Roma Tre University

Ph.D. in Computer Science and Engineering

Deep Learning Models for Research Paper Recommender Systems

Hebatallah Atef Ibrahim Mohamed Hassan

Deep Learning Models for Research Paper Recommender Systems

A thesis presented by

Hebatallah Atef Ibrahim Mohamed Hassan

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in Computer Science and Engineering

Roma Tre University

Department of Engineering

Spring 2020

Abstract

The amount of scholarly publications has been rapidly increasing during the last decades. The need to intelligently process such amount and produce recommendations to researchers is becoming urgent.

In the last years, Recommender System (RS) approaches have emerged to facilitate finding publications related to the researchers' area of interest. However, natural language ambiguity is still a challenge to generate accurate recommendations exploiting textual features. The goal of this doctoral thesis is to propose deep learning models, which could learn semantic representations of research papers in order to obtain effective recommendations. In other words, proposing models that help in providing recommendations based on the semantic similarity between research papers.

In this study, we make several contributions that address the problem of considering the semantic similarity between research papers: Firstly, we propose a supervised approach that adopts gated recurrent networks with attention mechanism, for aggregating important words and sentences from research paper titles and abstracts, in order to increase the general representation and visualization of the key concepts in research papers. This approach has been exploited for predicting social tags from research papers, since tags help in organizing, sharing and even recommending research papers. Secondly, we propose a tag-aware research paper recommendation approach, that utilizes the same model proposed for tag prediction, in extracting tag-based document representations. We show how semantic document representations based on social tags can be combined with the traditional collaborative filtering methods to yield superior performance with any number of ratings. Finally, we propose an unsupervised approach for non-personalized research paper recommendations, which leverages pre-trained sentence encoders based on deep learning models.

Acknowledgments

It would not have been possible to complete my PhD thesis without the support of so many great people.

First, I would like to express my gratitude and appreciation to my PhD advisor Prof. Alessandro Micarelli, for giving me the opportunity to join the *Artificial Intelligence (AI)* lab, motivating and supporting me during the years of my study. I sincerely thank my second advisor Dr. Giuseppe Sansonetti for his continuous support and assistance during the time of research. His invaluable comments helped me to improve the quality of this work. Besides my advisors, I would also like to thank Dr. Fabio Gasparetti, Prof. Carla Limongelli and my colleagues Matteo Amadei, Marzia Dominici and Carlo De Medio, for their insightful discussions, and for their support and motivation.

I am very grateful to Dr. Joeran Beel who gave me the opportunity to do a research visit at his group, in the ADAPT Centre at Trinity College Dublin. His guidance and collaboration have made it possible for me to achieve my potential. I would also like to thank his team members: Andrew Collins, Alan

Griffin and Conor O'Shea, who helped me to understand their recommender system, and provided me with a nice and friendly environment during my internship.

I would like to thank all my friends at home in Egypt and abroad, for their encouragement and enthusiasm during the years of my study. Special thanks to my friends in Rome: Maria-Aisha, Shadi and Niloofar, they helped make my time in Rome more fun and productive. I am also grateful for all the amazing people I had the chance to get to know at conferences and summer schools.

Last but not least, I am very grateful to my beloved family for their love, patience, encouragement and support throughout my journey. My Mom and my best friend, Ms. Ibtesam Hamed. My Dad and my role model, Dr. Atef Hassan, *Fellow of the Royal College of Surgeons in Edinburgh and Ireland.* My wonderful sisters Maha and Menna, brother Mohamed. I wouldn't be where I am today without you all.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The massive growth of information on the Internet makes finding information both challenging and time consuming. Traditional search engines require the user to manually enter keywords in order to search for relevant information. The results of the search query are displayed to the user based on the order of relevance to the keywords. One of the problems with traditional keyword-based search engines, is that the user may find it difficult to choose the search keywords able to return the best results, especially if the user is searching for information in a new domain.

Recommender systems (RSs) provide a solution to this problem by automatically capturing user preferences and recommending related information that may be of interest to the target user. There are two ways in which RSs are able to capture user preferences: explicitly, by enabling the user to enter her preferences, or implicitly, by monitoring the user's activities such as browsing the Web or reading documents. Collected preferences are stored in a user profile. New items (e.g., research papers) are then compared with the user profile and those items which are sufficiently similar are recommended to the user. Existing RSs offer efficient personalized services in a variety of domains such as movies, music, books, research papers and e-commerce [PKCK12].

In the context of research papers, RSs can facilitate for researchers organizing, finding and even sharing research papers. One critical challenge in research paper RSs, is capturing the semantic similarity between papers. As investigated in [Has17], a common practice is to model text in a document as a set of word features, i.e., bag-of-words (BOW). Often, some feature selection techniques are applied, such as stop-words removal or stemming, to only keep meaningful features. However, these representation methods do not have the capacity of modelling the semantics embedded in text data. In fact, a word can express different meanings and different words can be used to describe the same meaning. Such word ambiguities are often referred to as the *polysemy problem* and the *synonymy problem*, respectively.

