Exploiting Embedding in Content-Based Recommender Systems

Yanbo Huang
Exploiting Embedding in Content-Based Recommender Systems

MASTER OF SCIENCE THESIS

For the degree of Master of Science in Computer Science at Delft University of Technology

Yanbo Huang

November 29, 2016

Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS) · Delft University of Technology
The work in this thesis was supported by XING. Their cooperation is hereby gratefully acknowledged.
The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS) for acceptance a thesis entitled

**Exploiting Embedding in Content-Based Recommender Systems**

by

**Yanbo Huang**

in partial fulfillment of the requirements for the degree of

**Master of Science Computer Science**

Dated: November 29, 2016

Committee Member(s):

______________________________
Dr. Martha Larson

______________________________
Prof. Alan Hanjalic

______________________________
Dr. Claudia Hauff

______________________________
Dr. Fabian Abel
(Advisory member)

______________________________
Drs. Jeroen Vuurens
Abstract

XING is a leading career-oriented social networking site in Europe, which usually recommend job ads to their customers. One of the widely used methods in Recommender Systems is content-based filtering, which analyzes the description of item characteristics and the user profile illustrating user’s preferences. Due to the sparsity of its dataset, i.e. many job postings are rarely interacted with, XING has been using content-based recommender system to promote the quality of the recommendations. Recent word embedding technique learns semantically meaningful representations for words from co-occurrence in sentences, which enables the effective comparison between words. Based on the Word2Vec technique, XING represents job postings by the average embedding over words they contain. This study explores three alternative methods to represent job postings for the task of recommending jobs to users.

In the first experiment, we explore whether the use of a subset of words is more effective to represent the job postings. In the second experiment, instead of averaging over word embeddings, we directly learn document embeddings using Paragraph2Vec. And finally, the third experiment uses Word Mover’s Distance to estimate the similarity between job postings. Our experiments show that the embeddings that are learned with Paragraph2Vec result in a better estimation of which job postings are similar, but only when high-dimensional settings are used. The Word Mover’s Distance algorithm is computationally expensive, therefore we use existing lower-bounds that allowed us to complete a small-scale experiment within the available time. The results indicate that Word Mover’s Distance is not as effective as the average over word embeddings and Paragraph2Vec.

In the final part of this thesis, we present the Link2Vec, a novel item representation method based on Word2Vec, which learns semantic representations for items based on the context surrounding the hyperlinks that refer to the item, e.g. hyperlinks to the item’s Wikipedia page. Our experiments show that the effectiveness of the embeddings learned with Link2Vec improves with the amount of training data. For the evaluation on the MovieLens dataset, we only obtained a limited set of hyperlinks, which resulted in results that approximate a baseline that uses the average over word embeddings.
# Table of Contents

Acknowledgements ix

1 Introduction 1
   1-1 Recommender System in XING .......................... 1
   1-2 Contribution ........................................... 2
   1-3 Structure of the Thesis ................................ 3

2 Background 5
   2-1 Word2vec ............................................... 5
   2-2 Word Averaging Method .................................. 7
   2-3 Paragraph2Vec .......................................... 7
   2-4 Word Mover’s Distance .................................. 8
   2-5 XING’s Baseline ......................................... 9

3 Approaches 13
   3-1 Small Size Word Aggregating ........................... 13
   3-2 Paragraph Vector ....................................... 14
   3-3 Word Mover’s Distance .................................. 14
      3-3-1 Document Centroid Distance ......................... 14
      3-3-2 Relaxed Word Mover Distance ....................... 15

4 Experiments and Evaluations 19
   4-1 Dataset Description and Setup ......................... 19
   4-2 Evaluation Method .................................... 19
   4-3 Word Vector Aggregation ............................... 21
   4-4 Word Mover’s Distance ................................ 22
   4-5 Paragraph Vector ...................................... 23

Master of Science Thesis Yanbo Huang
# Table of Contents

5 Link2Vec  
5-1 Introduction ................................................. 27  
5-2 Experiment Setup and Evaluation  
  5-2-1 Dataset Description and Setup ........................... 28  
  5-2-2 Evaluation ............................................. 30  
5-3 Results ....................................................... 31  
5-4 Discussion of Link2Vec ....................................... 32  

6 Conclusions ..................................................... 33  

Bibliography ....................................................... 35
## List of Figures

2-1 CBOW model: each input \( w(t) \) is a vector which is a column in the matrix \( W \). The hidden layer is the summation or average of all input vectors. ................................................. 6

2-2 Skip-gram model: each input \( w(t) \) is a vector which is a column in the matrix \( W \). .......................................................................................................................... 7

2-3 Distributed Memory Model of Paragraph2vec (PV-DM). .................................................................................................................. 8

2-4 The distance between two documents is the minimum cumulative distance that all words in the document 1 need to travel to exactly match document 2 [5]. ........................................... 9

3-1 T-SNE visualization of posting ids in the semantic vector space. The label for each point is the posting id for each posting. ........................................................................................................ 16

3-2 T-SNE visualizations of words in the semantic embedding. .............................................................................................................. 17

3-3 Histogram of pairwise Euclidean distance with different words selection methods for vector averaging: Five words with highest TF-IDF scores words, five randomly selected words, and all words. The x-axis is the Cosine distance between document vectors while the y-axis represents the amount of document pairs with that distance. ........................................ 18

4-1 The definition of similar postings: Two postings \( p_1 \) and \( p_2 \) are considered to be similar if they were co-interacted by at least 2 users. ............................................................................................................ 21

4-2 The TF-IDF scores for words in a posting in a descending order. ........................................................................................................ 22

4-3 F1@20 scores for three aggregation methods with different word vector dimensions. ........................................................................ 24

4-4 F1@20 scores for using XING Posting Representation method and Paragraph2Vec to learn document embeddings with different vector sizes. ............................................................................. 25

5-1 Three paragraphs of the contexts for the hyperlink. The words with color are the anchor texts for this hyperlink. The hyperlink in the page is link to the Wikipedia page for the movie "The Matrix". .................................................................................. 28

5-2 One of the result pages in Wikireverse for retrieveing https://en.wikipedia.org/wiki/The_Matrix. These links quote this Wikipedia link in their web pages. ......................................................... 29
List of Tables

4-1 User Interactions in XING’s Dataset .................................................. 19
4-2 Snippet of XING’s User-Posting Interaction log ................................. 20
4-3 Job Postings in XING’s Dataset .......................................................... 20
4-4 Scores of different evaluation metrics and different aggregation methods when learning word embedding on 500 dimensions .......................... 23
4-5 Scores of different evaluation metrics of the Word Mover Distance when learning word embedding on 100 dimensions ............................... 25
4-6 Evaluation Metrics for Paragraph2Vec when training word embedding on 500 dimension ................................................................. 26
5-1 Scores of different evaluation metrics and different document embedding techniques when learning word embedding on 300 dimensions ....................... 31
5-2 Scores of different evaluation metrics and different document embedding techniques when learning word embedding on 500 dimensions ....................... 31
Acknowledgements

I would like to express my gratitude to Dr. Martha Larson for her valuable guidance and extraordinary support in this thesis process. Furthermore, I would like to thank Drs. Jeroen Vuurens for the useful comments, remarks and engagement through the learning process of this master thesis. Also thanks to Dr. Fabian Abel, who acted as an advisory member of the thesis committee, and Dr. Daniel Kohlsdorf from XING. I would like to thank them for sharing XING’s job postings and user-interaction dataset and their useful advices during this master thesis. In addition, thanks to SURFsara for supplying the high performance computing platform.

