The search query was a brand, and the result is products from that brand.

In Figure 3.1 is possible to find a representation of the relevance label distribution of the dataset.

As it is possible to observe on Figure 3.1, there are more relevant products on the query-product pairs than products that do not correspond to the user search intent, which means that, on this datase, there is a higher probability of retrieving products that do belong to the search query than to retrieve irrelevant products. Besides that, there is an average of 7.68 products per query, which implies that the dataset is not rich in terms of search results, but it compensates on the number of product-query pairs since there is a total of 186131 unique product-query pairs. However, each product is, on average, connected with 1.74 queries which means that the majority of the products do not belong to more than one search query. On average, each search query presents an average of 3.15 words with a standard deviation of 1 word. From the analysis, it is possible to state that 66% of the search terms are composed by 2 to 4 words. On Figure 3.2 it is possible to see the distribution of word count by query.

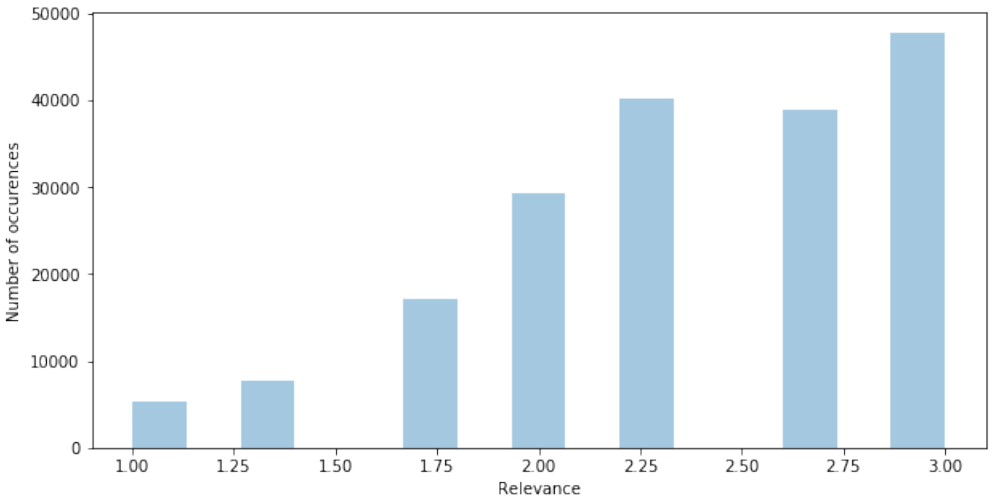

Figure 3.1: Distribution of the relevance labels on the Home Depot dataset.

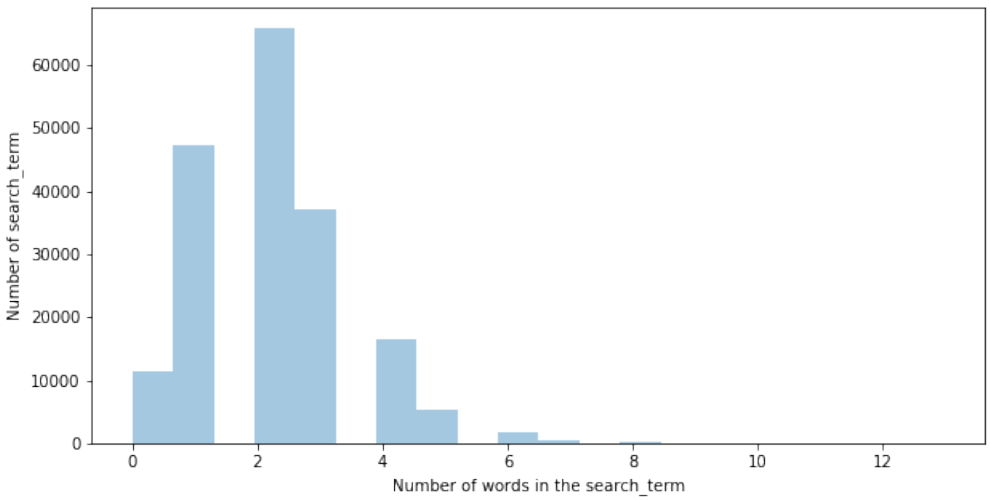

Figure 3.2: Distribution of the word counts on the search terms on the Home Depot dataset.

From the chart present in Figure 3.2, it is possible to state that the search queries are probably more focused on the products, rather the intention of the search. It is possible to conclude that the search terms are objective and with a low number of terms, implying that the users are not expressing their intent (such as 'I want to'), but are focusing on the products. The low word count per query can be seen as a positive point when cleaning the textual data, since it is not necessary to remove the intent of the user and it is possible to focus on other aspects, such as understand how the numeric values are affecting the results.

After this analysis, it was performed an in-depth examination of the connections between the search terms and the products. For that, the graph was constructed, and its properties were analysed. On the following plot (Figure 3.3), it is possible to see a subset of the graph only considering five queries.

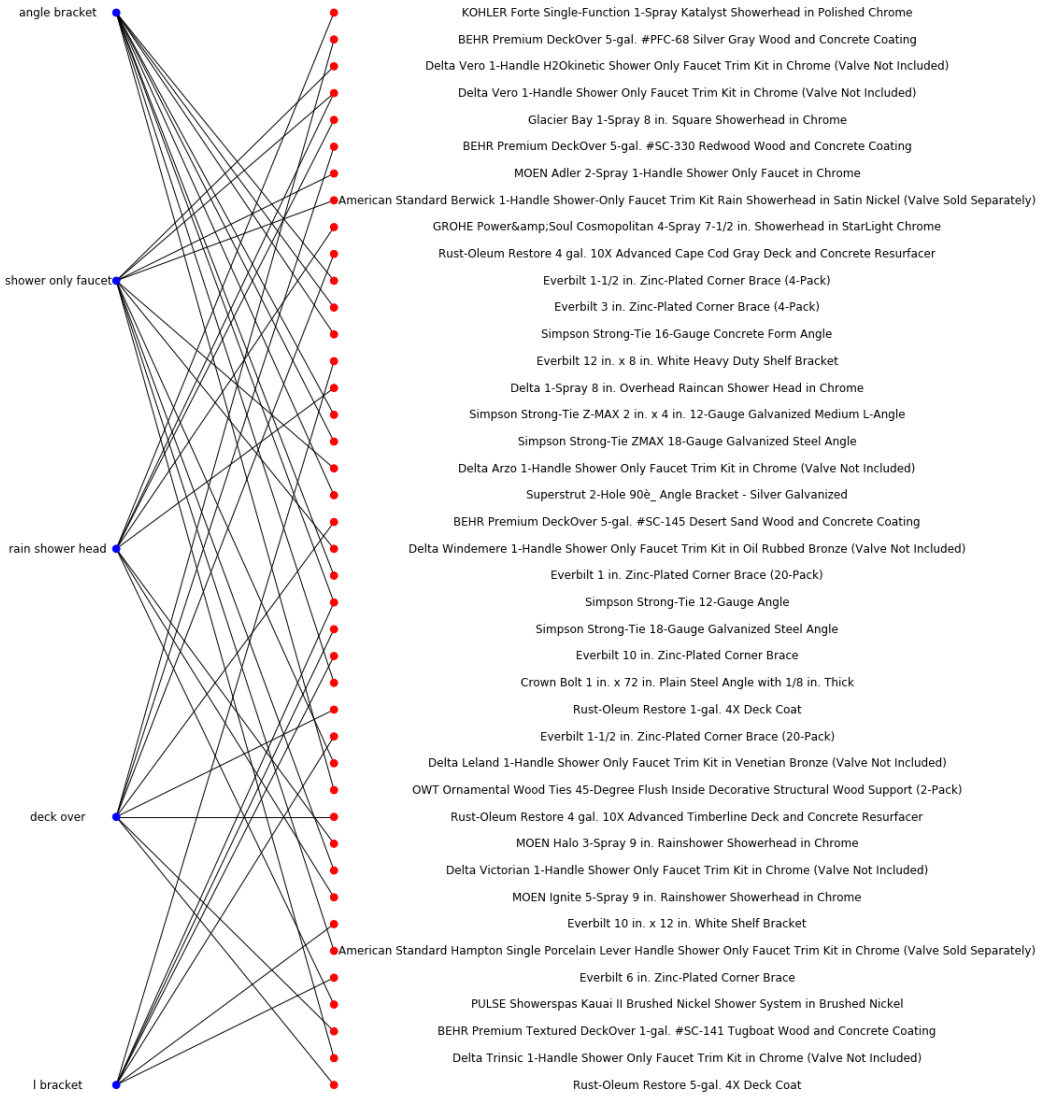

Figure 3.3: Example of the graph structure of the Home Depot dataset.

In conclusion, the Home Depot dataset is a robust dataset that can be used to analyse eCommerce relevance products and to understand how product-queries pairs behave with different methods. Due to these characteristics, it is possible to use this data to the present study.

3.2 CrowdFlower

The CrowdFlower's dataset is also a public dataset that can be found on Figure Eight platform[2] and also on kaggle[3]. This dataset is somewhat similar to the HomeDepot's, with the exception that this one was created using query-result pairing enriched on the CrowdFlower platform. With the intent to evaluate search relevancy, the company with the aid of human curators evaluated the results of 261 search terms from a set of eCommerce websites. The dataset presents the following features:

- Query: the actual search term provided by the user.
- Product title: the title of the product returned for the search query.
- Product descriptions: text description of each product.
- Source: the eCommerce website in which the search was performed.
- Product Image: Uniform Resource Locator (URL) of the product image.
- Rank: Product position returned on the search.

Besides the features stated above, the dataset also contains a relevance value associated to each query-product pair. This feature is the mean of relevance score attributed by three human raters. With the intent of explaining how relevant is the product for the matched query, it presents 4 levels, and each level has a different set of rules (Figure Eight, 2015):

1. Off-topic:
    - The intent of the query was not matched;
    - The shown results are irrelevant to the search query.
2. Acceptable:
    - The intent of the query is poorly matched;
    - The result is somewhat related with the query, but it is not a good match.
3. Good:
    - It matches most of the query intent, or at least the most important part of the query;
    - The most part of the intent is satisfied but the results do not provide a full, clear and complete answer to the search.

[2]https://www.figure-eight.com/data-for-everyone/

[3]https://www.kaggle.com/c/crowdflower-search-relevance/rules

4. Excellent:

    - The query intent is clearly satisfied;
    - The result is high quality;
    - The specifics of the query appear in the result.

After the analysis of the relevance labels, it is important to verify the distribution of the labels on the dataset. This is present on Figure 3.4.

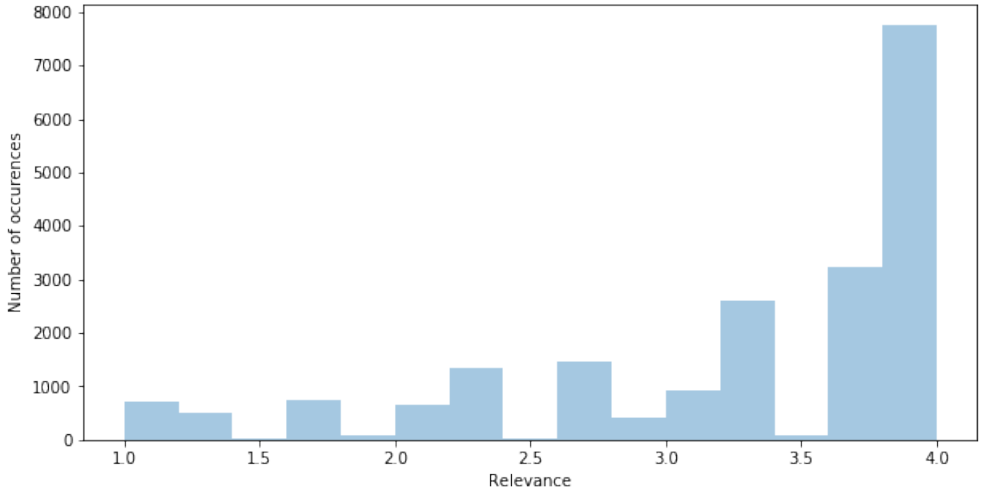

Figure 3.4: Distribution of the relevance labels on the CrowdFlower dataset.

As it is possible to conclude by observing Figure 3.4, there is a higher set of products that match the label *Excellent* in terms of relevance, and the average value for this feature is equal to 3.25. One attractive characteristic of this dataset is the presence of the standard deviation of the relevance value since it allows an analysis of how the human curators were on concordance with each other. The main difference between this dataset and the previous one is the number of products per query. CrowdFlower dataset, on average, presents 123.75 products per search term, which means that the result lists are abundant with products. Nonetheless, the number of queries per product is, on average, 1.10 queries, implying that most of the products are only connected to one search term.

Afterwards, it was necessary to understand the queries formulation, their word count and if the intent is expressed. On average, each search query has 2.40 words with a standard deviation of 0.85 words. From the analysis, it is possible to state that 81% of the search terms have between 2 and 3 words. On Figure 3.5 is possible to visualize the word count distribution.

Similarly to Home Depot's dataset, the search intent is also not present on the search terms, and they were formulated with a focus on the products. Therefore, the textual cleaning of the dataset will be simple and straightforward without the need to remove common words that are usually used to express the user intent.

Lastly, a thorough analysis of the graph was conducted, and a subset of the graph was plotted (present in Figure 3.6). Due to the high cardinality of the dataset, 60 products were chosen to be shown on the example; otherwise, it would not be possible to see an example of the connections.