A variety of methods have been proposed to consider the semantic relationships between words or entities. One traditional way of solving the word ambiguity problem is topic modelling (e.g., LDA [MLA14]) which applies statistical methods to analyze latent topics with associated words. By learning the distributions between topics and words, each document is represented as a linear combination of topics instead of words, resulting in a low dimensional representation. However, topic modelling methods derive the low-rank representation of documents using single words based on the corpus itself, which may not generate enough discriminative semantic information. Another way to solve the above problem is semantic annotation, which incorporates semantic information. These ontologies have limitations of coverage, e.g., WordNet[1] is an online dictionary that mainly covers lexical information rather than entities. A major challenge for semantic annotation is to find and utilize a comprehensive and domain-independent external knowledge that can cover all the semantic information mentioned in a corpus [GM09].

Deep learning models have recently shown great potential for learning effective representations and achieved state-of-the-art performance in many Natural Language Processing (NLP) applications. However, most of the focus in the research community has been on general purpose text, such as news or social media activity. Scientific text is notably different, with a much higher density of specialized terminology. In this thesis, we propose the long-term goal of understanding and interpreting scientific literature, and this work is intended to be one of the small first steps in this direction. We, therefore, focus on developing deep learning models for understanding semantics of scientific texts for the recommendation task.

1.2 Research Goals

The general aim of this thesis is "*to propose deep learning models that could capture semantic similarity between research papers, in order to obtain effective recommendations*". To do this, we have pursued the following research goals.

RG1: Survey the existing approaches of research paper RSs and investigate if any of them have utilized deep learning for content representation. It is crucial to survey the current approaches of research paper RSs and study their limitations. Therefore, we conduct a survey list-

[1]https://wordnet.princeton.edu/

ing the existing approaches categorized into three types of RSs: *content-based filtering*, *collaborative filtering* and *hybrid* approaches. We investigate if word embeddings or more complex deep learning models are used in any of those approaches for content representation. We also study the limitations of the current research paper RSs. (Chapter 3).

RG2: Investigate the performance of word embeddings based methods on research papers, and propose an approach that could enhance content representation. We evaluate state-of-the-art word embeddings based techniques for sentence and document representation. Since we assume that research papers with similar tags have semantic relation, we evaluate those techniques on the task of tag prediction. Moreover, we propose a deep learning model that outperforms the state-of-the-art, for extracting semantic representation of research papers for the same task. (Chapter 4).

RG3: Explore the usefulness of research paper social tags assigned by users in recommending research papers. Since we believe that papers with common social tags are semantically related, we investigate if we can use tags as metadata for research papers, and we study how can we utilize those tags in extracting semantic document representations for research paper RSs. (Chapter 5).

RG4: Investigate how well pre-trained sentence encoders perform in non-personalized recommendation scenario. We experiment with some of the well-known pre-trained sentence encoders (e.g., Google's BERT [DCLT18]) in the task of identification of related research papers for non-profiled users. The goal is to identify which of the encoders could achieve good understanding of semantics, and how can we utilize them for recommendations. (Chapter 6).

1.3 Contributions

This thesis focuses on investigating possible deep learning approaches, mainly content-based ones, for RSs in the research papers domain. Three main contributions are provided as follows:

- Presenting a methodology that adopts a deep learning model, for extracting key concepts from research paper titles and abstracts, in order to obtain effective representations of research papers. This approach is exploited for tag prediction.
- Utilizing the same model of tag prediction in extracting semantic document representations, and incorporating those representations into a collaborative filtering method, resulting in tag-aware research paper RS.
- Proposing a recommendation approach for non-profiled users, in a system based on Apache Lucene[2] search engine. This approach exploits pre-trained deep sentence encoders, for extracting the semantic features of research papers. We implement this approach on a real system called *Darwin* & *Goliath*[3] [BGO19], which offers a recommendation-as-a-service and integrates with JabRef[4], a reference management software.

Some parts of this thesis have been published in international conferences. For instance, literature review of research paper RSs is included in [Has17]. Tag prediction for research papers is presented in [HSGM18]. Finally, scientific paper reranking and recommendation based on general purpose and pre-trained sentence encoders is presented in [HSGM19].

[2]https://lucene.apache.org/
[3]https://darwingoliath.com/
[4]http://www.jabref.org/

1.4 Thesis Outline

This thesis is structured as follows. Chapter 2 presents basic information that is crucial for understanding this doctoral thesis. It includes an introduction to neural networks, deep learning concepts, possible approaches for semantic textual representations such as word embeddings and sentence embeddings, and some other definitions.

In Chapter 3, a literature survey of the current research paper recommender system approaches is given, highlighting the different techniques used, mainly content-based, collaborative filtering and hybrid ones. This chapter also discusses some of the challenges of research paper RSs.

Chapter 4 presents an approach that adopts hierarchical recurrent neural networks, more specifically gated recurrent units (GRUs) with attention mechanism, for extracting key concepts from research paper titles and abstracts. The proposed approach is exploited for tag classification and prediction. We provide an experimental evaluation showing that this approach is effective in comparison with state-of-the-art techniques related to text features extraction.