Their help has been essential to the completion of this work.

Delft, University of Technology

November 29, 2016

Yanbo Huang
“The best programs are written so that computing machines can perform them quickly and so that human beings can understand them clearly. A programmer is ideally an essayist who works with traditional aesthetic and literary forms as well as mathematical concepts, to communicate the way that an algorithm works and to convince a reader that the results will be correct.”

— Donald Ervin Knuth
Recommender Systems are utilized in a variety of areas: applications in the fields of music, movie, job recommendation and etc. Rather than supplying static user interfaces, recommender systems improve users’ experience by personalizing what they see, often leading to greater engagement and loyalty. In return, businesses receive more explicit information regarding user preferences which helps to improve the recommendations iteratively and create a virtuous circle.

In the development of recommender systems, many applications take either of two typical approaches: collaborative filtering or content-based filtering. Collaborative filtering is a class of methods that recommend items to users based on the preferences others have expressed for those items [16]. In some e-commerce websites, like Amazon.com, a well-known example for this approach is “People who bought A and B also bought C”. By taking other user behaviors into account, collaborative filtering generates recommendations using group knowledge based on users with similar traits. The main advantage of collaborative filtering is, that there is no need to provide detailed and up-to-date item descriptions to the system. However, items are rarely recommended when they lack sufficient recorded user preferences [1]. This phenomenon is referred to as the new-item cold start problem. Content-based filtering, alternatively, analyzes the description of item characteristics and the user profile illustrating user’s preferences. The recommendation task of this approach then consists of determining items that match the user’s preference best, which typically performs better than collaborative filtering for items that are rarely interacted with.

1-1 Recommender System in XING

XING\(^1\) is a leading career-oriented social networking site in Europe, which serves around 15.2 million members, mostly in German speaking countries, i.e. Germany, Austria and Switzerland. Its core business largely depends on subscriptions and revenue from members

\(^1\text{https://www.xing.com/}\)

Master of Science Thesis Yanbo Huang
that apply to posted jobs, which is stimulated by recommending posted jobs to members. In
the view of the business model, XING provides basic job searching and job recommendation
services for free. Based on the free member society, XING also provides additional services to
its premium members offering them easy access to other members and efficient job searching.
Offering high quality of job recommendation attracts more potential customers to use its
services, eventually increasing XING’s revenue. Since XING’s dataset is sparse, i.e. many
job postings are rarely interacted with, XING has been using a content-based instead of a
collaborative-filtering recommender system to promote the quality of the recommendations.
The reason for a specific recommendation is either of “you viewed similar jobs ads” or it
“matches your profile and activity”. In XING’s framework, they first use ElasticSearch to
retrieve a short list of job ads based on a user profile, and then re-rank the jobs in this list
according to their similarity to a randomly selected job posting bookmarked by the user in
the past. They assume that people will be interested in the job ads being similar with the ads
they previously liked. In XING’s system, if the users bookmark some job postings, it shows
they have high preference for them. To simplify the calculations, they just randomly select
one bookmarked posting. XING’s final recommendations to its users are a combination of
several components, i.e. career path features, community features and the re-ranked list.

The objective of this research is to improve XING’s re-ranking algorithm. At the core of
XING’s re-ranking algorithms, jobs are represented in an embedding space, that is used to
estimate the similarity between jobs offered. To estimate the similarity between jobs, they
learn distributed vectors for words using Word2Vec [8], and use the average word vector of
the words that appear in a job posting to represent it. The Cosine Similarity between two
job posting vectors is then used to estimate their similarity.

1-2 Contribution

In this work, we experiment with two ways to refine the job representations by selecting a sub-
set of words to represent a job posting. Additionally, we experiment with Paragraph2Vec (Le
& Milolov, 2014) to directly learn distributed document vectors. Alternatively, the similarity
between the text of two job postings is estimated by the Word Mover’s Distance (WMD). For
the evaluation of these methods, we design an offline testing method.

In the second part of this thesis, in Chapter 5, we propose a novel method called Link2Vec,
which learns item representations based on the textual context of hyperlinks that refer to
targeted items. When people quote a specific hyperlink, such as the Amazon url for a book,
a Wikipedia page link for a singer, or a IMDB url for a movie, this hyperlink represents
the item to which it refers. After discussion with the data scientists in XING, we know
that their job posting urls are rarely quoted by other websites. Therefore, we performed
this experiment on the Movielens dataset, using hyperlinks towards the movies’ Wikipedia
pages as references to these movies. To learn a representation for a movie, we replace the
hyperlinks in referring documents by a token that represents the movie, and learn their
vector representations as normal words in the Word2Vec framework. In the evaluation, we
compare the Link2Vec approach to a movie representation that corresponds to the average
over Word2Vec embeddings of the words that appear on the movies’ Wikipedia page, as
proposed by Musto et al. [10].
1-3 Structure of the Thesis

The rest of this thesis is organized as follows: Chapter 2 explains in detail of several concepts mentioned in this part: Word2Vec, Word Vector Aggregation, Paragraph2Vec, and Word Mover’s Distance. Chapter 3 describes XING’s baseline and the methods we experiment with for learning job posting embeddings and estimate pairwise distances. In Chapter 4, we present the experiments results and discuss how to evaluate XING’s baseline and other methods. In the second part, Chapter 5, we describe the Link2Vec and measure its effectiveness on the Movielens dataset. Chapter 6 concludes the whole thesis work and present open issues for future work.
In the area of statistical language modeling, Bengio et al. [2] proposed a neural probabilistic language model by learning a distributed representation for words, which showed to significantly outperform N-gram models. Mikolov et al. introduced a technique called Word2Vec [8] that can be used to efficiently learn high-quality semantic vectors for words based on large text corpus. Since words that are similar tend to appear in similar contexts, semantic vectors for similar words are located closely in the vector space. Moreover, because the word vectors capture many linguistic regularities, and because of the compositionality of the space, with the remarkable linear relationships, we can conduct the vector arithmetic operations, for example, vec("king") - vec("man") + vec("woman") is close to vec("queen") and vec("Beijing") - vec("China") + vec("Japan") is near to vec("Tokyo").

In order to learn these word embeddings, Mikolov et al. proposed two variants: Continuous Bag of Words (CBOW) which uses the context to predict a target word, and Skip-gram which predicts the context of each word.

In Word2Vec framework, every word $w$ in the word dictionary $V$ is mapped to a vector $w(t)$, which is a column in the matrix $W$ (matrix $W$ is randomly initialized). The CBOW model predicts a word $w(t)$ using its context $w(t-n),...,w(t-1),w(t+1),...,w(t+n)$, while the Skip-gram model predicts each word in the context using the word $w(t)$.

In the CBOW variant, the hidden layer is the summation or average of the input word vectors. Given a sentence $w_1, w_2, ..., w_T$, the objective function is to maximise the log likelihood of the language model:

$$\frac{1}{T} \sum_{t=k}^{T-k} \log P(w_t|w_{t-k}, ..., w_{t-1}, w_{t+1}, ..., w_{t+k})$$  \hspace{1cm} (2-1)$$

The softmax function is used to calculate the posterior distribution of the target word given a specific context:
Figure 2-1: CBOW model: each input $w(t)$ is a vector which is a column in the matrix $W$. The hidden layer is the summation or average of all input vectors.

$$P(w_t|w_{t-k}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+k}) = \frac{e^{y_t}}{\sum_i e^{y_i}}$$ \hspace{1cm} (2-2)$$

where $y_i$ is the output of the i-th unit in the output layer.