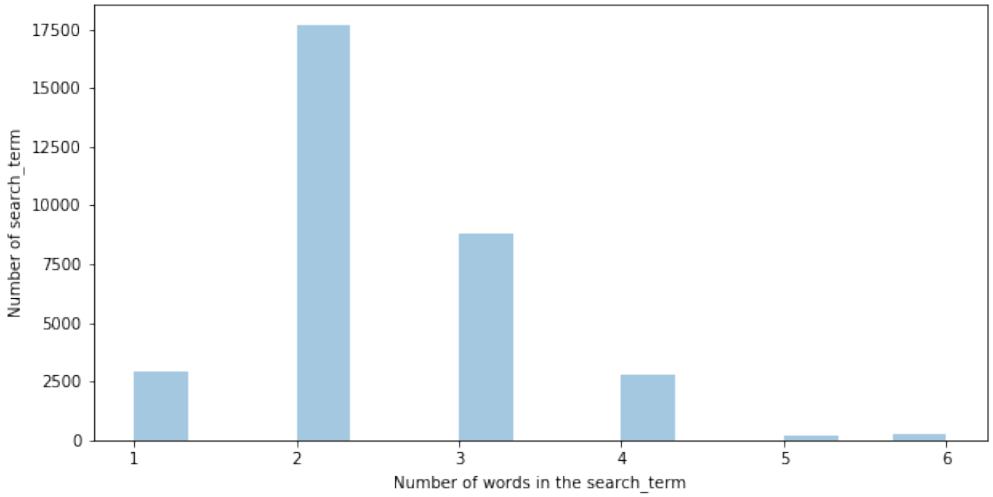

Figure 3.5: Distribution of the word counts on the search terms on the CrowdFlower dataset.

As previously mentioned, most of the products are only connected to one search term, and that statistic is visible on this subset. Only the product with the name *"Jeunesse Instantly Ageless – New Box of 50 Sachets – Eye - Face Wrinkle Cream"* is connected to two search terms *"face cream"* and *"eye cream"*.

To conclude, the CrowdFlower dataset is also a robust dataset that can be used on eCommerce relevance problems. Due to its characteristics, it was also selected to be used to test the assumptions described in this document.

Table 3.1 compiles the main analysis obtained from both datasets.

Table 3.1: Comparison between the two datasets.

Metric	Home Depot	CrowdFlower
Average relevance	2.38	3.25
Average number of products per query	7.68	123.75
Average query per product	1.74	1.10
Average number of words on each search query	2.15	2.40
Number of unique product-query pairs	186131	30720

After this analysis, some conclusions can be taken regarding the possible performance of the analysed datasets. Both datasets present low connectivity, which means that there is a small number of products that are connected to more than one query. The main goal of the tasks being performed and described on this document has the intent of transferring information from the product title and description (if the following information is available) to the search term and vice-versa. Given the fact that the datasets present low connectivity, the obtained results may not be representative of the implemented methods. For this reason, a more in-depth analysis was performed on the results section.

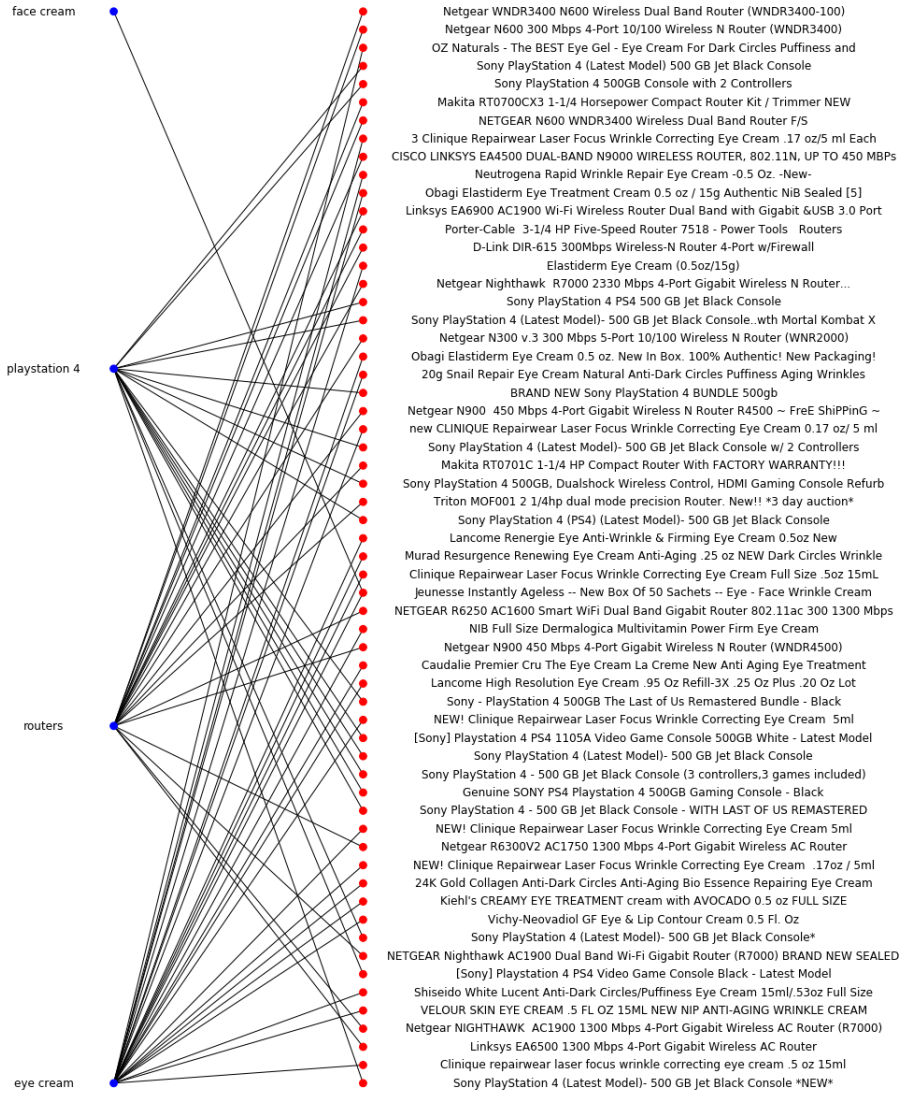

Figure 3.6: Example of the graph structure of the CrowdFlower dataset.

Chapter 4

Design and Experiment

The main goal of this chapter is to describe design decisions and correspondent implementation details for the methods developed in this work.

First of all, the reader will be able to see and analyse an overview of all the artefacts created. Then, a detailed description of the implementation and models used is provided.

4.1 Overall Design

As stated previously in Chapter 1, the main goal of this work is to use graph-based methods capable of connecting products with queries and experiment with different natural language models to observe the differences in the results.

With the intent of accomplishing the primary goal of this work, a workflow was designed and implemented. The design of this workflow took into consideration the fact that there was a necessity to make the entire project configurable since all of the results will be analysed on two different datasets that can have different characteristics and variables. For this reason, it was essential to consider that both experiences could be somewhat different and would need different thresholds. Figure 4.1 contains a flowchart with the workflow designed and implemented during the extension of the current work.

As it is possible to observe from the analysis of Figure 4.1, there are several steps necessary to accomplish the final evaluation of the methods implemented. For starting the process, human interaction is necessary to make sure that all of the configurations are correct for the wanted experiment.

After initialising the process, the data being used for the experiment will be read, and the process will begin. The first step is feature engineering, where the *click_score* variable (described in section 4.2) is generated. Afterwards, the data will go through the process of data cleaning on which all of the processes explained in section 4.3 will take place. The combination of those two steps is considered the pre-processing stage of the workflow.

Following those preliminary steps, the next executed action is to transform the product titles and search terms into machine language. With the intent of accomplishing this step, the user will be able to configure one of the three natural language models available for the current work: TF-IDF, Word2Vec and BERT. The details of each language model implementation and the reasoning behind choosing them will be explained further in Section 4.4.

The last step is the creation of the bipartite graph, leading to the passage of information between each of the connected nodes. In this step, a method similar to the click graph method

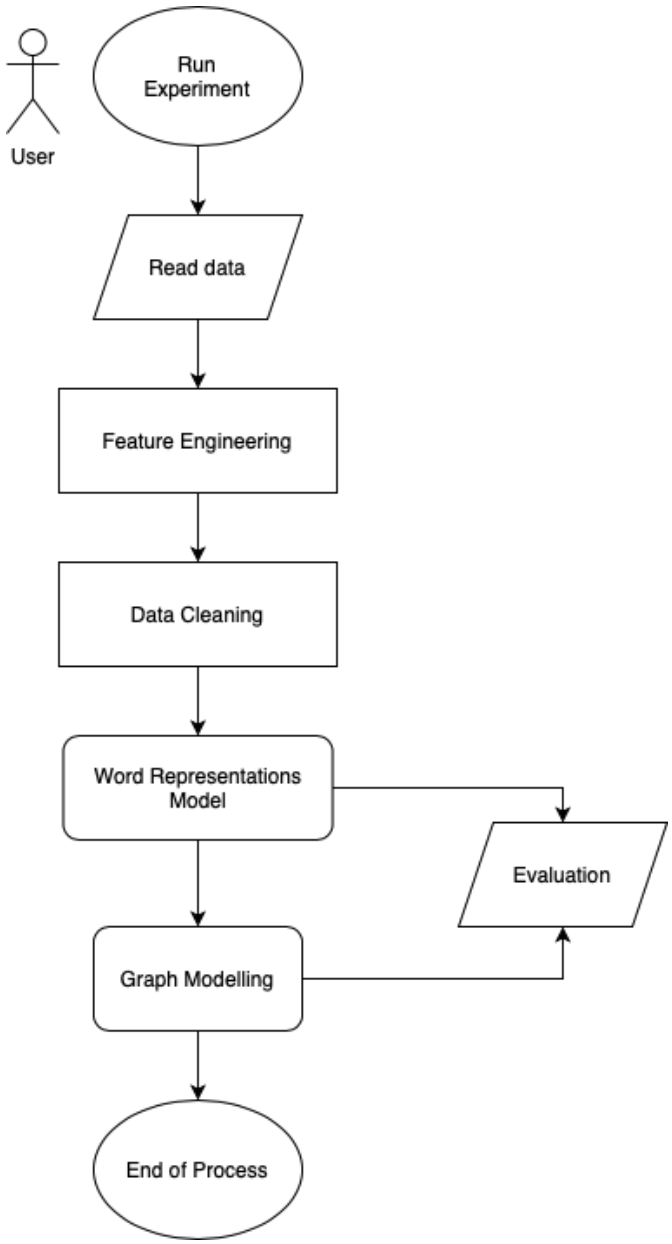

Figure 4.1: Workflow designed.

presented by Jiang et al. (2016) was used. Since there is not an available implementation of this method, it was necessary to recreate the steps described in the paper to accomplish the experiments. Later on this chapter, the reader will be able to observe and analyse the interpretation of the mentioned paper and implementation steps of this method.

Succeeding each of the last two steps, the evaluation of the results will take place. This step contains the implementation of all of the necessary metrics to evaluate the methods. The results of this step are described in Chapter 5.

As a final note, it is essential to state that all of the experiments were conducted on a single laptop, a 2018 Macbook Pro with 16 GB of RAM, Intel Iris Plus Graphics 655 and a processor of the type Quad-Core Intel Core i5 (2.3 GHz of processing speed, four cores, and 6MB of L3 Cache).

4.2 Feature Engineering

Ranking relevance datasets are usually manually annotated in a continuous or binary scale, depending on the wanted output. Continuous relevance indicates how the products are relevant to the search query, while a binary scale indicates if the products are relevant or non-relevant (Manning et al., 2008). This indicator assists in the process of ordering lists of products by how relevant they can be considering the context, their attributes, and how well they can be displayed together.

As stated in the previous Chapter 3, both datasets used to analyse and experiment the current work are not entirely suited to be used as a click-through log data. However, some simulations led to the creation of new features, that pointed to new and more suitable datasets for the tasks being performed. The main goal of this step is to create a new feature capable of correlating clicks with a given product's relevance.

Some previous studies try to simulate human behaviour in order to find what users are looking for and how they search for products (Bag, Tiwari, & Chan, 2019; Hendriksen, Kuiper, Nauts, & Schelter, 2020; Lo, Frankowski, & Leskovec, 2016). Even though the primary goal of the cited studies is somewhat different, there is a similar point in all of them, users want to find the most relevant products on the top positions, and they end up clicking more on those products. Although, it is possible to infer clicks from the relevance labels present on both datasets (considering the previously cited studies) it is necessary also to consider that this new feature should not be used on real-world problems since it can lead to problems such as position bias (Aslanyan & Porwal, 2019; X. Wang, Golbandi, Bendersky, Metzler, & Najork, 2018).

On eCommerce search engines, two major components should be taken into consideration when using user feedback (such as clicks) as an indicator of how relevant the given product is to the query. Position bias is one of them since users are more likely to engage and click on products that are ranked higher. The other one, presentation bias, does not have an impact on the current work, since we only have access to what was shown to the user. However, when building a search engine, it is necessary to also take into consideration the presentation bias, since users are only going to engage with products that are being shown to them (Aslanyan & Porwal, 2019). Both of these types of bias must be taken into consideration, especially if the main goal is to increase the serendipity and novelty of the results.

For the matter of this study, it was necessary to create a feature capable of simulating user behaviour considering the relevance scale present on the datasets. However, it is not possible

to simulate real user behaviour since many factors might lead to a user clicking on a given product or not. Nonetheless, it was still taken into consideration several of those factors like the fact that users tend to click on the first products being shown since they are usually the most relevant ones.

In order to accomplish this new feature, it was necessary to establish a relevance value that would be considered as the separation between the most clicked products and the rest of the catalogue. In this scenario, the relevance label is a proxy for the number of clicks that a specific product would have if it were matched to a query. The first assumption made is that the more relevant products should be ranked higher, which would lead to more clicks and interactions from the users.

Several distributions were considered when designing the task since it is not expected for the real values to follow a random click distribution. However, it was also not the intent of this work to ensure that the distribution of clicks would follow a clean pattern. Since the dataset analysis performed on Chapter 3 showed that a high percentage of products paired with the queries were considered to be of high relevance, it was established that the relevance threshold would be value correspondent to the percentile 60 of the relevance label, of each dataset. The products that have a relevance higher than the relevance value on the percentile 60 are considered to be the most clicked products. For the sake of the exercise, it was established that the products with a relevance lower than this percentile would have a maximum of ten clicks, and the products with a higher relevance could have between 10 and 100 clicks. After this attribution, the click values are normalised between zero and one. One corresponds to a higher probability of that product being clicked and zero means that the product was not clicked when showed as a response to the query. In Figure 4.2, it is possible to analyse the distribution of this new feature on both datasets.

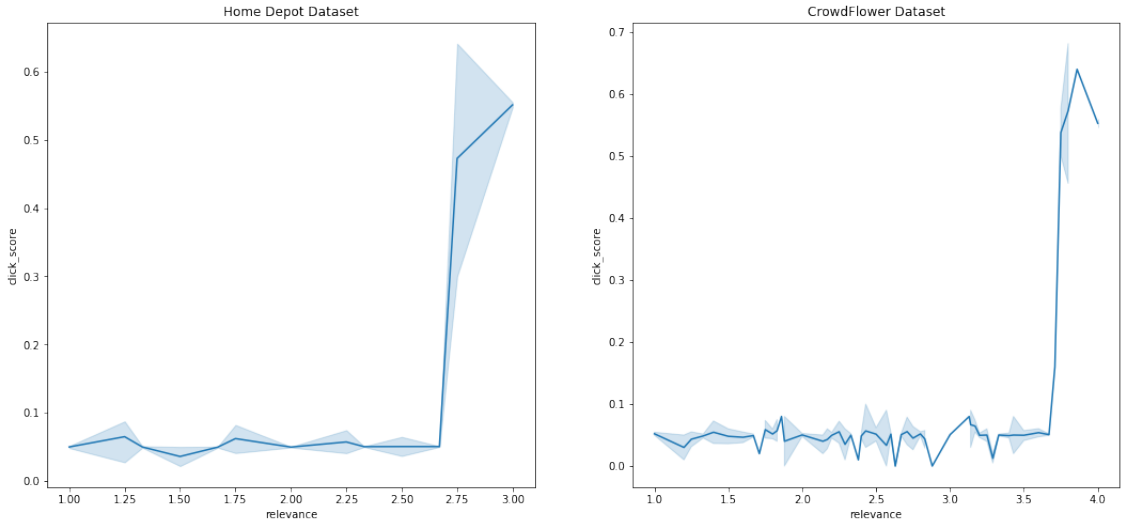

Figure 4.2: Distribution of the *click_score* feature of the Home Depot dataset (left) and the CrowdFlower dataset (right).

As it is possible to state from the analysis of Figure 4.2 there is a high gap between the relevant and non-relevant products. On average, the non-relevant products have a *click_score* value lower than 0.1. As previously mentioned, on real data we would not see such gap between the two types of products, since users may click on lower-ranked products only

because they found it interesting but not necessarily because it was a good match to the search results.

Due to this work time-frame, and the limitations of the available datasets, this distribution was kept. The main reason behind this decision was the low connectivity between products and queries, which means that there is a small number of products that are connected to more than one query. Due to this, it was decided to turn the relevant products even more relevant to compensate. However, this decision also has downsizes, since there is the possibility of creating a higher semantic gap between the train and test set. On chapter 5, all the assumptions and decisions are taken into consideration when evaluating all of the processes.

4.3 Data Cleaning

As already stated in Chapter 2, queries are usually formed in a natural human format, meaning that they may be filled with unnecessary or complicated words that do not have an impact on the expected output. Given this, one of the necessary steps when working with natural language processing tasks is data cleaning. This section has the intent of enumerating and explaining the textual pre-processing done on top of both datasets.

Data cleaning is a process that consists of removing any non-necessary information present on the original dataset. The main goal of this procedure is to extract any noise and irrelevant data that may lead to unpredictable results. When taking into consideration textual data, there is the necessity to be extra careful since a small change in one of the characters can lead to different representations. Figure 4.3 describes the flow of this process.

On the next subsections, it is possible to find an explanation of each step of the flowchart of Figure 4.3. Alongside with its implementations and expected outputs.

4.3.1 Removal of Non-ASCII Characters

The first step of this processing has the intent of removing all characters that are not encoded correctly according to the American Standard Code for Information Interchange (ASCII). After some analysis, it appears that some of the text present on the datasets contains strange characters that do not belong to the original terms. Even though there was extra care when first reading the original files, it was not possible to retrieve the original words with the correct encoding. For this reason, there is a high percentage of terms in both datasets that are not compliant with the ASCII encoding. The impact of this characteristic is that when looking at them, there are some letters attached to the original words. In order to remove those types of characters the Python built-in functions *encode* and *decode*[1] were used. Figure 4.4 has a real example of a product title from the CrowdFlower dataset. On this case, the word *Nespresso* contains more characters than it should have.

After making the textual data compliant with the ASCII character encoding, it is possible to state, by observing Figure 4.4, that all the characters that were not compliant with the rules were removed, leaving the word clean of any non-compliant text.

Since this mismatch originated all the accents existent on the dataset, there is no necessity to strip the accents from the terms. However, if the dataset still contained terms with diacritical marks, this extra step would be necessary. However, this would be unlikely since both datasets are in English. Nonetheless, to accomplish this the Python package named

[1] https://docs.python.org/3/library/codecs.html#standard-encodings

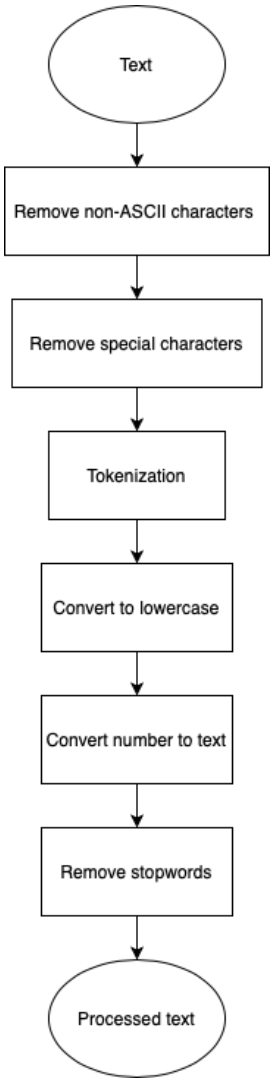

Figure 4.3: Data cleaning flowchart.

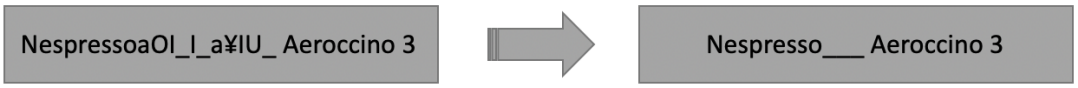

Figure 4.4: Example of the pre-processing step responsible of removing not ascii characters.

unicode.normalize [2] could be used. The Unicode package allows several types of normalisations that are based on the definition of the canonical and compatibility equivalences, for this work was used what this library calls the normal form D (NFD). On this method, also known as the canonical decomposition, every character is translated into its decomposed form.

It is important to note that even if two strings are normalised and look the same to the human eye, this does not mean that they will be equal when compared by machines, one of them still might have characters combined or similar.

4.3.2 Removal of Special Characters

The second step of this process consists of removing the special chars that might be present in the text. The main goal of this step is to remove any special characters that may be within the textual data. Since it is only necessary to obtain the textual meaning of the terms, those characters are not necessary to understand the customer search intent nor will be relevant when considering the product titles. To achieve this, it was created a list of unnecessary characters that should be removed from the data. The following table (Table 4.1) contains the list of characters to be removed from the all of the textual data being used on the current project.

Table 4.1: List of characters to be removed from the queries and product titles.

Characters type	Characters
Punctuation	. ? ! , ; :
Dashes	/ — _ - |
Brackets	( ) { } [ ]
Quotation marks	” ’ ‘ ’
Mathematical Symbols	>< + * %
Special	$ & « » # ° ª º

As previously stated, there is a possibility that most of the characters listed in Table 4.1 do not belong to any of the sentences present on the datasets used. However, in order to make sure that if they exist, they will be removed, they were added to the list. After this step, all of those characters are replaced by a single space. An example can be see on Figure 4.5, in which the input given was the output from the previous pre-processing step (Figure 4.4).

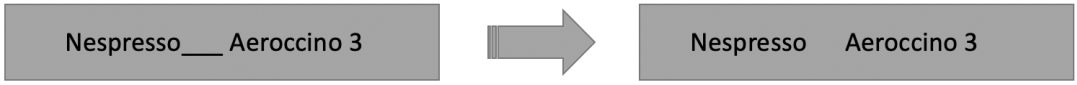

Figure 4.5: Example of the pre-processing step responsible of removing special characters.

From the Figure 4.5 it is possible to conclude that all additional characters (present on the Table 4.1) are removed and substituted by a single space. The space of those characters are being substituted and not completely deleted since some product names or queries may contain the listed characters. On those cases, we want to keep all of the terms but separate from each other.

[2]https://docs.python.org/3/library/unicodedata.html

Now the product title being analysed is much more readable (both by humans and machines), nonetheless there are still some steps missing from the final output.

4.3.3 Tokenization

The third step of this pre-processing consists of splitting all the words present in the sentences by doing tokenization. The tokenization is a splitting process that allows the break of a string of text into smaller pieces, called tokens (Gupta, 2019). Since words, numbers and even punctuation marks can be considered as tokens, it is important to remove the unwanted characters previously to this step, since there is not the need to continue processing information that is not relevant to the final output. With the intent to accomplish this process, the Python package *nltk.word_tokenize* was used[3]. Continuing with the same example, in Figure 4.6 is possible to see what happens to a product title when this step is applied.

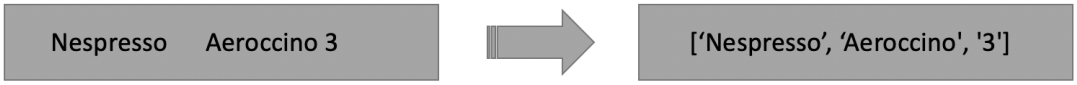

Figure 4.6: Example of the tokenization process during pre-processing.

From the analysis of Figure 4.6, it is possible to notice that three tokens were created from the input sentence. This step has a significant contribution to the performance of a natural language processing model since it allows the identification of the words that are part of the original text. The importance of this step relies on the fact that text analysis and interpretation is straightforward when only considering the words present in the said text. This step is responsible for preparing the data to remove stopwords and meaningless words, such as '*is*', '*want*', '*buy*' and many more. On this point, another analysis was conducted. After accomplishing each processing, there is a need to re-observe the data to make sure that the data is ready for the next steps.

4.3.4 Conversion to Lowercase

After validating that the tokens are correctly formed, the fourth step of the data pre-processing consists in turning all the tokens into lower case terms. To accomplish this, the Python built-in function *string.lower*[4] was used. Figure 4.7 contains an example of how this method works.

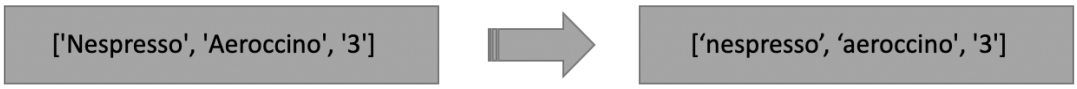

Figure 4.7: Example of the lower casing process during pre-processing.

From the analysis of Figure 4.7, it is possible to observe that the final output is a list of tokens, in which all of the textual terms are in lowercase. This step is one of the most important because there is a guarantee that the textual representations being calculated downstream

[3]https://www.nltk.org/api/nltk.tokenize.html
[4]https://docs.python.org/2/library/string.html#string.lower

are going to have the same value for the same term. For instance, for a machine, the word *Nespresso* have a somewhat different representation from the word *nespresso*. For a human reader, both words are the same since they have the same meaning, but this is not the case for the machine. With this step, it is possible to guarantee that independently of how the terms are being written, they will be seen as the same.

4.3.5 Conversion of Numbers to Text

Through experimentation and analysis, it is possible to conclude that some of the pre-trained language models are not able to translate integers to its respective representation. For this reason, it is necessary to process the numbers present either in the queries or in the names of the products since they can be relevant for the matter. On the example being used on this section, the number is important for the search results, since the user is explicitly wanting do find that model of the Nespresso machine, so there is a need to keep this relevant information. Apart from these cases, the Home Depot dataset is also filled with even more prominent cases. This dataset contains gardening materials in which the size and diameter of the tools are also crucial for the search intent. To accomplish this step the Python package *inflect*[5] was used. Figure 4.8 contains an example of how this process works.

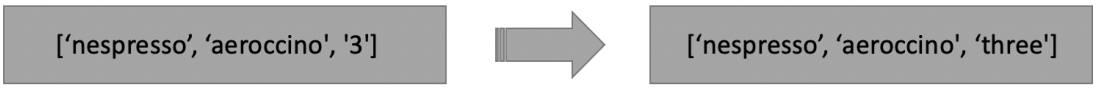

Figure 4.8: Example of the transformation of numbers into text during pre-processing.

As the reader can notice in Figure 4.8, the number contained on the example is converted to its long textual form. For some of the natural language processing models, this step is not necessary since these are trained from scratch with the textual corpus of the train dataset. Another way of dealing with pre-trained models is to expand their vocabulary, but this will not be considered during this work due to time constraints.

4.3.6 Removal of Stop Words

The last step of this process is the removal of stop words. Those words do not add meaning to the sentences and can be removed without changing the meaning of the text. When taking into consideration search engines, some words can introduce noise on some queries. To prevent this from happening that kind of words were removed, by using the Python package *nltk.corpus.stopwords*[6] for the English vocabulary was used.

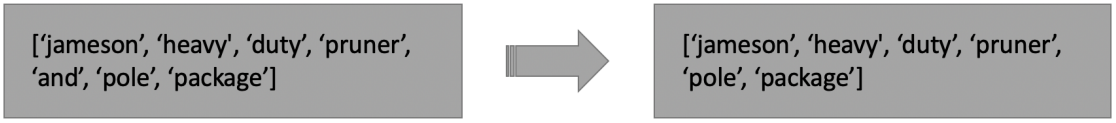

Figure 4.9: Example of the stop words removal during pre-processing.