For the Skip-gram model, the input layer is a so called “1-hot vector”, which copies the embedding for a word in the word vector matrix to the hidden layer. The objective function predicts the words that appear in the targeted word’s context:

$$\frac{1}{T} \sum_{t=k}^{T-k} \sum_{j=-k, j \neq 0}^{k} \log P(w_{t+j}|w_t)$$ \hspace{1cm} (2-3)$$

Where $T$ is the length of the whole sentence. For both variants, back propagation of the neural network and Stochastic Gradient Descent (SGD) on the objective function are performed by iteratively updating the parameter $\Theta = (W, U)$:

$$\Theta \leftarrow \Theta - \eta \frac{\partial \log P}{\partial \Theta}$$ \hspace{1cm} (2-4)$$

Where $\eta$ is the learning rate. The parameter $\Theta$ consists of the matrix $W$ containing word vectors and the weights $U$ in the neural network.

The objective of the Word2Vec framework is to predict the context of words or words based on their context. The word embeddings are learned through maximizing the objective function.

Although Word2Vec can learn semantic vectors for single words, and is also effectively used for short phrase segments [9], it is generally not employed for long text representation.
2-2 Word Averaging Method

Figure 2-2: Skip-gram model: each input \( w(t) \) is a vector which is a column in the matrix \( W \).

2-2 Word Averaging Method

For the task of recommending movies and books, Musto et al. proposed to represent movies and books by averaging the embeddings of all words that appear on their Wikipedia page [10]. The embeddings for the words are learned by using Word2Vec on the English Wikipedia dataset.

The experimental evaluations on MovieLens and DBbook show that their word averaging method can achieve comparable effectiveness in F-measure to state-of-the-art collaborative filtering baselines, such as the User-to-User [16] and Item-to-Item k-nearest neighbors, as well as the Bayesian Personalized Ranking Matrix Factorization [13].

2-3 Paragraph2Vec

In 2014, Le & Mikolov proposed a method that learns fixed length feature representations for various length texts [6] called Paragraph2Vec, which is inspired by the original Word2Vec framework. Figure 2-3 shows the Distributed Memory Model of Paragraph Vectors (PV-DM), which is an extension of the CBOW model in Word2Vec. In the architecture of Paragraph2Vec, the input contains a vector that represents the paragraph. This paragraph vector represents the missing information of a paragraph and acts as the paragraph’s topic.

Word vectors in the matrix \( W \) are shared over the whole training process, while the paragraph id vector, which is a column in the matrix \( D \), is only shared within one paragraph. SGD is used to optimize the parameter \( \Theta \) in a iterative way. \( \Theta \) consists of the word vector matrix \( W \), the paragraph vector matrix \( D \) and the neural networks weights \( U \). In addition, for unseen paragraphs, their paragraph vectors can be inferred by adding more columns in \( D \).
and applying Stochastic Gradient Descending on $D$ while holding other parameters fixed [6]. This is the so called “inference stage”.

### 2-4 Word Mover’s Distance

The Word Mover’s Distance (WMD) is claimed to achieve better results than other baselines (bag-of-words, TF-IDF, BM25 Okapi) when used for estimating document similarity [5]. It is inspired by the Earth Mover Distance, which can be used to find the most efficient solution to a transportation problem. To illustrate the Earth Mover Distance, suppose that there are $n$ good suppliers, with a limited amount of goods, which are needed to supply $m$ consumers with limited capacity. The cost for the unit good for a supplier and consumer pair is given. The so called transportation problem is to figure out a “least-expensive” flow of goods from the suppliers to the consumers that satisfies the consumers’ demands. The Earth Mover’s Distance addresses the transportation problem by measuring the distance between two distributions in some regions, where the pairwise distance between points is the ground distance. It was implemented in some color and texture images [14, 15], where this distance can be applied to the distributions of points, such as colors or texture features [7]. In 2015, Kusner et al. proposed a novel distance function between text documents called Word Mover’s Distance (WMD) [5]. For the text based corpus, if a document can be viewed as a distribution of its words, the Earth Mover’s Distance can be then used to estimate the distance between two documents. The Word Mover Distance is viewed as a instance of the Earth Mover Distance. Figure 2-4 illustrates the concepts of the Word Mover’s Distance, where the semantic space is learned by the Word2Vec model.

WMD assumes that the dissimilarity between two words is a natural building block to create a distance between two documents. It uses the embeddings that are learned with Word2Vec to estimate the distance between pairs of words. Words dissimilarities (or word travel cost)
The distance between two documents is the minimum cumulative distance that all words in the document 1 need to travel to exactly match document 2 [5].

$$c(w_i, w_j) = \| V(w_i) - V(w_j) \|_2$$  \hspace{1cm} (2-5)

where, $w_i$ and $w_j$ are two words, while $V(w_i)$ and $V(w_j)$ are their learned word embeddings.

The distance (travel cost) between two documents is defined as the minimum (weighted) cumulative cost required to move all words from document $d$ to document $d'$:

$$C_{wmd}(d, d') = \min_{F \geq 0} \sum_{w_i \in d} \sum_{w_j \in d'} F_{w_i, w_j} c(w_i, w_j)$$

subject to: $\sum_{w_j} F_{w_i, w_j} = d_i$ and $\sum_{w_i} F_{w_i, w_j} = d'_j$  \hspace{1cm} (2-6)

where $F$ is a flow matrix indicating how much of $w_i$ in document $d$ travels to $w_j$ in document $d'$. $d_i$ stands for a document representation by normalized bag-of-words (nBOW), for example, word $i$ appear $c_i$ times in the document, it denotes

$$d_i = \frac{c_i}{\sum_{j=1}^n c_j}$$

2-5  XING’s Baseline

For a better illustration, here we list a example of XING’s job postings:

2459561 Research Scientist / Engineer (f/m) for Human-Machine Collaboration A strong position in research development is a prerequisite for ABB’s business success. Essential contributions grow out of the collaboration between ABB’s research and operational organizations. Our Corporate Research Center
in Ladenburg, close to Heidelberg in Germany, is one of seven global ABB Corporate Research facilities and focuses on Factory Automation, Service Solutions, Automation System Architectures & Engineering as well as Automation for Power. Location: Ladenburg, Baden Wurttemberg, Germany Job Function: Research and Development Job Reference Code: DE53895116 Tasks: We are continuously looking for highly qualified scientists and engineers in the field of human-machine collaboration. You will work on the fascinating boundary of research, development, and application within an interdisciplinary team of highly motivated scientists. Your work covers everything from idea creation to problem definition and from concept development to implementation. During the pilot testing of new products and solutions you will be in touch with customers and their factories. Your technical and cross-cultural communicational skills will be required to establish the link between internal research, external research, and ABB’s international business units. You will develop innovative

The ABB Group of companies operates in around 100 countries and employs about 150,000 people. ABB AG Recruiting Centre Julia Fink Phone +49 (0) 621 381 3500 Corporate Research Centre Dr. Thomas Reisinger Phone + 49 (0) 6203 71 6237 www.facebook.com/ABBcareersCentralEurope Our ambition. Your legacy. Research Scientist / Engineer f/m for Human-Machine Collaboration

Research Science, Electrical Engineer, Mechanical Engineer, Mechatronic, Forschung, Ingenieurwesen, Elektroingenieur, Elektroingenieurin, Maschinenbauingenieur, MaschinenbauingenieurinResearch Science, Electrical Engineer, Mechanical Engineer, maschinenbauingenieurin Degree - Electrical Engineering, Mechanical Engineering, Mechatronics or equivalent work experience en Design Url Ladenburg DE Baden-Wurttemberg 41200 1010 289317 1 NULL 1 2013-11-20 09:30:13 2013-11-20 09:30:14 2014-01-19 09:30:14 0 1

We parsed approximately 3000,000 job postings in XING’s dataset and for each of them, we used tab to separate different categories, which include Posting Id, Job Title, Description, Tags, Skills and so on.

XING uses ElasticSearch to retrieve a short list of job ads based on a user profile, and then re-ranks them according to their similarity to a randomly selected job posting that was bookmarked by the user in the past. Currently, they take the average of almost all word embeddings for words in a posting as the vector representation of that job posting.

XING initially uses word embeddings that are learned jointly from the English Wikipedia dataset and German Wikipedia dataset because there are English and German job postings in their websites. However, the learned embeddings lack of meaning for words that are specific to the jobs domain. For example, when retrieving the word ‘xing’ for the most similar words, the learned model returns some Chinese surnames (zhao, huang, li, ...). To outcome this shortcoming, they add a collection of the latest job postings in the learning corpus. The same searching for the word ‘xing’ results in the words like ‘facebook’, ‘linkedin’ and ‘twitter’, which is more reasonable.

To represent a job posting, they then average all word embeddings for the words that appear in a job posting, excepting English and German stop words. The pairwise distance between job
postings is estimated by the Cosine Distance between their vectors. In the remainder of this report, we refer to this word vector averaging method as **XING Posting Representation Method**.
3-1 Small Size Word Aggregating

Figure 3-2 illustrates how the words are distributed in the semantic embedding space using a t-SNE projection. For this illustration, 937 were randomly selected from the learned vocabulary. It is clear to see that similar words located closely in the vector space. In this figure, the blue rectangle shows words for cities and countries, while the red rectangle demonstrates the words related to computer science. Since similar words are likely to be located closely, if we represent each posting by the average of word embedding in it, we probably ends up with some central locations, which makes the postings less distinguishable. Therefore, instead of considering all words inside the document, we suspect that some words in a posting may be better to describe a job than others. To verify our suspicion, we represented job postings by the average of n-highest TF-IDF terms, n-randomly chosen terms, and observed that the average pairwise distance is higher when choosing high TF-IDF words (Figure 3-3).

For this experiment, we randomly sampled 5000 job postings from the Xing’s dataset, and the word embeddings were pre-trained on the corpus of the latest English Wikipedia, the latest German Wikipedia and a collection of XING’s job postings. We learned document vectors by the average of selected word vectors, and then we calculated pairwise Cosine Distances of document vectors. The mean pairwise distance of averaging all word vectors was 0.398. Alternatively, we averaged the word embeddings of the five words with the highest TF-IDF scores in a document. This resulted in a mean distance between documents of 0.93. For another variant, document vectors were generated by the average of the 5 randomly selected words. We repeated this experiment 100 times, and the mean distance was around 0.792.

In Figure 3-3, we can observe that the average of semantic vectors over all words in a document makes them more equi-distant, and therefore the distance between documents may be less meaningful. Instead of considering all words inside the document, we suspect that some words in a posting may better to describe the meaning of the job than others. We hypothesize that representing documents by a subset of the most descriptive words, will make the representations more distinguishable and improve the estimation of the similarity between
documents. Based on this thinking, we tested this hypothesis by designing two ways to refine the initial word embeddings averaging method:

**TF-IDF Down-Sampling**  TF-IDF is a standard method for selecting discriminative words in the documents. We choose the words with highest TF-IDF scores to represent a job posting.

**Choosing Tags**  Tags inside a document often contain important and concise information. For XING’s job postings, the tag sections usually contain 5 to 20 words which usually describe the skills needed, the position and etc. For example, tags for one of the XING’s job ads for web developer is: “Leipzig, php, css, jquery, mysql, web development, web developer, web entwickler, typo, web programmierung, entwicklung, ajax”. The tags contain key information of what the job seekers are looking for. We can generate document vectors by only averaging the embedding of words in the tags.

### 3-2 Paragraph Vector

Paragraph2Vec is an unsupervised learning method for learning the distributed vector representation for various length texts and was successfully applied for various tasks in [6]. We use the PV-DM model to learn document representations for XING’s job postings. As illustrated in Figure 2-3, in the input layer, we take the posting id in the first part of the job posting as the paragraph token. Through the learning process in a posting, the paragraph token will not change. The model learns each paragraph token as a semantic vector, which is used to represent the job posting. Figure 3-1 demonstrates the learned embeddings using t-SNE for 100 job postings being similar with the job posting ‘2597425’.

### 3-3 Word Mover’s Distance

As discussed in Chapter 2, the Word Mover’s Distance (WMD) estimates the distance between two text documents as the instance of the Earth Mover Distance (EMD). The same pre-trained word vectors as that we used in the Word Aggregation experiment are applied to estimate the distance between words. The time complexity for EMD is $O(n^3 \log n)$, where $n$ stands for the number of unique words in the documents [12]. What’s more, it takes several seconds to compute the WMD between two job postings on a standard laptop, rendering the method is unfeasible for using on large-scale collections. Since it is computationally expensive to find the k-nearest neighbors using WMD for a query when the corpus is large [5], to address this problem, we use two lower bounds to prune away much documents without computing their WMD distances with the query.

#### 3-3-1 Document Centroid Distance

Rubner et al. proposed an easy-to-compute lower bound which is the distance between their “centroids”:
In this lower bound calculation, every document is represented by its weighted average of word vectors. To simplify the computation, we directly re-use the average document vectors in the XING Posting Representation Method, where every word in the posting is weighted equally. This lower bound can significantly narrows the searching for the promising candidates in the finding of the k-nearest neighbors, resulting in a much faster approximated solution. Our experiments shows that it takes around one minute to compute the pairwise Document Centroid Distance between all documents (3000,000 job postings) and a query.

### 3-3-2 Relaxed Word Mover Distance

Reviewing Equation 2-6, a tighter bound of WMD can be obtained by removing one of the two constraints in the objective function WMD. They are:

\[
\min_{F \geq 0} \sum_{w_i \in d} \sum_{w_j \in d'} F_{w_i,w_j} c(w_i,w_j) \\
\text{subject to: } \sum_{w_j} F_{w_i,w_j} = d_i
\]

and

\[
\min_{F \geq 0} \sum_{w_i \in d} \sum_{w_j \in d'} F_{w_i,w_j} c(w_i,w_j) \\
\text{subject to: } \sum_{w_i} F_{w_i,w_j} = d'_j
\]

Following the work of Kusner et al., Equation 3-4 indicates that the optimal solution for the first equation is that finding the most similar word vector \(v_j\) in the document \(d'\) for each word vector \(v_i\) in the document \(d\), which means all weights of \(v_i\) are “flow” to \(v_j\) in the flow map.
Similarly, the optimal solution for the second equation is to do it in a reverse way, finding the most similar word vector $v_i$ in the document $d$ for each word vector $v_j$ in the document $d'$. 

$$\sum_{w_j \in d'} F_{w_iw_j} c(w_i, w_j) \geq \sum_{w_j \in d'} F_{w_iw_j} c(w_i, w_j^*)$$

$$= c(w_i, w_j^*) \sum_{w_j \in d'} F_{w_iw_j}$$

$$= c(w_i, w_j^*) d_i$$

(3-4)

Where $j^* = \arg \min_j c(w_i, w_j)$, and $d_i$ is the term frequency of word $w_i$ in document $d$.