[5]https://pypi.org/project/inflect/
[6]https://www.nltk.org/book/ch02.html

From the analysis of Figure 4.9, it is possible to visualise that the word '*and*' was removed from the input text. When considering a natural language model, this word is not adding any additional information so that it can be removed.

In conclusion, these were the steps performed to clean the queries and product titles present on both datasets. The methods and processes applied were built, taking into consideration the type of data available and what was necessary to clean it.

4.4 Word Representations

This section describes the language models used, how they were implemented and their respective configurations. During this work, three types of natural language models were used: TF-IDF, Word2Vec and BERT. Those models were chosen due to the differences between each implementation and architectures. Since one of the main goals of this work is to compare the results obtained by combining the proposed graph-based approach with different types of word representations, it is essential to experiment and analyse different types of models that can lead to entirely different results.

The TF-IDF model was chosen due to its simplicity. This model is easy to understand and also has a good performance on word matching tasks.

Regarding Word2Vec, this model was chosen since it is possible to retrieve the relationship between words from the resultant embeddings. Besides that, this model does not need significant amounts of storage or computational power, which means that the experiments will not be time-consuming and this is a big advantage since it allows the repetition of several experiments in a fast manner.

BERT was mainly chosen because of its growing popularity and the fact that this model is capable of generating embeddings that are context-sensitive. It is also important to note that BERT was chosen over the XLNet model, because of the computational power differences. XLNet is a model that requires high computational power and memory, which raised some *a priori* concerns since there was a possibility that the computer in which all of the experiments ran was not capable of handling this model processing.

Each one of the models is unique and has different ways of dealing with text. The main goal of the following subsections is to explain the process of building and configuring each of the models experimented during this work.

4.4.1 TF-IDF

The TF-IDF (Term Frequency-Inverse Document Frequency) method was already introduced on the Section 2.2.1. As already stated, this is one of the classical models used to perform tasks such as natural language processing. Besides being a reasonable simpler method, it is still being used, and on some tasks it has a good performance.

With the intent to implement this model, it was necessary to create the TF-IDF model by training it with all of the corpus existent from the train set. In order to achieve this step, the previously processed product titles and search terms were merged into one single corpus, and this was considered the training data for the model. For this model in particular the Python library *sklearn*, more specifically the call to the *TfidfVectorizer*[7] method, was used.

[7]https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

The TF-IDF method has the characteristic of having a unique representation for each of the words present on the corpus. Consequently, the Home Depot dataset word representations contain 21420 dimensions, and the CrowdFlower dataset vectors contain 19609 dimensions. The size of the vector representations is intrinsically connected with the number of unique words/terms present in the data.

4.4.2 Word2Vec

As already introduced on section 2.2.1, the Word2Vec model is capable of creating a vector representative of the distributed numerical representations of the word features (Mikolov, Yih, & Zweig, 2013). Since this type of models is very common and can be expensive to train, especially if there is a need to contain a vast vocabulary, there are already pre-trained models available. For the current work, the Word2Vec trained on the Google News corpus[8] was used, this model was built with 3 million words and phrases.

Differently from the TF-IDF model, Word2Vec output dimensions can be configurable, and there is no ideal number for it since it can depend on the application and the tasks being performed. Since a pre-trained model for 300 dimensions was used, the word vectors will also have that dimension if they belong to the vocabulary. Otherwise, that word will not have a valid representation.

This model outputs a vector for each word, so it is necessary to combine those vectors to retrieve only one representation per product title or search term, since there is the need to have the entire sentence representation and not only the values of each word. From the analysis of several studies, it is possible to conclude that there are some common ways to combine the embeddings vectors, some of them are the following:

1. Sum of the word embeddings (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013b).
2. Mean of the word (Artetxe, Labaka, Lopez-Gazpio, & Agirre, 2018; Q. Le & Mikolov, 2014).
3. Concatenation of the word vectors with the context (W. Yin & Schütze, 2016).

After analysing the results and the implementation detailed of the cited papers, it was possible to notice that the sum and mean of the word embeddings were the more straightforward solutions and both presented excellent results. Due to the simple implementation, it was decided to try both approaches and analyse the results obtained from both methods when conducting the experiments.

4.4.3 BERT

From the introduction made on the section 2.2.1, BERT represents a trending encoding technique that models some of the most recent state-of-the-art results on NLP tasks, with a focus on learning the context of the words based on the surrounding text (Devlin et al., 2018). This type of model is expensive to train and to use, so it was necessary to adopt an already pre-trained model. Besides that, BERT is a transformer model, which means that it was designed to handle sequential data and attend to the different positions of the input

[8]https://code.google.com/archive/p/word2vec/

sequence with the intent of computing a representation of that sequence (Vaswani et al., 2017). For this reason, the transformers models are high performant when they are trained on significant amounts of data. As a result of this pre-requisite, a pre-trained model provided by *Google*[9] was used. Similarly to Word2Vec model, since the BERT model being used is also pre-trained, the output dimensions are also dependent on how the model was trained. For the current model being used, the output embedding vector contains 768 dimensions, but on this model, it already represents the sentence embedding and not the embeddings of the individual words.

Using a pre-trained model can greatly reduce the time that the experiments take to run. However, it is still processed at a high cost when compared with the other models. The embeddings predictions of the training corpus took approximately five hours for the Home Depot train set and almost two hours for the CrowdFlower data, on the machine specified at the beginning of Section 4.1.

Due to the lack of computational power, it was not possible to experiment with more complex networks since the process would be killed due to lack of memory. Nonetheless, it was still possible to conduct experiments (that will be described in Chapter 5) with simpler architectures of the BERT and retrieve the output of the embedding. For the experiments, it was adopted the DistilBert model (provided by HugginFace[10]) as it has an available pre-trained model that meets the available computation power. Although more complex and larger models are available, it is believed that the selected model can represent the ability to approach with reliability.

4.5 Network Representation Learning

It is important to note that regarding the graph-based approach, it was decided to create a method similar to the one presented by Jiang et al. (2016), denominated as click graph. The main reasoning behind the choice of this method is the fact that it allows the connection between two different types of nodes: products and queries. When comparing this technique with the other models described on 2.2.2, this one presented advantages and disadvantages relative to the remaining algorithms - which were also described in the same section. Even though it was intended to experiment with more recent models, such as BiNE, it was not possible to modify the datasets to incorporate user data. Otherwise, the datasets would become even more sparse than what they already are, which can lead to inconclusive results.

Nonetheless, the fact that click graph approach does not have an open-source implementation it is an advantage since it created an opportunity to study the method in detail. Therefore, it was possible to code and execute this technique from scratch.

The following subsection describe the implementation of the proposed method, its design and the similarity calculations performed during the network search.

4.5.1 Bipartite Graph Information Transfer, BiGIT

According to Jiang et al. (2016), the main goal of the click graph is to propagate the information between the two different nodes of a previously created bipartite graph. The algorithm has the intent of learning the vectors representations based on the content and

[9] https://github.com/google-research/bert

[10] https://github.com/huggingface/transformers

click information available from the data. The authors state that the proposed method is capable of improving the ranking performance since there are records present on the click log. The connections end up being stronger once the click graph algorithm is applied, which can be seen by analysing the training set results. Regarding queries or products that are not present on the training set - Jiang et al. (2016) calls them *click-absent* -, the authors propose a two-step vector estimation in which they generate the word representations based on the vectors that were already created by the click-graph propagation.

Regarding the implementation of the first step, the execution of the data propagation method is straightforward. From the paper presented by Jiang et al. (2016), it is possible to represent the documents based on their relevant queries and vice versa. On this case, the clicks are indispensable indicators of the relevance of an item or query.

In a simplified way, it is possible to state that this method contains three main steps (Jiang et al., 2016):

1. Initialisation of the vectors. The method can start from the queries or the products.

2. The clicks weights the previously initialised vectors. After this, the information is propagated to the connected nodes on the opposite side of the graph.

3. On the following iteration, the same happens, and the vectors are propagated to the other side.

All of these steps are repeated until convergence. In regard to the information propagation, it depends on which side the first vector is initialised. If the algorithm starts on the product side, the initialised vector matrix is denoted as $D^{(0)}$. On the *n-th* iteration ($n \geq 1$), the query vectors $Q^{(n)}$ are computed by using a weighted sum of the co-clicked documents in which the weight depends on the number of clicks. On the specific case of this work, the click weights will be equal to the *click_score* variable explained on the section 4.2.

According to Jiang et al. (2016), given the *i-th* query q_i, its correspondent vector representation Q_i^n is calculated as:

$$Q_i^{(n)} = \frac{1}{\| \sum_j^{|Doc|} C_{i,j} \cdot D_j^{(n-1)} \|_2} \sum_{j=1}^{|Doc|} C_{i,j} \cdot D_j^{(n-1)} \tag{4.1}$$

In which, D_j^{n-1} is the vector representation for the *j-th* document in the iteration $(n-1)$. Then the document vectors, $D^{(n)}$ are also updated accordingly to their co-clicked query vectors:

$$D_j^{(n)} = \frac{1}{\| \sum_i^{|Query|} C_{i,j} \cdot Q_i^{(n)} \|_2} \sum_{i=1}^{|Doc|} C_{i,j} \cdot Q_j^{(n)} \tag{4.2}$$

As stated by the authors, with this propagation, it is possible to learn the vectors for both queries and documents. This method also allows the reduction of the lexical gap between both sets of terms, leading to a improved query-document relevance.

As previously noted, this method does not have an open-source implementation available. The goal of this work is to develop an adapted approach similar to the click graph. With this

idea in mind, the Bipartite Graph Information Transfer (BiGIT) was created. The code was made available as a public repository in GitHub[11].

BiGIT implementation contains several similarities to the original paper. For instance, it was kept the logic behind the knowledge transference between the nodes of the bipartite graph.

The first step of this method is to create the bipartite graph from the information present on the dataset. For this the python package *networkx*[12] was used. In this stage, it is created a simple *networkx* graph, and then the product and queries are added as nodes. After this, the weighted edges are included on the graph by the usage of the *click_score* feature defined on Section 4.2. This newly created graph will be used for the remaining of the method.

Afterwards, the knowledge transfer between both type of graph nodes will be initiated. Differently from the click graph description present on the paper, BiGIT does not have a stop method. On the proposed approach, the number of iterations in which the propagation will happen is coded as one of the configurations that can be changed and modified accordingly to the wanted output. Nonetheless, it is possible to configure the starting point of the method. The user can change the configuration responsible to chose if the propagation-based method will start on the product or the query side of the bipartite graph.

Subsequently, the propagation method starts. On this phase of the implementation, the mathematical equations present on 4.1 and 4.2 (that were retrieved from Jiang et al. (2016)) were the foundation for the implemented approach. In this stage of the process, it is necessary to have the bipartite graph created alongside with the vector representations for the queries and products. In Algorithm 1 code block, there is a *pseudocode* implementation of the mathematical expression described in Equation 4.2.

Algorithm 1: Knowledge transference from queries to products in *pseudocode*.

```
Input: Q: Query vectors
       D: Document vectors
       G: Bipartite graph
Output: Q: Query vectors
        D: Document vectors
for word_vector in D do
    update = array[length(word_vector)]
    total_weight = 0
    for position, value in G[word_vector] do
        update = update + value['weight'] × Q[position]
        total_weight = total_weight + value['weight']
    if total_weight <> 0 then
        D[word_vector] = D[word_vector] + (update / total_weight)
        D[word_vector] = D[word_vector] / 2
```

As it is possible to state from the analysis of the *pseudocode* from Algorithm 1, there are two main components responsible for the knowledge transfer to the product vectors: the vectors from all of the queries connected to those products and the edge score associated

[11] https://github.com/isabelchaves/BiGIT

[12] https://networkx.github.io

with the connection - in this work, the edge score is the feature *click_score*, but it can also be the number of clicks that the product had when paired with the given query. For each word vector is calculated the *total_weight* value, that is the sum of the weighted edges connected with that node. Besides that, the sum of the query vectors times the score of the connecting edge is also computed. Finally, the mean between the document query and the resultant query vector (weighted by the *total_weight* value) is calculated, and the obtained value is the new vector representation for that given product.

As it is possible to state, this work does not take into consideration all of the implementation details described in the original paper (Jiang et al., 2016). One of the most significant changes is in how the word vectors are being initialised. The cited paper retrieved its word vectors by weighting the words in proportion to their frequency on the corpus and normalise it with the L2 norm. This work has a higher focus on understanding how the previously described methods of word embeddings perform when using the clicks information on a graph-based method.

After updating the graph, it is necessary to evaluate if the connections are well established or if there are errors on the updated nodes. Gomma and Fahmy (2013) presented a survey on all of the existent methods to measure text similarity. The authors present methods capable of measuring five types of similarities: characters, terms, corpus, knowledge and hybrid methods. All of the approaches contain differences according to the type of similarity being measured and the expected output.

After some investigation and analysis of several papers, it was decided to use the Python package *NMSLIB*[13] (*Non-Metric Space Library*) introduced by Boytsov and Naidan (2013). The authors were able to create a useful toolkit that allows searching on generic and non-metric spaces, and the evaluation of several similarity search methods (Boytsov & Naidan, 2013). This package allows the creation of the vector spaces alongside with a simple way of calculating the vectors similarity since it already has those methods implemented.

With the intent of measuring the similarity between two vectors the cosine similarity was used, which is a configuration for *NSMLIB* package. Mathematically, this method measures the cosine of the angle between two vectors that were projected into a multi-dimensional space and can be defined by the following equation (Manning et al., 2008):

$$similarity = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{4.3}$$

A and B are two non-zero vectors, and θ is the angle between them. The value of i represents one dimension of the vectors. On this work, each dimension represents a word or term of the query or product title.