The two relaxed solutions are combined to get a tighter bound, which they refer to as $\text{RWMD}(d, d') = \max(\text{RWMD1}(d, d'), \text{RWMD2}(d, d'))$. Our experiments show that the RWMD is more efficient by pruning around 97% of the job postings. We follow these steps for the fast $k$-nearest neighbors searching:

1. For a job posting query, sorting all postings by their Document Centroid Distances with the query, which are very cheap to compute.
2. For the top 20 postings according to the Document Centroid Distance, calculating the WMD between the query and each of them, and putting them into a list.
3. For the remaining postings, checking if their RWMD is larger than the largest WMD in the list. If so, pruning it. If not, calculating its WMD with the query and update the list.

Figure 3-1: T-SNE visualization of posting ids in the semantic vector space. The label for each point is the posting id for each posting.
(a) 937 words randomly selected from the learned vocabulary

(b) Words related to city and country

(c) Words related to computer science

Figure 3-2: T-SNE visualizations of words in the semantic embedding.
Figure 3-3: Histogram of pairwise Euclidean distance with different words selection methods for vector averaging: Five words with highest TF-IDF scores, five randomly selected words, and all words. The x-axis is the Cosine distance between document vectors while the y-axis represents the amount of document pairs with that distance.
Chapter 4

Experiments and Evaluations

In this chapter, we compare XING’s baseline to our methods for document representations. For the evaluation, we build a “ground truth” dataset using the user-posting interaction logs.

4-1 Dataset Description and Setup

All experiments are conducted on data sets provided by XING representing subset snapshots of the overall XING database. The datasets contain interactions (Table 4-1) and job postings (Table 4-3) starting from Jan 1st 2014 till Feb 15th 2015.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Total Iterations</th>
<th>Number of Distinct Users</th>
<th>Number of Distinct Postings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clicks</td>
<td>43152056</td>
<td>2496734</td>
<td>1832061</td>
</tr>
<tr>
<td>Bookmarks</td>
<td>782831</td>
<td>219692</td>
<td>314416</td>
</tr>
<tr>
<td>Replies</td>
<td>1686301</td>
<td>383044</td>
<td>414222</td>
</tr>
</tbody>
</table>

In interaction logs, the system recorded which user interacted with which job posting, together with the type of interaction and a timestamp. Specifically, there are three types of interactions: clicking (the user clicked a posting), bookmarking (the user bookmarked a posting), and replying (the user sent a message to the owner of the job posting). All of these interactions indicate user’s interests for the posting. To simplify the evaluation process, we merged all these logs together and build a User-Posting Interaction log. A snippet of this log is given in Table 4-2.

4-2 Evaluation Method

To evaluate their recommender system, XING uses on-line A/B testing. Unfortunately, we did not have access to an on-line evaluation setting and therefore we had to evaluate the
Table 4-2: Snippet of XING’s User-Posting Interaction log

<table>
<thead>
<tr>
<th>User Id</th>
<th>Posting Id</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>9330156</td>
<td>2462806</td>
<td>2014-01-01 00:01:48</td>
</tr>
<tr>
<td>9330156</td>
<td>2460585</td>
<td>2014-01-01 00:03:07</td>
</tr>
<tr>
<td>9330156</td>
<td>2455769</td>
<td>2014-01-01 00:04:14</td>
</tr>
<tr>
<td>2934883</td>
<td>2447430</td>
<td>2014-01-01 00:04:33</td>
</tr>
<tr>
<td>9330156</td>
<td>2452356</td>
<td>2014-01-01 00:05:08</td>
</tr>
<tr>
<td>4676628</td>
<td>2481579</td>
<td>2014-01-01 00:06:52</td>
</tr>
<tr>
<td>8201030</td>
<td>2480940</td>
<td>2014-01-01 00:40:06</td>
</tr>
<tr>
<td>4695434</td>
<td>2458772</td>
<td>2014-01-01 01:27:38</td>
</tr>
<tr>
<td>7043919</td>
<td>2481290</td>
<td>2014-01-01 01:29:02</td>
</tr>
<tr>
<td>2229333</td>
<td>2477997</td>
<td>2014-01-01 01:45:34</td>
</tr>
<tr>
<td>8631631</td>
<td>2481232</td>
<td>2014-01-01 02:04:51</td>
</tr>
<tr>
<td>3361901</td>
<td>2481820</td>
<td>2014-01-01 03:18:13</td>
</tr>
</tbody>
</table>

Table 4-3: Job Postings in XING’s Dataset

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Number of Distinct Postings</td>
<td>3019196</td>
</tr>
<tr>
<td>Posting on Performed Interaction</td>
<td>1835667</td>
</tr>
</tbody>
</table>

Experiments and Evaluations

We extracted a ground truth dataset from the User-Posting Interaction log showed in Table 4-1 and Table 4-2, which contains the interactions users did with job postings, including clicking, bookmarking, and replying. In this dataset, on average each job postings was interacted with by 19 users, and on average each user has interacted with around 14 job postings. According to XING, the interactions by a single user are often made with similar job postings. Reasoning that a strong indication of similarity between jobs is when the same user interacts with both, we consider that two jobs are similar when at least 2 users interacted with both. For each job, a list of similar jobs can be extracted which we use as the ground truth. The reason for choosing at least 2 users is to reduce the misinterpretation due to users’ mis-interactions. Figure 4-1 shows how we define similar items based on the User-Posting Interaction log.

For testing, we randomly sampled 5000 postings that have similar jobs in the ground truth set from the total job postings collection (in Table 4-3). To evaluate a model, for each job in the sample set, we retrieved a ranked list of top 20 most similar postings, and then compared it with the list extracted from the ground truth. That is to say, models are evaluated against the ground truth based on its effectiveness of estimating similarities between job postings. The evaluation metrics are: Recall@k, Precision@k, and F1@k, where k = [4, 10, 20] in accordance to the evaluation metrics XING uses internally. (Note: in XING’s homepage, they normally list 20 job recommendations for their customers)
Figure 4-1: The definition of similar postings: Two postings $p_1$ and $p_2$ are considered to be similar if they were co-interacted by at least 2 users.

4-3 Word Vector Aggregation

We used the latest English Wikipedia, German Wikipedia and 3,000,000 job postings from XING to jointly learn word embeddings over all three collections. For document representation, we estimated job posting vectors using three methods discussed in Chapter 3:

- **All**: the average over the word embeddings of the words that appear in the document.
- **TF-IDF**: the average over the word embeddings of five words with highest TF-IDF scores.
- **Tags**: the average over the word embeddings of the words that appear in tags.

average all word vectors, average 5 word vectors which highest TF-IDF scores, average vectors of tags (tag and skill parts in the job posting). We plotted the TF-IDF scores of the words in a large (specify large) number of documents, ordered by their TF-IDF score, and observed that an elbow appears in this plot around the fifth word. As an example, Figure 4-2 shows the TF-IDF scores for terms in a document. Therefore, we experimented with selecting 5 highest TF-IDF score terms to represent a document.