This method takes into consideration the total length of the vectors for the similarity calculation, instead of only looking for the terms separately (Manning et al., 2008). This fact is really important because the main goal of the BiGIT method is to reduce the lexical gap between the queries and the products. For this reason, it is important to use a method that is capable of evaluating the entire word vector, rather than analysing the terms individually.

[13] https://github.com/nmslib/nmslib

Chapter 5

Experiment Results

This chapter has the objective of explaining all of the experiments performed during this work. It defines the metrics being used, the tested cases and the results of all of the tests performed.

The obtained results are thoroughly analysed, taking into consideration the configurations described in Chapter 4. The goal of the analysis is to understand the advantages and disadvantages of each method.

5.1 Evaluation Metrics

The accuracy of an information retrieval system does not only revolve around the notion of relevant or non-relevant documents; the user judgment and context must be taken into consideration. A document is relevant if it answers the user needs, but it is not required to contain all of the words of the initial query, as stated by Harman (1992). The only way to evaluate an information retrieval system is by measuring the effectiveness associated with the relevancy of the retrieved items. The main goal of this section is to present and discuss the metrics used to compare the different approaches and evaluate the scenarios studied.

5.1.1 F-Score

F-Score is a way of combining the results of precision and recall metrics from the model (Sasaki, 2007). Those are considered to be the most frequent and simple measures that allow the evaluation of the effectiveness of an information retrieval system (Manning et al., 2008). When evaluating a system capable of retrieving a set of documents for a given query, it is necessary to acknowledge two concepts: retrieved and relevant. The first refers to the retrieved documents, and the second considers the document's relevancy for the query. Using these concepts, it is possible to examine the contingency table present in Table 5.1 (Goutte & Gaussier, 2005).

Table 5.1: Contingency table of the retrieved documents' relevancy.

	Relevant	**Non Relevant**
Retrieved	True Positive (tp)	False Positive (fp)
Not Retrieved	False Negative (fn)	True Negatives (tn)

With the information present in Table 5.1, it is possible to define the precision and recall in terms of documents retrieved and relevant. The precision corresponds to the documents retrieved that are indeed relevant and is expressed by the following mathematical expression (Manning et al., 2008).

$$\begin{aligned} Precision &= \frac{\text{number of relevant documents retrieved}}{\text{number of retrieved items}} = P(relevant|retrieved) \\ &= \frac{TP}{(TP+FP)} \end{aligned} \tag{5.1}$$

Recall corresponds to the relevancy proportion of the retrieved documents. (Manning et al., 2008).

$$\begin{aligned} Recall &= \frac{\text{number of items retrieved}}{\text{number of relevant items}} = P(retrieved|relevant) \\ &= \frac{TP}{(TP+FN)} \end{aligned} \tag{5.2}$$

According to Manning et al. (2008), taking both metrics into account brings value because in some problems, one of the metrics is more important than the other. For instance, on some search engines, there is the need to retrieve only the most relevant items (such as web search engines). Opposingly, there are systems in which there is a higher need to contain all of the possible search results even when some of them are not entirely relevant for the search queries. The precision increases in an inversely proportional manner to the number of documents. The F-score metric is capable of measuring the trade-off between precision and recall (Manning et al., 2008). This metric can be expressed as follows:

$$\begin{aligned} F_1 &= \frac{2}{\frac{1}{recall} \times \frac{1}{precision}} = 2 \times \frac{precision \times recall}{precision + recall} \\ &= \frac{TP}{TP + \frac{1}{2}(FP+FN)} \end{aligned} \tag{5.3}$$

The highest possible value for F_1 is one, and it indicates a perfect balance between precision and recall. If the trade-off between these two metrics is tending to one of them, the value of F_1 will quickly decrease (Manning et al., 2008).

5.1.2 Mean Average Precision, MAP

In order to understand how to calculate the Mean Average Precision (MAP), it is necessary to know the meaning of the Average Precision (AP). As already mentioned, precision and recall metrics are based on the entire list of documents that are returned by a system. Once a ranking system is being considered, the order of the documents lists must be taken into

deliberation. The AP is a metric able to compute the average value of the precision ($P(i)$, in which i is a document) for the entire list (Craswell & Robertson, 2009), that can be retrieved by the following:

$$AP = \frac{\sum_{i=1}^{n}(P(i) \times rel(i))}{\text{number of relevant documents}} \tag{5.4}$$

Given that, the MAP is the arithmetic mean of the AP values for an information retrieval system for each query, in a set of n queries (Beitzel, Jensen, & Frieder, 2009).

$$MAP = \frac{\sum_{i=1}^{n} AP(i)}{n} \tag{5.5}$$

5.1.3 Mean Reciprocal Rank, MRR

In information retrieval, the Reciprocal Rank (RR) allows the calculation of the rank of the first relevant document retrieved, as stated by Craswell (2009). The Mean Reciprocal Rank (MRR) is the average of the RR across queries. This metric is associated with user models that states that the users only want to see one relevant document and assumes that they will only look through the ranking until they found the said document. When considering a set of n queries, this metric can be expressed as follows:

$$MRR = \frac{1}{|n|}\sum_{i=1}^{|n|}\frac{1}{rank_i} \tag{5.6}$$

5.1.4 Normalized Discounted Cumulative Gain, nDCG

The Normalized Discounted Cumulative Gain (nDCG) is a measure of ranking quality, its principle is based on top of the assumption that the highly relevant documents are more useful than moderately relevant documents, and the latter is more useful than the irrelevant documents (Järvelin & Kekäläinen, 2002).

The assumption of the Discounted Cumulative Gain (DCG) metric is based on the same statement presented above without the normalization. According to this metric, documents found towards the end of an ordered list should be penalized, and a logarithmically proportional reduction is applied to the graded relevance when compared to the position of the final result.

$$DCG = \sum_{i=1}^{n}\frac{relevance_i}{\log_2(i+1)} \tag{5.7}$$

The response of an information retrieval system can vary in length (depending on the query and the catalogue available) which makes impossible to use DCG alone as a comparison

of the system performance. By normalizing the results, it is possible to observe the results for all of the queries, since it scales the values based on the best result seen, known as the Ideal DCG (iDCG).

$$nDCG = \frac{DCG}{iDCG} \tag{5.8}$$

5.2 Results Analysis

In this section, a detailed results analysis was conducted with the intent of understanding the performance of the methods on the designed scenarios.

As previously mentioned, one of this work's goals is to understand if different word representations methods lead to different results - when combined with the knowledge transference performed by the BiGIT. The initialisation of this model can be conducted on both sides of the bipartite graph, which can lead to different results. For this reason, for each NLP model, a comparative analysis of its performance will be conducted, considering both sides of the graph as starting points. To that, the notation $Q \rightarrow D$ is used when the starting point is the query vectors, and $D \rightarrow Q$ when the starting point is the product vectors. As stated in Section 4.5, the number of iteration is considered to be one of the methods' configuration. Several experiments were performed to find the convergence of the method on both datasets used during this work. From this analysis, it was found that performing more than 100 iterations did not affect the result metrics, so this was the chosen value.

From the analysis of the feature engineering conducted on section 4.2, it is possible to state that at least part of the experiment is deterministic. When creating the new feature, *click_score*, there was the need to add a seed to the random component of the method. This seed guarantees that each time the algorithm is run, the same input will correspond to the same output. Nonetheless, the remaining steps are not deterministic, so with the intent of conducting a thorough analysis, each experiment was run three times, and the mean and standard deviation of each metric was calculated.

As previously described in Chapter 3, both datasets have significant differences in terms of their structure. Consequently, it is necessary to conduct a separate analysis of the results obtained from each dataset since they represent different test scenarios. During the next two subsections, the reader will find a thorough analysis of the results obtained, allowing a more transparent evaluation of the methods and the context in which they provide the most value. Nonetheless, the split between the train and test set performed was the same for both datasets. The test data only contains unseen queries.

It is vital to notice that all of the results were compared with the ground truth present on both datasets, the relevance label. For this reason, when calculating the evaluation metrics, the retrieved ordered list was compared with a list ordered by the relevance label - as presented in Chapter 3.

Finally, it is necessary to consider that some metrics are usually calculated on top of k sized lists, which is the case of the MAP and nDCG. However, neither dataset is composed of lists of products with a fixed length for all of its queries. Moreover, since the datasets are not extensive, not all products belonging to the list are retrieved as relevant when searching on a similar vector space. Thus, the value of k was not defined *a priori*. Those metrics are

taking into consideration the length of the products list that is actually being retrieved. Lastly, those records are compared with the expected list of products.

5.2.1 Home Depot

The Home Depot dataset, as already mentioned in section 3.1, does not contain highly connected data. On average, each product is connected with 1.74 queries, which can lead to mismatches or lousy representations resulting from the BiGIT model.

As previously stated, this analysis will be focused on the unknown queries, the test set. An analysis on the train set will not be conducted since it would only show the improvements done on top of the known queries to the system, which could not be translated into on an overall improvement for a real search engine. As already stated in Section 4.5, the graph-based approach can be initialized on both types of nodes: the documents nodes (D) or the queries nodes (Q). Therefore, all of the analysis conducted in this section will take into consideration both initialization points.

First of all, it was necessary to understand the overall performance of the models. For this reason, for each experiment, an analysis of the average number of products retrieved was conducted. This analysis had the intent of understanding which of the methods are capable of retrieving the most number of products per search query. Table 5.2 contains the materialization of this analysis. It is important to notice that this first analysis does not take into consideration the relevance of the retrieved products, only the number of products that are being retrieved on each experiment.

Table 5.2: Average number of retrieved products per query for the Home Depot test set.

Word Representation Model	BiGIT Configuration	Average number of retrieved products
TF-IDF	Without (Baseline)	**1.647**
	$Q \rightarrow D$	1.025
	$D \rightarrow Q$	0.913
Word2Vec (average of the word vectors)	Without	0.179
	$Q \rightarrow D$	0.132
	$D \rightarrow Q$	0.202
Word2Vec (sum of the word vectors)	Without	0.179
	$Q \rightarrow D$	0.088
	$D \rightarrow Q$	0.137
BERT	Without	0.180
	$Q \rightarrow D$	0.198
	$D \rightarrow Q$	0.268

According to the results presented in Table 5.2, it is possible to state that TF-IDF experiments are the ones that, on average, retrieve the highest number of products per query. The experiment that only uses the original word vectors for similarity comparison is the one capable of retrieving, on average, the higher number of products per query. The remaining use cases have a lower performance on this metric.

Regarding Word2Vec, there is not a high difference between the two combination methodologies. When comparing the results obtained by the plain Word2Vec implementation, the average number of products being retrieved by those two methods is the same. Nonetheless, the average number of products per query when averaging the word vectors is higher when combining the resultant word representations with the graph proposed approach. On this set

of experiments, it appears that better results can be retrieved when initialising the graph-based approach on the document nodes.

The experiments in which BERT is being used are the only ones that have better performance when combined with BiGIT approach. Independently of the node side in which the process is initialised, on average, both use cases have a higher number of products retrieved per query when compared with the option without the knowledge transfer between the two type of nodes. Nonetheless, an initialisation on the document side is the best one for this case.

One important takeaway from these results is the fact that when compared the results from the several word representation methods combined with the proposed graph approach, the starting point of the graph initialisation leads to different results. The experiments in which it is combined the Word2Vec and the BERT models with BiGIT, the higher number of products retrieved happens when initialising the graph propagation on the query side. However, the experiments with the TF-IDF methods show a different pattern. On this set of experiments, it is retrieved more products when initialising the BiGIT model on the document side.

After this analysis, the metrics results from all of the conducted experiments were retrieved. Table 5.3 aggregates those results. With the intent of performing the best analysis possible, all experiments were conducted three times, and it was retrieved the mean and the standard deviation of the outputs combined. With this procedure, it is possible to verify if there are significant differences between each run.

Table 5.3: Results for the Home Depot test set for the several experiments conducted.

Word Representation Model	**BiGIT Configuration**	**Metrics**							
		F-Score		**MAP @ k**		**MRR**		**nDCG @ k**	
		μ	σ	μ	σ	μ	σ	μ	σ
TF-IDF	Without (Baseline)	**0.3732**	0.0002	**0.2730**	0.0001	**0.5894**	0.0006	**0.3420**	0.0002
	$Q \rightarrow D$	0.2385	0.0005	0.1642	0.0004	0.4354	0.0010	0.2200	0.0004
	$D \rightarrow Q$	0.2134	0.0002	0.1456	0.0001	0.3994	0.0006	0.1970	0.0002
Word2Vec (average of the word vectors)	Without	0.0472	0	0.0291	0	0.1257	0	0.0464	0
	$Q \rightarrow D$	0.0325	0	0.0220	0	0.0622	0.0002	0.0299	0.0001
	$D \rightarrow Q$	0.0489	0.0002	0.0330	0.0002	0.0948	0.0004	0.0449	0.0003
Word2Vec (sum of the word vectors)	Without	0.0472	0	0.0291	0	0.1257	0	0.0464	0
	$Q \rightarrow D$	0.0224	0.0002	0.0144	0.0001	0.0495	0.0004	0.0209	0.0002
	$D \rightarrow Q$	0.0335	0.0001	0.0225	0.0001	0.0649	0.0001	0.0307	0.0001
BERT	Without	0.0474	0	0.0294	0	0.1219	0	0.0459	0
	$Q \rightarrow D$	0.0495	0.0001	0.0331	0.0001	0.0982	0.0002	0.0459	0.0001
	$D \rightarrow Q$	0.0666	0.0002	0.0448	0.0002	0.1294	0.0002	0.0615	0.0002

From the analysis of Table 5.3, it is possible to conclude that there are significant differences between each of the methods. The direct TF-IDF output (without the information provided by the BiGIT algorithm) is the one that presents better values in terms of all of the metrics analysed. Oppositely to what was expected, the metrics for the experiments executed with this word representation model become worst when applying the proposed graph algorithm. The fact that TF-IDF has an exceptional performance when considering word matching tasks can be the reasoning behind these results. One of the hypotheses driven from the results is the fact that the dataset does not contain enough data points to be able to reduce the lexical gap between the queries and products completely, thus not extracting the core properties of a bipartite graph algorithm. TF-IDF characteristics of being able to conduct a match between words retrieve good results, but the specific word must be present on both terms (query and product title), so the system is capable of retrieving the product.