Table 4-4 shows the results for different evaluation metrics for different aggregation methods when learning word embedding on 500 dimensions. we also experimented by changing the dimensionality of the learned word embeddings on the dimension sizes of 100, 200, 300, 400, and 500. When training Word2Vec model on 500 dimension, all methods obtain largest F1 scores.

Figure 4-3 shows the F1@20 scores for different aggregation methods when learning word embedding on different dimensions. To get these results, we learned Word2Vec models on different vector sizes. For each setting of the dimensionality, all methods use the same Word2Vec model and the only difference is that they use different aggregation methods to generate document vectors. In Chapter 3, we hypothesized that using a selection of words gives a better representation of job postings. However, averaging the highest TF-IDF scores word vectors...
and vectors of tags are less effective than the XING Posting Representation method. Therefore we reject the hypothesis and we analyse there are two reasons for the less effectiveness of the selection methods:

1. We chased a fixed length of the highest TF-IDF words using the average of the elbow in the plots. However, the elbow for the TF-IDF scores plotting varies from documents. More adaptive methods for selecting various length of the highest TF-IDF words should be implemented.

2. The tags are not automatically generated by XING’s system or choosen by the job posting owners, but are written by the job postings owners. Therefore, the content they write for tag is not fixed, and may not be formal.

For the XING Posting Representation method (we re-implemented it with other methods), Figure 4-3 shows that the higher the word vector dimension is learned, the higher the F1@20 scores are.

### 4-4 Word Mover’s Distance

The Word Mover’s Distance is a special case of the Earth Mover Distance [5]. For this experiment, we implemented a Python version of the WMD (code for our implementation is available at [https://github.com/Hennrik/WMD-Python](https://github.com/Hennrik/WMD-Python)), which is based on the pipeline in...
<table>
<thead>
<tr>
<th>Evaluation Metric</th>
<th>Aggregation Method</th>
<th>All</th>
<th>TF-IDF</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>recall@4</td>
<td></td>
<td>0.00900</td>
<td>0.00486</td>
<td>0.00571</td>
</tr>
<tr>
<td>recall@10</td>
<td></td>
<td>0.01183</td>
<td>0.00700</td>
<td>0.00708</td>
</tr>
<tr>
<td>recall@20</td>
<td></td>
<td>0.01502</td>
<td>0.00848</td>
<td>0.00854</td>
</tr>
<tr>
<td>precision@4</td>
<td></td>
<td>0.02930</td>
<td>0.01800</td>
<td>0.02263</td>
</tr>
<tr>
<td>precision@10</td>
<td></td>
<td>0.04901</td>
<td>0.03026</td>
<td>0.04105</td>
</tr>
<tr>
<td>precision@20</td>
<td></td>
<td>0.07092</td>
<td>0.04280</td>
<td>0.06411</td>
</tr>
<tr>
<td>F1@4</td>
<td></td>
<td>0.00543</td>
<td>0.00275</td>
<td>0.00338</td>
</tr>
<tr>
<td>F1@10</td>
<td></td>
<td>0.00772</td>
<td>0.00427</td>
<td>0.00467</td>
</tr>
<tr>
<td>F1@20</td>
<td></td>
<td>0.00892</td>
<td>0.00546</td>
<td>0.00595</td>
</tr>
</tbody>
</table>

[5]. In our implementation, word distance was estimated by the Euclidean Distance calculated against the Word2Vec vector space. We used the same word vectors as that used in the experiments of the word aggregation methods. Being the same as other methods, our objective is to get a ranked list of the 20 most similar postings for each query. We first learned a Word2Vec model on word vector dimensions equals 100, and for a query posting, we computed its Word Mover Distance with another posting. In a Virtual Machine with 32 cores and 100GB RAM, the time for computing Word Mover Distance between a pair of job postings was around several seconds in our experiments. Since we have around 3,000,000 postings, it is infeasible to compute the WMD between all job postings. To run an experiment within the available time, we employed the lower bound and the Relaxed Word Mover Distance (RWMD) to select potential candidates without calculating WMD for the majority of postings. Even with the implementation of lower bound and RWMD, we need around 1 hour to find the 20-nearest neighbors for just one query. In Chapter 3, we introduced the details of how to get the k-nearest neighbors for a query.

Constrained by time, we decided to randomly pick 100 job postings to obtain an indication of whether WMD can improve the similarity estimation between job postings. For each posting in these 100 job postings, we obtained a ranked list of the 20 most similar postings and then evaluated it with our evaluation metrics. Table 4-5 shows the results for experiments on the Word Mover’s Distance. Although the precision is high compared to the other approaches, we also observe that recall and F-score are lower.

4-5 Paragraph Vector

We learn embeddings for job postings using Doc2Vec in the deeping learning library Gensim\(^1\), which is a Paragraph2Vec implementation in Python. The paragraph vectors are learned over 3,000,000 XING’s job postings. To shorten the time needed for the experiment, we fixed

\(^1\)https://radimrehurek.com/gensim/models/doc2vec.html
several parameters to the default settings: Window Size = 8, Min_count = 2 (filtering out words with frequency less than 2), dm = 1 (PV-DM model), Iteration = 20. Vectors for job posting are learned by using a list of different paragraph vector sizes: 100, 200, 300, 400 and 500. We could not train vectors with a dimensionality higher than 500 because we had insufficient memory for the computation. For each vector size, after the learning process, pairwise distances between posting vectors are estimated by the Cosine Distance. Table 4-6 shows the results for Paragraph2Vec when training document embeddings on 500 dimension.

The results show an improvement in F-Measure when we increase the dimensionality of the paragraph vectors. Figure 4-4 shows the F1@20 scores for using XING Posting Representation method and Paragraph2Vec with different vector sizes. When vector size equals 500, we got the highest F1@20 score **0.00970** for Paragraph2Vec, which is 8.1% higher than XING Posting Representation method which is **0.00898**. This indicates that for estimating similarity between job postings, the Paragraph2Vec is more effective than XING Posting Representation method.
Figure 4-4: F1@20 scores for using XING Posting Representation method and Paragraph2Vec to learn document embeddings with different vector sizes.

Table 4-5: Scores of different evaluation metrics of the Word Mover Distance when learning word embedding on 100 dimensions

<table>
<thead>
<tr>
<th>Evaluation Metrics</th>
<th>Scores</th>
</tr>
</thead>
<tbody>
<tr>
<td>recall2</td>
<td>0.00103341348951</td>
</tr>
<tr>
<td>recall4</td>
<td>0.00110389143059</td>
</tr>
<tr>
<td>recall10</td>
<td>0.0014827008349</td>
</tr>
<tr>
<td>recall20</td>
<td>0.00464521206753</td>
</tr>
<tr>
<td>precision2</td>
<td>0.03</td>
</tr>
<tr>
<td>precision4</td>
<td>0.05</td>
</tr>
<tr>
<td>precision10</td>
<td>0.13</td>
</tr>
<tr>
<td>precision20</td>
<td>0.235</td>
</tr>
<tr>
<td>f2</td>
<td>0.00177643323881</td>
</tr>
<tr>
<td>f4</td>
<td>0.00191689109188</td>
</tr>
<tr>
<td>f10</td>
<td>0.0026704230057</td>
</tr>
<tr>
<td>f20</td>
<td>0.00724784474019</td>
</tr>
</tbody>
</table>
Table 4-6: Evaluation Metrics for Paragraph2Vec when training word embedding on 500 dimension

<table>
<thead>
<tr>
<th>Evaluation Metric</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>recall@4</td>
<td>0.01017</td>
</tr>
<tr>
<td>recall@10</td>
<td>0.01377</td>
</tr>
<tr>
<td>recall@20</td>
<td>0.01644</td>
</tr>
<tr>
<td>precision@4</td>
<td>0.03005</td>
</tr>
<tr>
<td>precision@10</td>
<td>0.05169</td>
</tr>
<tr>
<td>precision@20</td>
<td>0.07430</td>
</tr>
<tr>
<td>F1@4</td>
<td>0.00531</td>
</tr>
<tr>
<td>F1@10</td>
<td>0.00764</td>
</tr>
<tr>
<td>F1@20</td>
<td>0.00970</td>
</tr>
</tbody>
</table>
In this chapter, we experiment with a novel approach called Link2Vec, for which we learn item representation from the text context of hyperlinks that refer to a page about that item in a central repository such as Wikipedia. In the absence of hyperlinks that point towards XING’s job postings, we opted to use a different dataset for this Chapter. However, being similar to the previous chapters, this experiment also aims to improve the representation of items for the task of recommending items to users.

5-1 Introduction

Hyperlinks are everywhere on the websites. Koolen and Kamps show that the hyperlink information in the web-page can be useful for web search [4]. Some specific hyperlinks, such as the IMDB url for a movie, the Wikipedia link for a singer, or a Amazon url for a book, can be seen as the representations of the item which they refer to. The contexts of a hyperlink may contain descriptions of the item, discuss interesting aspects, or contain a user’s opinion. Figure 5-1 shows three paragraphs of the contexts surrounding a Wikipedia hyperlink of the movie “The Matrix”. If we search the word “Matrix”, it is easy to confuse the search engine because it does not know whether this word is for the movie or for the mathematics. Therefore, if we learn the Word2Vec model from some sources of dataset, the word “Matrix” may ends up with some mathematical meaning. However, when we look at the hyperlink for the Wikipedia page of the movie “The Matrix”, it does not has any ambiguity and it is unique, which is one of the good properties of hyperlinks in representing the items.

In our exploration, we found that in some web pages, the authors put contexts of the hyperlinks to talk about or show some related thing of these items, while some of others are just list these hyperlinks. Similar to learning the meaning of a word by the context it appears in, we can learn a meaning for the referred item by its context. According to XING, their job postings are rarely referred to by external web pages, therefore, for this experiment we use Movielens 1M. Our proposed model is a variation of the Word2Vec framework, and it uses the information of hyperlinks to learn embeddings for the items, therefore we name it Link2Vec.
It’s a slow awakening. As with characters from The Matrix movie, the dawn of reality is an unpleasant one for many. Some of us just want to drift off back to sleep. We’d choose to surrender to the machines, even though we know it’s a facade, a farce, a computer-generated illusion.

(a)

The book also looks at the future of robotics, one of the more science fictional elements of the book. It is predicted that humanoid robots will join the battlefield in the next ten or so years, alongside flesh-and-blood soldiers, that leaders might have robotic AI aides, and that the very nature of leadership is changing with instant communications. Like anyone who is a fan of science fiction, Singer also looks at the possibility of a robotic revolution, such as what has been seen in the Terminator, Battlestar Galactica and the Matrix, where machines come to know that they can be better than humans and push us aside. While this is taken a bit with a grain of salt, it’s certainly a concern, and even some soldiers note that they’re working on something that might end up causing problems for their grand kids. If robots do rise up, I don’t know that we’d have a chance.

(b)

When you look at how Google has positioned itself as a facilitator and future shareholder/owner of nearly every new innovation coming out of nearly every sci-tech and digital dev shop within its long reach – you can see Tyrell in Google. Powering a cybernetic life form with AI will be key for realizing the ultimate Google dream, or perhaps its much darker than that – drawing power off of humans as is illustrated in the pod farm scene (see image below), as depicted in the 1999 film, The Matrix.

(c)

Figure 5-1: Three paragraphs of the contexts for the hyperlink. The words with color are the anchor texts for this hyperlink. The hyperlink in the page is link to the Wikipedia page for the movie “The Matrix”.

5-2 Experiment Setup and Evaluation

5-2-1 Dataset Description and Setup

Initially, we planned to crawl contexts in web pages which contain hyperlinks for IMDB\(^1\) movie pages. To that end, we built a mappings from movie ids in MovieLens 1M dataset\(^2\) to their corresponding IMDB url pages. After that, Google command link was used to retrieve web pages contains a specific IMDB movie url. For example, link: http://www.imdb.com/title/tt0133093/ can be used to get web pages which contains the IMDB url for the movie “The Matrix” from Google. The problem for obtaining these web pages is that Google will block our IP if we request so often or extent a limited amount times. We solved this problem by using Tor\(^3\) and forcing it to change IP address when our current IP is blocked by Google.

\(^1\)http://www.imdb.com/
\(^2\)http://grouplens.org/datasets/movielens/1m/
\(^3\)https://www.torproject.org/
Even through these methods worked perfectly for crawling Google’s result pages, we obtained limited pages for a IMDB movie url, which is on average 5 pages.

Constrained by the limited amount of pages Google returns, we turned to another source of links for movies. We built mappings from movie ids in MovieLens 1M Dataset to their corresponding English Wikipedia pages, resulting in 3301 mappings. These English Wikipedia links were used to represent movies in our set. We retrieved each movie in this set by searching its English Wikipedia url through Wikireverse4 (“WikiReverse shows pages that link to Wikipedia articles. It is a dataset of 36 million links parsed from the July 2014 web crawl released by Common Crawl.”). Figure 5-2 demonstrates an example of the result page in Wikireverse when retrieving the English Wikipedia link of the movie “The Matrix”. After that, we crawled the result pages to get the outer links for each movie, and created another mapping from movie id to outer links. For the term “outer link”, we mean the link to the page that contains a hyperlink to a Wikipedia movie article. There are on average 33 links to the Wikepedia page of a movie in MovieLens 1M Dataset. Then, like Figure 5-1 shows, for each movie, in each outer link’s page, we retrieved the page to extract the sentences surrounding the hyperlinks and merged these sentences as the contents for that movie. We replaced the anchor text in each page to a special token. For example, for the movie “Matrix”, we changed the anchor text for the hyperlink in each outer link’s web page to “IMDBSPECIALMOVIESEN_TOKEN2571”. Thus, we can learn each movie as a normal token.

What’s more, we noticed that the movie’s full title appeared in the Wikipedia page for the movie is referred to this movie itself. Therefore, we processed the whole content of the Wikipedia page, and considered every occurrence of this movie’s full title as “special”

---

4https://wikireverse.org/
hyperlinks, and replaced them with the same tokens with our hyperlinks, which enlarged our dataset for learning Link2Vec model. Here is an example of how we conduct these replacements:

The Matrix was first released in the United States on March 31, 1999, and grossed over $460 million worldwide. It was generally well-received by critics, and won four Academy Awards as well as other accolades including BAFTA Awards and Saturn Awards. Reviewers praised The Matrix for its innovative visual effects, cinematography and its entertainment.