Consequently, if the structure of the product titles and search terms is changed - by the transformation that happens during the BiGIT algorithm - the retrieved results will not match precisely and in some cases, some noise was introduced on the terms. Nonetheless, there is a difference between the results of the two starting points of the BiGIT initialisation. For the TF-IDF case, initialising the BiGIT on the query side is presenting the best results. On this case, the results present a higher mean average precision, the most relevant product is more closer to the top positions (higher MRR), and the final ranked listed presented fewer shifts when compared with the perfect rank.

Word2Vec model does not have the same impact on the results when compared with the metrics output from the TF-IDF experiments. In terms of metrics values, Word2Vec presents worse results when compared with the previous model. Overall, and for this specific situation, Word2Vec is not a high performative model, and it should not be used for this task with these specific parameters, since good results are not guaranteed. Both sets of experiments present lower metric values when compared with the optimal ones. Nonetheless, averaging the word vectors is the method that retrieved the best results from both experiences. When analysing the metrics obtained from the three experiments with this aggregation method, it is possible to observe that the model has different behaviour when applying the BiGIT method. By comparing the experimental results that use the graph algorithm, it is possible to detect a difference between the results with a starting point on the query side from the ones in which the initialisation happens on the document side. On this specific case, it is possible to state that when the implemented graph-based algorithm is initialised on the document side, the output is more performative. Even though, when comparing this experiment with the results obtained from the standard Word2Vec model, there are no big differences from both methods. It is possible to state that in terms of precision, the method with the BiGIT is presenting better metrics than the simple Word2Vec representation.

Regarding BERT, this model was also not able to outperform the results obtained from TF-IDF. When considering the word representations alone, the metrics are somewhat similar to the ones obtained from the experiments with the Word2Vec representation. However, when testing BERT word representations with the BiGIT algorithm, the results show a different pattern. The experiments conducted with BERT are the only ones in which the metrics improve by the application of the proposed graph method, independently from the starting point of the graph-based approach. When initialising the method on the query side of the graph, the improvements are not significant. However, it is possible to notice an improvement on all metrics when initialising the graph-based method on the document side of the bipartite network.

Due to the datasets' low connectivity, the performance of the most connected part of the graph was also analysed. Therefore, the dataset was searched for products that contain more than 20 connections. The queries connected to those products were retrieved, resulting in a list of 650 queries that were used to filter the test set. Table 5.4 contains a summary of the results for those queries.

When comparing the results obtained from Table 5.3 with the metric values present on Table 5.4 it is possible to state that the experiments conducted on top of the TF-IDF word vectors do not have better performance on the highly connected part of the graph. Contrarily to the overall results (provided on Table 5.3), the metric values are higher when those word vectors are combined with the graph-based approach initialised on the document side of the network. Nonetheless, the explicit TF-IDF word vectors are still resulting in the best values.

Table 5.4: Results for the Home Depot most highly connected queries from the test set.

Word Representation Model	BiGIT Configuration	Metrics			
		F-Score	MAP @ k	MRR	nDCG @ k
TF-IDF	Without (Baseline)	**0.2573**	**0.1754**	**0.4831**	**0.2394**
	$Q \rightarrow D$	0.1183	0.0727	0.3176	0.1162
	$D \rightarrow Q$	0.1215	0.0747	0.3243	0.1191
Word2Vec (average of the word vectors)	Without	0.0327	0.0192	0.1081	0.0336
	$Q \rightarrow D$	0.0087	0.0052	0.0270	0.0087
	$D \rightarrow Q$	0.0175	0.0105	0.0541	0.0178
Word2Vec (sum of the word vectors)	Without	0.0327	0.0192	0.1081	0.0336
	$Q \rightarrow D$	0.0068	0.0041	0.0203	0.0067
	$D \rightarrow Q$	0.0098	0.0058	0.0304	0.0098
BERT	Without	0.0363	0.0215	0.1182	0.0374
	$Q \rightarrow D$	0.0162	0.0097	0.0473	0.0161
	$D \rightarrow Q$	0.0225	0.0135	0.0676	0.0225

When analysing the results obtained from the Word2Vec use cases, it is possible to state that the highly connected queries are not the top performative ones, independently of the word vector aggregation approach. On the contrary, the results are lower than the overall values. On those metrics, it is possible to see a higher distinction between all experiments.

Regarding the experiment conducted with BERT, the results are similar to the ones perceived on the other use cases. The top connected queries did not retrieve the higher metrics values, since the overall analysis presented higher metric values.

Even though the queries being evaluated are part of the most connected part of the bipartite graph, the introduced graph-based approach did not conduct an excellent job on terms of reducing the lexical gap between the queries and the products information.

5.2.2 CrowdFlower

Similarly to the HomeDepot dataset, the CrowdFlower is also a not highly connected dataset. As shown in section 3.2, on average, each product is connected with 1.10 queries. A disadvantage of this dataset, that was already mentioned in the same section, is the fact that low connectivity can lead to noisy representations resulting from the BiGIT method. Nonetheless, unlike the previous dataset, this one can be used on ranking tasks, since on average each query has 123.75 products as a response.

Analogously to the analysis performed on the Home Depot dataset (Section 5.2.1), this study will be focused on the unknown queries, the test set. An analysis on the train set will not be conducted since it would only show improvements on the queries known to the system. As already stated on Section 4.5, the graph based approach can be initialized on both types of nodes: the documents nodes (D) or the queries nodes (Q). Therefore, all of the analysis conducted on this section will take into consideration both initialisation points.

First, an analysis was conducted on the average number of products that are being retrieved from each experiment. Table 5.5 contains the results obtained. It is important to notice that this first analysis does not take into consideration the relevance of the products being retrieved but only the number of products that are being retrieved on each experiment.

From the analysis of the Table 5.5 it is possible to state that none of the methods was capable of retrieving a high value of products per search query. The obtained values are low, especially when taking into consideration the number of products per query available on the

Table 5.5: Average number of retrieved products per query for the CrowdFlower test set.

Word Representation Model	BiGIT Configuration	Average number of retrieved products
TF-IDF	Without (Baseline)	**1.971**
	$Q \rightarrow D$	1.088
	$D \rightarrow Q$	0.618
Word2Vec (average of the word vectors)	Without	0.118
	$Q \rightarrow D$	0.294
	$D \rightarrow Q$	0.059
Word2Vec (sum of the word vectors)	Without	0.118
	$Q \rightarrow D$	0.412
	$D \rightarrow Q$	0.382
BERT	Without	0.500
	$Q \rightarrow D$	0.294
	$D \rightarrow Q$	0.265

entire dataset. This value is a good indicator that it was not possible to reduce the lexical gap for the new queries since the average number of products being retrieved by each is much lower than the actual value of products being connected to the test set queries. As already stated in Section 5.2.1, several approaches were attempted to try to approximate the lexical representation for those type of queries. However, none of them resulted in a significant improvement when compared with a direct matching based on the word vectors.

An interesting takeaway from this analysis is the differences seen on the results from both Word2Vec methods. If the Word2Vec results are compared between them, the number of products being retrieved by each query is the same. However, if we combine the Word2Vec representations with the proposed approach, in which the lexical space is enriched, it is possible to observe different results. For this specific dataset, when combining the proposed graph-based algorithm with the sum of the word vectors provided by the Word2Vec model, on average, the number of products being retrieved is higher, which indicates that this method was capable of retrieving more products per query.

The experiments conducted with the BERT model were not capable of beating the results retrieved by the TF-IDF experiments. Nonetheless, the word vectors created by BERT were capable of retrieving a higher average number of products when compared with the results obtained when averaging the Word2Vec representations.

Following this analysis, and taking into consideration the results obtained, it was examined how the retrieved products match with the relevance ordered list. Table 5.6 aggregates the results from all experiments conducted. All experiments were also run three times, and the mean and standard deviation of all metrics were summarised on the following table.

Similarly to the Home Depot results, the TF-IDF is the best performative method, either independently and combined with the BiGIT algorithm learning. As it is possible to notice, the TF-IDF representations alone correspond to the best performative hypothesis. However, when combining this method with the BiGIT logic, all of the metrics decrease. Also, when combining the word vectors obtained by the TF-IDF with the designed graph approach, the differences rely on the starting point. For this word representation method, starting the knowledge transference on the query side of the graph brings the lexical space closer.

When taking into account the two approaches designed for the Word2Vec method, it is possible to observe that the sum of the word vectors - to obtain a single representation per query or product title - gives the best results (as already indicated in the values provided by

Table 5.6: Results for the CrowdFlower test set for the several experiments conducted.

Word Representation Model	BiGIT Configuration	Metrics							
		F-Score		MAP @ k		MRR		nDCG @ k	
		μ	σ	μ	σ	μ	σ	μ	σ
TF-IDF	Without (Baseline)	**0.0240**	0	**0.0124**	0	**0.3939**	0	**0.0350**	0
	$Q \rightarrow D$	0.0150	0.0004	0.0077	0.0002	0.2727	0	0.0230	0.0004
	$D \rightarrow Q$	0.0105	0.0005	0.0053	0.0002	0.2727	0	0.0175	0.0003
Word2Vec (average of the word vectors)	Without	0.0014	0	0.0007	0	0.0909	0	0.0036	0
	$Q \rightarrow D$	0.0050	0.0002	0.0027	0	0.0606	0	0.0069	0.0003
	$D \rightarrow Q$	0.0010	0.0002	0.0005	0.0001	0.0606	0	0.0025	0.0003
Word2Vec (sum of the word vectors)	Without	0.0014	0	0.0007	0	0.0909	0	0.0036	0
	$Q \rightarrow D$	0.0055	0.0015	0.0028	0.0001	0.0808	0.0142	0.0075	0.0011
	$D \rightarrow Q$	0.0064	0	0.0034	0	0.0606	0	0.0078	0
BERT	Without	0.0066	0	0.0034	0	0.2121	0	0.0128	0
	$Q \rightarrow D$	0.0034	0.0001	0.0018	0	0.0303	0	0.0045	0.0001
	$D \rightarrow Q$	0.0041	0.0006	0.0021	0.0003	0.1212	0	0.0071	0.0005

Table 5.5). Nonetheless, the metric values are not high enough to be considered relevant results, since this approach is not matching most of the relevant products.

BERT presents slightly better results when compared with the experiments performed on top of the word representations provided by the Word2Vec model. However, TF-IDF it is still outperforming BERT. When considering the experiments conducted with BERT word representations, this method alone outperforms the remaining experiments conducted with the same NLP method combined with the BiGIT algorithm. In other words, this implies that the graph-based approach proposal did not improve the representations for this language model when considering new queries.

Finally, it was analysed the behaviour of the metrics on the most connected nodes of the bipartite graph. For that, it was necessary to understand which products have high connectivity from all of the dataset, so the queries that are connected with them could be observed. With the intent of performing this analysis, the selected queries were the ones that matched with products that are associated with more than ten queries. Table 5.7 contains a summary of the metrics results from those queries.

Table 5.7: Results for the CrowdFlower most highly connected queries from the test set.

Word Representation Model	BiGIT Configuration	Metrics			
		F-Score	MAP @ k	MRR	nDCG @ k
TF-IDF	Without (Baseline)	**0.0201**	**0.0106**	**0.2222**	**0.0246**
	$Q \rightarrow D$	0.0129	0.0068	0.1111	0.0167
	$D \rightarrow Q$	0.0012	0.0006	0.1111	0.0035
Word2Vec (average of the word vectors)	Without	0.0025	0.0012	0.1111	0.0057
	$Q \rightarrow D$	0	0	0	0
	$D \rightarrow Q$	0	0	0	0
Word2Vec (sum of the word vectors)	Without	0	0	0	0
	$Q \rightarrow D$	0	0	0	0
	$D \rightarrow Q$	0	0	0	0
BERT	Without	0.0095	0.0050	0.1111	0.0137
	$Q \rightarrow D$	0.0118	0.0062	0.1111	0.0158
	$D \rightarrow Q$	0.0072	0.0037	0.1111	0.0115

From the analysis of Table 5.7, it is possible to conclude that the Word2Vec methods result in the worst metrics. Most of the experiments performed with this word representation model were not capable of retrieving any products associated with the top connected queries. The only exception is the experiment that uses the average of the word vectors alone, which

was capable of retrieving some products. In other words, this implies that the lexical gap between the queries and the products was even higher for the most connected subjects of the graphs, meaning that for this datasets this is not the right representation learning approach to take into consideration.

TF-IDF kept its performance. The metrics values are similar to the ones retrieved when combining all queries (Table 5.6), except for the experiment in which TF-IDF is combined with the BiGIT approach, initialised on the document side. Considering that the methods were able to keep its performance, this means the fact that those queries are part of the highly connective group of the network is not an essential factor for the algorithm.

Contrary to the other models, BERT was the only method that focused on the top connected queries. Those conclusions are given by the fact that the metric values obtained on these experiments are higher than the value of the metrics obtained on all test dataset (Table 5.6). On this case, the best results obtained are from combining the BERT representation vector with the graph-based propagation approach with a starting point on the query side of the bipartite graph. This experiment metrics output surpassed the remaining ones, which means that for the most connected queries, the lexical gap between the queries and the products is tighter.

5.2.3 Final Remarks

From the analysis conducted on both datasets, it is possible to state that none of the methods was able to retrieve disruptive values. For all of the experiments, none of the metrics evaluated showed promising results.

Since both datasets present significant differences in its distributions and are supposed to be used for different tasks, it is not possible to conduct a direct comparison. For instance, the Home Depot dataset is suitable to be used on a recommendation setting as stated on its documentation. On the other hand, CrowdFlower dataset is marked as appropriate data for a ranking task. Nonetheless, when looking at the tables that contain the overall results (Tables 5.3 and 5.6), it is possible to state that the obtained results are similar. Nevertheless, the CrowdFlower dataset metrics present a worse scenario since it resulted in lower metrics values.

The results obtained were not the expected ones, since none of the values obtained on each evaluation metric was close to the optimal value. Besides that, the proposed graph-based approach did not provide better results when compared with the results obtained by only using the word representations alone. Nonetheless, in an overall analysis, it was still possible to understand how each method works and what are its weaknesses.

Chapter 6

Conclusions

This chapter concludes this document with the analysis and comparison of the initially defined objectives with the work outputs and outcomes. The difficulties identified during this work are also described in this chapter, along with possible future work.

6.1 Achieved Objectives and Contributions

In Sections 1.3 and 1.4 the main objectives and contributions of this work were defined. Regarding the three main objectives defined, it is possible to conclude that all of them were accomplished. Detailed analysis of the click graph method proposed by Jiang et al. (2016) was conducted, which lead to the accomplishment of the first contribution of this work, the implementation of the BiGIT (Bipartite Graph Information Transfer). Besides that, it was also possible to conduct experimental studies with two non-explored datasets by the original paper: Home Depot and CrowdFlower datasets. Lastly, the BiGIT was also combined with several word representations models such as TF-IDF, Word2Vec and BERT.

The implementation of BiGIT leads to the opportunity to deeply understand how the click graph model works, its performance when experimented with new datasets and how it behaves when combined with several word representation learning methods.

Nonetheless, the results showed that this method needs a more connected network to have better performance since it was not able to have excellent results with none of the word representations methods tried. Moreover, the results show that TF-IDF provided the best outcome overall, which is also an indicator that it was not possible to conduct a deeper knowledge transference between the bipartite network.

Finally, it is possible to conclude that even though the results showed that the proposed hypothesis did not lead to good results, it provided clarity to future researchers direction.

The achieved objectives and contributions are summarized in Table 6.1.

Table 6.1: Summary of the achieved objectives and contributions.

Objective	Correspondent contribution	Outcome
Analyse and implement the method proposed by Jiang et al. (2016), the click graph model	Creation of BiGIT	Achieved

Objective	Correspondent contribution	Outcome
Analyse the behaviour of this method when using non-explored datasets	Analysis of Home Depot and CrowdFlower datasets	Achieved
Analyse the behaviour of this type of model when combined with different word representations approaches	Analysis of the results from the experiments performed with TF-IDF, Word2Vec and BERT	Achieved

6.2 Threats to Validity

A priori to the execution of this work, there were concerns regarding potential threats to the results obtained. The most problematic one is the available datasets that were used to evaluate the experiments. Unfortunately, there is not a significant repository of open datasets that can be used on a search engine context. The low connectivity of the datasets (as shown in Chapter 3) was one of the biggest threats to this work. From the analysis conducted, the Home Depot dataset has, on average, more queries per product than the CrowdFlower dataset. Nonetheless, the CrowdFlower presented a higher number of products connected to each query.

Moreover, the results analysed on Section 5.2 revealed that the low connectivity has an impact on the findings. For instance, the metric values obtained from the experiences with the CrowdFlower dataset (Table 5.6) are an order of magnitude lower than the results obtained on the Home Depot dataset experiments (Table 5.3). Besides that, the split between the train and test set, on both datasets, was not performed on a temporal basis as it would happen on a real scenario. On this work, the test set only contained unseen queries, which could lead to a mismatch between data from a real search engine in which it would be possible to see a repetition of some queries - due to the temporal split.

Another significant threat to the validness of this work is the fact that it was necessary to create a new variable, the *click_score*. This new feature, explained in Section 4.2, has the intent of simulating the user clicks. Nonetheless, and as already stated in the same section, it is not possible to simulate user behaviour, since it depends on many factors. However, to conduct this work this type of information is needed. For this reason, the relevance label present on both datasets was combined with a random component to create this new feature. This random element is also considered to be a threat to the results. On real-world search engines, these values would not follow a distribution similar to the ones showed in Figure 4.2. Sometimes the most relevant products are not the ones being clicked, due to several factors.

6.3 Future Work

Even though this work achieved all of the proposed objectives, there are always improvements that can be done. There are several different points on the implementation that can be improved. The first one has the intent of solving the points mentioned in Section 6.2. One improvement point can be using a bigger dataset, with more queries and the entire catalogue of an eCommerce platform. With this type of information, it is possible to have a more densely connected graph with more information to be shared between the queries and

the products. Another aspect that could be considered is using a dataset that contains user information, so it would open the possibility of getting personalised results. Those types of datasets allow experimenting with more complex algorithms such as BiNE, as a network representation learning alternative. Initially, it was planned to do experiments with this method. However, due to the low connectivity of the data, it was not possible to simulate different users. Lastly, the products vectors could also be improved by adding additional information about the product, such as product descriptions.

Another improvement that can be considered is related to text data cleaning. The datasets used during this work contain a very intricate composition of words and combinations of specific details of the products being searched for, which can lead to inaccurate text representations. Therefore, further analysis could be conducted on how these specific types of product names and query representations could be improved without declining the vector representations.

Regarding the BiGIT implementation, two main factors should be considered as future work. First, the BiGIT implementation should consider a stop option instead of having the number of iterations configurable. It should contain a set of ways in which the graph propagation would stop if it reached that stopping point. The second improvement is related to the way the surrounding vector space is being searched when considering a new query. One new strategy can be considering similar queries and then trying to get the response products for all of those queries. These improvements were not considered during the current work due to time restrictions.

Furthermore, an article is planned to be written based on the output of this work and publicly published on a recognised platform or conference so that a different perspective of reviews can be gathered.

References

Artetxe, M., Labaka, G., Lopez-Gazpio, I., & Agirre, E. (2018, October). Uncovering divergent linguistic information in word embeddings with lessons for intrinsic and extrinsic evaluation. In *Proceedings of the 22nd conference on computational natural language learning* (pp. 282–291). Brussels, Belgium: Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/K18-1028 doi: 10.18653/v1/K18-1028

Aslanyan, G., & Porwal, U. (2019). Position bias estimation for unbiased learning-to-rank in ecommerce search. In N. R. Brisaboa & S. J. Puglisi (Eds.), *String processing and information retrieval* (pp. 47–64). Cham: Springer International Publishing.

Azad, H. K., & Deepak, A. (2019). Query expansion techniques for information retrieval: A survey. *Information Processing and Management*, *56*(5), 1698–1735. doi: 10.1016/j.ipm.2019.05.009

Baeza-Yates, R., & Tiberi, A. (2007). Extracting semantic relations from query logs. In *Proceedings of the 13th acm sigkdd international conference on knowledge discovery and data mining* (p. 76–85). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/1281192.1281204 doi: 10.1145/1281192.1281204

Baeza-Yates, R. A., & Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Bag, S., Tiwari, M., & Chan, F. (2019, 01). Predicting the consumer's purchase intention of durable goods: An attribute-level analysis. *Journal of Business Research*, *94*, 408-419. doi: 10.1016/j.jbusres.2017.11.031

Beitzel, S. M., Jensen, E. C., & Frieder, O. (2009). Map. In L. LIU & M. T. ÖZSU (Eds.), *Encyclopedia of database systems* (pp. 1691–1692). Boston, MA: Springer US. Retrieved from https://doi.org/10.1007/978-0-387-39940-9_492 doi: 10.1007/978-0-387-39940-9_492

Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003, mar). A Neural Probabilistic Language Model. *J. Mach. Learn. Res.*, *3*, 1137–1155. Retrieved from http://dl.acm.org/citation.cfm?id=944919.944966

Boytsov, L., & Naidan, B. (2013). Engineering efficient and effective non-metric space library. In N. R. Brisaboa, O. Pedreira, & P. Zezula (Eds.), *Similarity search and applications - 6th international conference, SISAP 2013, A coruña, spain, october 2-4, 2013, proceedings* (Vol. 8199, pp. 280–293). Springer. Retrieved from https://doi.org/10.1007/978-3-642-41062-8_28 doi: 10.1007/978-3-642-41062-8_28

Brenner, E. P., Zhao, J., Kutiyanawala, A., & Yan, Z. (2018). End-to-end neural ranking for eCommerce product search: An application of task models and textual embeddings. In *Ceur workshop proceedings* (Vol. 2319).

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. (2005). Learning to rank using gradient descent. *ICML 2005 - Proceedings of the 22nd*

International Conference on Machine Learning, 89–96. Retrieved from http://portal.acm.org/citation.cfm?doid=1102351.1102363 doi: 10.1145/1102351.1102363

Burges, C. J. C. (2010). From RankNet to LambdaRank to LambdaMART: An Overview. *Learning*, *11*, 23–581. doi: 10.1111/j.1467-8535.2010.01085.x

Cao, Y., Xu, J., Liu, T.-Y., Li, H., Huang, Y., & Hon, H.-W. (2006). Adapting ranking svm to document retrieval. In *Proceedings of the 29th annual international acm sigir conference on research and development in information retrieval* (p. 186–193). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/1148170.1148205 doi: 10.1145/1148170.1148205

Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., & Li, H. (2007, April). *Learning to rank: From pairwise approach to listwise approach* (Tech. Rep. No. MSR-TR-2007-40). Retrieved from https://www.microsoft.com/en-us/research/publication/learning-to-rank-from-pairwise-approach-to-listwise-approach/

Chen, T.-Y. L. W., & Lan, Y. (2009). A unified view of loss functions in learning to rank..

Crammer, K., & Singer, Y. (2002). Pranking with ranking. *Advances in Neural Information Processing Systems*. doi: 10.7551/mitpress/1120.003.0087

Craswell, N. (2009). Mean Reciprocal Rank. In L. LIU & M. T. ÖZSU (Eds.), *Encyclopedia of database systems* (p. 1703). Boston, MA: Springer US. Retrieved from https://doi.org/10.1007/978-0-387-39940-9{_}488 doi: 10.1007/978-0-387-39940-9_488

Craswell, N., & Robertson, S. (2009). Average precision at n. In L. LIU & M. T. ÖZSU (Eds.), *Encyclopedia of database systems* (pp. 193–194). Boston, MA: Springer US. Retrieved from https://doi.org/10.1007/978-0-387-39940-9_487 doi: 10.1007/978-0-387-39940-9_487

Croft, W., Metzler, D., & Strohman, T. (2009). *Search engines: Information retrieval in practice*.

Cui, H., Wen, J.-R., Nie, J.-Y., & Ma, W.-Y. (2002). Probabilistic query expansion using query logs. In *Proceedings of the 11th international conference on world wide web* (p. 325–332). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/511446.511489 doi: 10.1145/511446.511489

Damerau, F. J. (1964, March). A technique for computer detection and correction of spelling errors. *Commun. ACM*, *7*(3), 171–176. Retrieved from https://doi.org/10.1145/363958.363994 doi: 10.1145/363958.363994

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018, jun). 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)* (pp. 4171–4186). Minneapolis, Minnesota: Association for Computational Linguistics. Retrieved from http://arxiv.org/abs/1810.04805 doi: 10.18653/v1/N19-1423

Duan, H., & Hsu, B. J. (2011). Online spelling correction for query completion. *Proceedings of the 20th International Conference on World Wide Web, WWW 2011*, 117–126. doi: 10.1145/1963405.1963425

Figure Eight. (2015, 11). *Crowdflower search results relevance.* Retrieved 2019-10-14, from https://www.kaggle.com/c/crowdflower-search-relevance

Freund, Y., Iyer, R., Schapire, R. E., & Singer, Y. (2003, December). An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, *4*(null), 933–969.

Gao, M., Chen, L., He, X., & Zhou, A. (2018). BiNE: Bipartite network embedding. *41st International ACM SIGIR Conference on Research and Development in Information Retrieval,*

SIGIR 2018, 715–724. doi: 10.1145/3209978.3209987

Goldberg, Y. (2018). *Neural network methods for natural language processing* (Vol. 44) (No. 1). Morgan & Claypool Publishers. doi: 10.1162/COLI_r_00312

Gomma, W. H., & Fahmy, A. A. (2013). A Survey of Text Similarity Approaches. *International Journal of Computer Applications*, *68*(13), 13–18.

Goswami, A., Zhai, C., & Mohapatra, P. (2018). Towards optimization of e-commerce search and discovery. In *ecom@sigir*.

Goutte, C., & Gaussier, E. (2005). A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In *Proceedings of the 27th european conference on advances in information retrieval research* (p. 345–359). Berlin, Heidelberg: Springer-Verlag. Retrieved from https://doi.org/10.1007/978-3-540-31865-1_25 doi: 10.1007/978-3-540-31865-1_25

Grover, A., & Leskovec, J. (2016). Node2vec: Scalable feature learning for networks. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *13-17-August-2016*, 855–864. doi: 10.1145/2939672.2939754

Gupta, M. (2019, January). *NLP | How tokenizing text, sentence, words works.* Retrieved 2020-08-02, from https://www.geeksforgeeks.org/nlp-how-tokenizing-text-sentence-words-works/ (Library Catalog: www.geeksforgeeks.org Section: Python)

Harman, D. (1992). Evaluation issues in information retrieval. *Information Processing and Management*, *28*(4), 439–440. doi: 10.1016/0306-4573(92)90001-G

He, X., Gao, M., Kan, M. Y., & Wang, D. (2017). BiRank: Towards Ranking on Bipartite Graphs. *IEEE Transactions on Knowledge and Data Engineering*, *29*(1), 57–71. doi: 10.1109/TKDE.2016.2611584

He, Y., Tang, J., Ouyang, H., Kang, C., Yin, D., & Chang, Y. (2016). Learning to rewrite queries. In *Proceedings of the 25th acm international on conference on information and knowledge management* (p. 1443–1452). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/2983323.2983835 doi: 10.1145/2983323.2983835

Hendriksen, M., Kuiper, E., Nauts, P., & Schelter, S. (2020). Analyzing and Predicting Purchase Intent in E-commerce : Anonymous vs . Identified Customers. In *Proceedings of the 2020 sigir work- shop on ecommerce (sigir ecom'20).*

Huang, Z., Cautis, B., Cheng, R., Zheng, Y., Mamoulis, N., & Yan, J. (2018). Entity-based query recommendation for long-tail queries. *ACM Transactions on Knowledge Discovery from Data*, *12*(6). doi: 10.1145/3233186

Järvelin, K., & Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, *20*(4), 422–446. doi: 10.1145/582415.582418

Jiang, S., Hu, Y., Kang, C., Daly, T., Yin, D., Chang, Y., & Zhai, C. (2016). Learning query and document relevance from a web-scale click graph. *SIGIR 2016 - Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 185–194. doi: 10.1145/2911451.2911531

Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, *28*, 11–21.