IMDBSPECIALMOVIENTOKEN2571 was first released in the United States on March 31, 1999, and grossed over $460 million worldwide. It was generally well-received by critics, and won four Academy Awards as well as other accolades including BAFTA Awards and Saturn Awards. Reviewers praised IMDBSPECIALMOVIENTOKEN2571 for its innovative visual effects, cinematography and its entertainment.

where the number 2571 is an unique id for the movie “The Matrix” in MovieLens Dataset.

The learned semantic meaning for each special movie token is highly affected by its context words. To improve the accuracy of the learned semantic meaning for words in general, we added the latest (at the date of 11/10/2016) English Wikipedia Dataset in our learning dataset. The final learning set consists of the modified Wikipedia pages for movies, the scraped sentences for Wikipedia links, and the latest English Wikipedia corpus. After the learning process, every movie token is represented by a semantic vector.

5-2-2 Evaluation

For the evaluation, we compared against an approach that was recently proposed by Musto et al. [11], which learns Word2Vec models with word vector dimensions in 300 and 500 by using English Wikipedia dataset and represents every movie in the MovieLens 1M Dataset by the average of all embeddings of words that appear in this movie’s Wikipedia page. In the remaining pages, we refer it as Simple Average Method. For a fair comparison, we follow their recommendation pipeline and evaluation method.

Recommendation Pipeline:

1. Semantic vector $v$ are directly learned to represent each special movie token $i$.

2. Given a set of users $U$, the vector representation of each user profile is learnt as the centroid of the movies that the user previously liked.

3. Given vector representations for user profiles and movies, for each user $u$, Cosine similarities are calculated between its profile and movies: top $k$ movie ids with largest similarities are returned as the recommendations to the user.

Evaluation Method:
We use ratings [3] in MovieLens 1M Dataset, which contains 1,000,209 anonymous ratings of approximately 3,900 movies made by 6,040 MovieLens users and we only keep the movies which are learned by our models, which is around 3200 movies. In this rating dataset, movies are rated in 5 point scale (1 to 5). We consider ratings of 4 and 5 as positive ratings, which means the users like them. For the Simple Average Method and the Link2Vec, we carry out 5-folds cross validation. Following the steps in the Recommendation Pipeline, the training set is used to build the user profile while the test set is for evaluating the returned recommendations.

5-3 Results

Table 5-1 and Table 5-2 show the results of the experiments among these two document embedding techniques. We re-implemented the Simple Average Method strictly following their main steps. However, we were not able to reproduce the score reported in [11], where they have F1 score around 0.55. We observe that Link2Vec is less effective than the Simple Average Method. Since for these experiments we were only able to retrieve a limited set of hyperlinks that refer to the movies’ Wikipedia pages, and also given that Word2Vec was often found to provide more accurate embeddings when trained with more data [9], it may be that Link2Vec embeddings will become more accurate if more training data is used. However, in this experiment we were not able to show that Link2Vec may be useful for item recommendation.

<table>
<thead>
<tr>
<th>Evaluation Metric</th>
<th>Methods</th>
<th>Simple Average Method(300)</th>
<th>Link2Vec(300)</th>
</tr>
</thead>
<tbody>
<tr>
<td>F@5</td>
<td></td>
<td>0.01819</td>
<td>0.01720</td>
</tr>
<tr>
<td>F@10</td>
<td></td>
<td>0.02532</td>
<td>0.02267</td>
</tr>
<tr>
<td>F@15</td>
<td></td>
<td>0.02945</td>
<td>0.02501</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Evaluation Metric</th>
<th>Methods</th>
<th>Simple Average Method(500)</th>
<th>Link2Vec(500)</th>
</tr>
</thead>
<tbody>
<tr>
<td>F@5</td>
<td></td>
<td>0.01775</td>
<td>0.01821</td>
</tr>
<tr>
<td>F@10</td>
<td></td>
<td>0.02489</td>
<td>0.02356</td>
</tr>
<tr>
<td>F@15</td>
<td></td>
<td>0.02881</td>
<td>0.02644</td>
</tr>
</tbody>
</table>
5-4 Discussion of Link2Vec

Apart from the results above, we did additional experiments. These experiments reveal some interesting parts of Link2Vec and show right directions for the future work of implementing Link2Vec. Our augmentations are as follows:

1. Larger Training Dataset
   One of our experiment which tried to learn a Link2Vec model by removing the modified (changing movie name to special token) Wikipedia pages in the training dataset shows the F1@15 score is around 0.013, which is much worse than including the modified Wikipedia pages. This supports the intuition that using more training data improves the accuracy of Link2Vec. We also noticed that in the Wikireverse’s result pages, we can not get some pages which we know them exactly contain a retrieved hyperlink of a Wikipedia page for a movie, which reduces the outer links we can obtain for a query. For the evaluation of using more content, we leave it to the future work. We can consider extracting page links for web pages containing Wikipedia urls, IMDB urls, or Rotten Tomatoes urls for movies from the Common Crawl corpus\(^5\) (“The Common Crawl corpus contains petabytes of data collected over the last 7 years. It contains raw web page data, extracted metadata and text extractions.”), which could helps us to get more contents for the hyperlinks.

2. Combination of Link2Vec and the Simple Average method
   We found that the recommendation lists made by Link2Vec and Simple Average Method are not identical. Our experiments show that for two recommendation lists with the length of 15 made by Link2Vec and Simple Average Method, there are on average 0.662 movies in common. Moreover, in term of the relevant movies (movies also in the test set) in the recommendation lists, there are on average 0.05765 movies in common. These indicate that the Link2Vec actually allows us to use the text information in a different way. Therefore, in the future work, we could consider combining the Link2Vec with the Simple Average Method to create a hybrid recommender system which probably performances better than either one individually.

\(^5\)http://commoncrawl.org/
Chapter 6

Conclusions

In this study, we evaluated several methods for the distributed representations of XING’s job postings. We hypothesized that the word selection averaging methods result in closer positions of similar job postings than the original XING Posting Representation method. However, the experiments show that using the average over all words results in a better representation to find similar jobs. In another experiment, we show that using Paragraph2Vec to learn document embeddings outperformed the XING Posting Representation method for job posting similarity estimation when using more than 200 dimensions. The Paragraph2Vec learns relationships between paragraph token and words in a paragraph. From this logic, the paragraph token can acts as the topic of the paragraph, and the corresponding paragraph vector can be more reliable to represent a paragraph. We observed an increase in effectiveness when using Paragraph2Vec with a higher dimensionality. With a larger computation power, XING can learn paragraph vectors larger than 500 dimensions, and finish the testing of the Paragraph2Vec’s performance.

We also experimented with Word Mover’s Distance, for which the job postings are just represented by the words they contain, and WMD estimates the distance between documents by using Earth Mover’s Distance technique and by taking the Euclidean distances between words in an embedding space as the travel costs. Given the computational complexity and limited amount of time, we were only able to compute a small number of samples. Although the Precision for the tested job postings was high compared to the other experiments, Recall was considerably lower, indicating that the test set may have been skewed towards popular items.

In the last Chapter, we introduced Link2Vec. The results for Link2Vec on Movielens 1M are slightly less than a method that uses the average embeddings of a movie’s Wikipedia page. However, since we only obtained a limited set of hyperlinks for the movies in MovieLens, an interesting direction for future work may be to use Link2Vec with a larger training set. What’s more, since Link2Vec allows us to use the text information in a different way, another promising direction for future work may be to combine the Link2Vec with the Simple Average Method to create a hybrid recommender system.