Kozen, D. C. (1992). Depth-First and Breadth-First Search. In *The design and analysis of algorithms* (pp. 19–24). New York, NY: Springer New York. Retrieved from https://doi.org/10.1007/978-1-4612-4400-4{_}4 doi: 10.1007/978-1-4612-4400-4_4

Lan, Y., Zhu, Y., Guo, J., Niu, S., & Cheng, X. (2014). Position-aware ListMLE: A sequential

learning process for ranking. *Uncertainty in Artificial Intelligence - Proceedings of the 30th Conference, UAI 2014*, 449–458.

Langville, A. N., & Meyer, C. D. (2011). *Google's PageRank and beyond: The science of search engine rankings*. Princeton, NJ, USA: Princeton University Press. doi: 10.1007/bf02985759

Le, Q., & Mikolov, T. (2014, 22–24 Jun). Distributed representations of sentences and documents. In E. P. Xing & T. Jebara (Eds.), (Vol. 32, pp. 1188–1196). Bejing, China: PMLR. Retrieved from http://proceedings.mlr.press/v32/le14.html

Le, T. M., & Lauw, H. W. (2014, dec). Probabilistic Latent Document Network Embedding. In *Proceedings - ieee international conference on data mining, icdm* (Vol. 2015-January, pp. 270–279). doi: 10.1109/ICDM.2014.119

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet physics. Doklady*, *10*, 707-710.

Li, H., & Xu, J. (2014, June). Semantic matching in search. *Foundations and Trends in Information Retrieval*, *7*(5), 343–469. Retrieved from https://doi.org/10.1561/1500000035 doi: 10.1561/1500000035

Li, P., Burges, C. J. C., & Wu, Q. (2007). Mcrank: Learning to rank using multiple classification and gradient boosting. In *Proceedings of the 20th international conference on neural information processing systems* (p. 897–904). Red Hook, NY, USA: Curran Associates Inc.

Li, R., Li, L., Wu, X., Zhou, Y., & Wang, W. (2019). Click feedback-aware query recommendation using adversarial examples. *The Web Conference 2019 - Proceedings of the World Wide Web Conference, WWW 2019*, 2978–2984. doi: 10.1145/3308558.3313412

Liao, L., He, X., Zhang, H., & Chua, T. S. (2018, dec). Attributed Social Network Embedding. *IEEE Transactions on Knowledge and Data Engineering*, *30*(12), 2257–2270. doi: 10.1109/TKDE.2018.2819980

Liu, S., Xiao, F., Ou, W., & Si, L. (2017). Cascade ranking for operational e-commerce search. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *Part F129685*, 1557–1565. doi: 10.1145/3097983.3098011

Liu, T.-Y. (2009, March). Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, *3*(3), 225–331. Retrieved from https://doi.org/10.1561/1500000016 doi: 10.1561/1500000016

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., … Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *ArXiv*(1). Retrieved from http://arxiv.org/abs/1907.11692

Liu, Z., Xing, Y., Lian, J., Lian, D., Li, Z., & Xie, X. (2019). A novel user representation paradigm for making personalized candidate retrieval. *ArXiv*, *abs/1907.06323*.

Lo, C., Frankowski, D., & Leskovec, J. (2016). Understanding behaviors that lead to purchasing: A case study of pinterest. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *13-17-August-2016*, 531–540. doi: 10.1145/2939672.2939729

Mandal, A., Khan, I., & Kumar, P. S. (2019). Query rewriting using automatic synonym extraction for e-commerce search. *CEUR Workshop Proceedings*, *2410*.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. New York, NY, USA: Cambridge University Press.

Martins, B., & Silva, M. J. (2004). Spelling correction for search engine queries. In J. L. Vicedo, P. Martínez-Barco, R. Muńoz, & M. Saiz Noeda (Eds.), *Advances in natural language processing* (pp. 372–383). Berlin, Heidelberg: Springer Berlin Heidelberg.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. In Y. Bengio & Y. LeCun (Eds.), *1st international conference on learning representations, {iclr} 2013, scottsdale, arizona, usa, may 2-4, 2013, workshop track proceedings*. Retrieved from http://arxiv.org/abs/1301.3781

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th international conference on neural information processing systems - volume 2* (p. 3111–3119). Red Hook, NY, USA: Curran Associates Inc.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013a). Distributed Representations of Words and Phrases and their Compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems 26* (pp. 3111–3119). Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

Mikolov, T., Yih, W. T., & Zweig, G. (2013, jun). Linguistic regularities in continuous spaceword representations. In *Naacl hlt 2013 - 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies, proceedings of the main conference* (pp. 746–751). Atlanta, Georgia: Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/N13-1090

Mitkov, R. (2003). *The oxford handbook of computational linguistics (oxford handbooks in linguistics s.)*. USA: Oxford University Press, Inc.

Müller, C., & Gurevych, I. (2009). A study on the semantic relatedness of query and document terms in information retrieval. *EMNLP 2009 - Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: A Meeting of SIGDAT, a Special Interest Group of ACL, Held in Conjunction with ACL-IJCNLP 2009*(August), 1338–1347. doi: 10.3115/1699648.1699680

Pang, L., Lan, Y., Guo, J., Xu, J., Xu, J., & Cheng, X. (2017). DeepRank: A new deep architecture for relevance ranking in information retrieval. *International Conference on Information and Knowledge Management, Proceedings*, *Part F131841*, 257–266. doi: 10.1145/3132847.3132914

Pennington, J., Socher, R., & Manning, C. D. (2014, oct). GloVe: Global vectors for word representation. In *Emnlp 2014 - 2014 conference on empirical methods in natural language processing, proceedings of the conference* (pp. 1532–1543). Doha, Qatar: Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/D14-1162 doi: 10.3115/v1/d14-1162

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). DeepWalk: Online Learning of Social Representations. In *Proceedings of the 20th acm sigkdd international conference on knowledge discovery and data mining* (pp. 701–710). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/2623330.2623732 doi: 10.1145/2623330.2623732

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018, jun). Deep Contextualized Word Representations. In *Proceedings of the 2018 conference of the north american chapter of the association for computational linguistics: Human language technologies, volume 1 (long papers)* (pp. 2227–2237). New Orleans, Louisiana: Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/N18-1202 doi: 10.18653/v1/n18-1202

Pirnay-Dummer, P., Ifenthaler, D., & Seel, N. M. (2012). Semantic Networks. In N. M. Seel

(Ed.), *Encyclopedia of the sciences of learning* (pp. 3025–3029). Boston, MA: Springer US. Retrieved from https://doi.org/10.1007/978-1-4419-1428-6{_}1933 doi: 10.1007/978-1-4419-1428-6_1933

Rao, V. R. (2018, April). *Here's why you should use Python for scientific research.* Retrieved 2020-08-26, from https://developer.ibm.com/technologies/python/blogs/use-python-for-scientific-research/

Riseman, E. M., & Hanson, A. R. (1974). A contextual postprocessing system for error correction using binary n-grams. *IEEE Transactions on Computers*, *C-23*(5), 480-493.

Robertson, S., & Zaragoza, H. (2009, apr). The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, *3*(4), 333–389. Retrieved from http://dx.doi.org/10.1561/1500000019 doi: 10.1561/1500000019

Robinson, D. (2017, September). *Why is Python Growing So Quickly?* Retrieved 2020-08-26, from https://stackoverflow.blog/2017/09/14/python-growing-quickly/

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1988). Neurocomputing: Foundations of research. In J. A. Anderson & E. Rosenfeld (Eds.), (pp. 696–699). Cambridge, MA, USA: MIT Press. Retrieved from http://dl.acm.org/citation.cfm?id=65669.104451

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv*(NeurIPS), 1-5. Retrieved from http://arxiv.org/abs/1910.01108

Sasaki, Y. (2007). *The truth of the F-measure.* Old Dominion University, Department of Computer Science. Retrieved from http://www.cs.odu.edu/~mukka/cs795sum09dm/Lecturenotes/Day3/F-measure-YS-26Oct07.pdf

Sondhi, P., Sharma, M., Kolari, P., & Zhai, C. (2018). A taxonomy of queries for e-commerce search. *41st International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2018*, 1245–1248. doi: 10.1145/3209978.3210152

Sorokina, D., & Cantú-Paz, E. (2016). Amazon search: The joy of ranking products. *SIGIR 2016 - Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 459–460. doi: 10.1145/2911451.2926725

Srinath, K. R. (2017). Python – The Fastest Growing Programming Language. *International Research Journal of Engineering and Technology*, 354–357. Retrieved from https://www.irjet.net/archives/V4/i12/IRJET-V4I1266.pdf

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015). LINE: Large-scale information network embedding. *WWW 2015 - Proceedings of the 24th International Conference on World Wide Web*(March), 1067–1077. doi: 10.1145/2736277.2741093

Taylor, M., Guiver, J., Robertson, S., & Minka, T. (2008, February). Softrank: Optimising non-smooth rank metrics. In *Wsdm 2008* (WSDM 2008 ed.). Retrieved from https://www.microsoft.com/en-us/research/publication/softrank-optimising-non-smooth-rank-metrics/

The Home Depot. (2016, 01). *Home Depot Product Search Relevance.* Retrieved 2019-10-14, from https://kaggle.com/c/home-depot-product-search-relevance

Tsai, M.-F., Liu, T.-Y., Qin, T., Chen, H.-H., & Ma, W.-Y. (2007). Frank: A ranking method with fidelity loss. In *Proceedings of the 30th annual international acm sigir conference on research and development in information retrieval* (p. 383–390). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/1277741.1277808 doi: 10.1145/1277741.1277808

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I.

(2017). Attention is all you Need. In I. Guyon et al. (Eds.), *Advances in neural information processing systems 30* (pp. 5998–6008). Curran Associates, Inc. Retrieved 2020-08-26, from http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

Vechtomova, O. (2009). Query expansion for information retrieval. In L. LIU & M. T. ÖZSU (Eds.), *Encyclopedia of database systems* (pp. 2254–2257). Boston, MA: Springer US. Retrieved from https://doi.org/10.1007/978-0-387-39940-9_947 doi: 10.1007/978-0-387-39940-9_947

Wang, D., Cui, P., & Zhu, W. (2016). Structural deep network embedding. In *Proceedings of the acm sigkdd international conference on knowledge discovery and data mining* (Vol. 13-17-August-2016, pp. 1225–1234). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/2939672.2939753 doi: 10.1145/2939672.2939753

Wang, J., & Li, S. (2012). Query-driven iterated neighborhood graph search for large scale indexing. In *Acm multimedia 2012* (pp. 179–188).

Wang, X., Golbandi, N., Bendersky, M., Metzler, D., & Najork, M. (2018). Position bias estimation for unbiased learning to rank in personal search. In *Proceedings of the eleventh acm international conference on web search and data mining* (p. 610–618). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3159652.3159732 doi: 10.1145/3159652.3159732

Webster, J. J., & Kit, C. (1992). Tokenization as the initial phase in nlp. In *Proceedings of the 14th conference on computational linguistics - volume 4* (p. 1106–1110). USA: Association for Computational Linguistics. Retrieved from https://doi.org/10.3115/992424.992434 doi: 10.3115/992424.992434

Wu, W., Li, H., & Xu, J. (2013). Learning query and document similarities from click-through bipartite graph with metadata. In *Wsdm 2013 - proceedings of the 6th acm international conference on web search and data mining* (pp. 687–696). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/2433396.2433481 doi: 10.1145/2433396.2433481

Xu, J., & Li, H. (2007). Adarank: A boosting algorithm for information retrieval. In *Proceedings of the 30th annual international acm sigir conference on research and development in information retrieval* (p. 391–398). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/1277741.1277809 doi: 10.1145/1277741.1277809

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32* (pp. 5753–5763). Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/8812-xlnet-generalized-autoregressive-pretraining-for-language-understanding.pdf

Yin, D., Hu, Y., Tang, J., Daly, T., Zhou, M., Ouyang, H., … Chang, Y. (2016). Ranking relevance in yahoo search. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *13-17-Augu*, 323–332. Retrieved from http://www.kdd.org/kdd2016/papers/files/adf0361-yinA.pdf doi: 10.1145/2939672.2939677

Yin, W., & Schütze, H. (2016, August). Learning word meta-embeddings. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 1351–1360). Berlin, Germany: Association for Computational Linguistics. Retrieved

from https://www.aclweb.org/anthology/P16-1128 doi: 10.18653/v1/P16-1128

Zhang, H., Wang, S., Zhang, K., Tang, Z., Jiang, Y., Xiao, Y., … Yang, W.-Y. (2020). Towards personalized and semantic retrieval: An end-to-end solution for e-commerce search via embedding learning. In *Proceedings of the 43rd international acm sigir conference on research and development in information retrieval* (p. 2407–2416). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3397271.3401446 doi: 10.1145/3397271.3401446

Zhang, Y., Wang, D., & Zhang, Y. (2019). Neural IR meets graph embedding: A ranking model for product search. In *The web conference 2019 - proceedings of the world wide web conference, www 2019* (pp. 2390–2400). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/3308558.3313468 doi: 10.1145/3308558.3313468