Chapter 5 describes an approach that exploits the tag prediction model introduced in Chapter 4, for extracting tag-aware document representations, and incorporating those representations in a collaborative filtering method, resulting in a hybrid research paper RS. We demonstrate that this methodology is effective in comparison with considering BOW representation or tags as features of research papers.

Chapter 6 describes an approach for non-personalized research paper RSs. This approach is based on integrating pre-trained sentence encoders with traditional Apache Lucene/Solr BM25 technique. We provide an empirical evaluation between the performance of five well-known pre-trained sentence encoders in the same scenario. In addition to the offline evaluation, we present the results of a conducted user study showing the effectiveness of our proposed approach. Finally, the thesis is concluded in Chapter 7 by discussing the consequences of the findings, and potential directions for future research.

Chapter 2

Background and Definitions

2.1 Vector Space Model

The Vector Space Model (VSM) is a model for representing documents in numeric form. It is a widely accepted standard that is used whenever a numerical representation of text is needed [KMTD14, TLPP+14]. The basic idea of the VSM is to represent text as a bag-of-words (BOW). In VSM, the ordering of the words in a document is ignored and the document is represented by a vector of word frequencies.

More formally, a vocabulary is established where each w_i word or term has a unique integer index i. A document d_j is represented by a vector v_i where each element v_{ij} in the vector stores the $tf(i, j)$ frequency of word w_i in document d_j. For the standard VSM model, BOW is generalized such that each v_{ij} value does not necessarily show the exact term frequency, but stores a weight that represents a relevance measure of the term in the document. Some of the most popular weighting schemes are TF-IDF [SB88] and BM25 [RWJ+95]; more details in the following sections.

TF-IDF

TF-IDF stands for term frequency - inverse document frequency. The motivation behind TF-IDF is that, the BOW weighting scheme gives high weight to naturally more frequent words, which are not necessarily more relevant, such as stop-words. TF-IDF introduces the idea of document frequency, which is the number of documents containing a particular term. Using this additional measure, we can detect which words are naturally more common through the whole dataset and are given less weight in favor of terms that are more interesting and mark the difference between documents, i.e. frequent in a document but not across all the documents.

More formally, TF-IDF is the product of the Term Frequency (TF) and Inverse Document Frequency (IDF) scores of the term, as shown in the following equation:

$$TF\text{-}IDF = \frac{TF}{IDF} \tag{2.1}$$

TF summarizes how often a given word appears within a document. It can be calculated as follows:

$$TF = \frac{number\ of\ times\ the\ term\ appears\ in\ the\ document}{total\ number\ of\ words\ in\ the\ document} \tag{2.2}$$

IDF downscales words that appear a lot across documents. A term has a high IDF score if it appears in a few documents. Conversely, if the term is very common among documents (i.e., stop-words such as “the”, “a”, “is”), the term would have a low IDF score. IDF can be calculated as follows:

$$IDF = In(\frac{number\ of\ Documents}{number\ of\ Documents\ the\ term\ appears\ in}) \tag{2.3}$$

In conclusion, the higher the TF-IDF score, the rarer the term is.

BM25

BM25 is a ranking function that extends TF-IDF scheme and it is considered to be the ideal weighting for search. Given a query Q, containing keywords q_1,...,q_n, the BM25 score of a document D is calculated as following:

$$score(D,Q) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{TF(q_i,D) \cdot (k_1+1)}{TF(q_i,D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\mathrm{D_{avg}}}\right)} \tag{2.4}$$

where IDF(q_i) is the inverse document frequency of word q_i, TF(q_i, D) is its term frequency in document D, $|D|$ is the number of words in the document, D_{avg} is the average size of a document, and k_1 and b are free parameters.

Cosine Similarity

Similarity measure is an important aspect of the VSM model. The standard document comparison metric is cosine similarity [S+01]. The cosine similarity between two vectors (or two documents on the Vector Space) is a measure that calculates the cosine of the angle between them. It is affected by the terms the two vectors have in common. Cosine similarity thus has some meaningful semantics for ranking similar documents, based on mutual term frequency. It is calculated using the dot product and magnitude of each vector as shown in the following equation:

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|\|\vec{b}\|} \tag{2.5}$$

2.2 Deep Learning Models

Deep learning (DL) represents a huge step forward for machine learning. DL is based on the way the human brain processes information and learns. It has kept breaking barriers during the last years in the state-of-the-art of many predictive tasks [Sch15]. There is still no clear theoretical foundation that explains why Deep Neural Networks (DNN) are so effective. DNNs conquered the field of computer vision with the introduction of Convolutional Neural Networks (CNN) a few years ago. By now, DNNs have replaced most classical algorithms for pattern recognition tasks such as object recognition [VDDP18]. The same thing has happened in NLP with the introduction of Recurrent Neural Networks (RNN). Machine translation between common languages is now almost an easy task [CVMG+14]. There is work on document and image question answering, where the DNN seems to understand the contents in great detail and can retrieve or generate short snippets that answer very specific details about the media [XMS16]. Moreover, DNNs have also lead to breakthroughs in music processing, from classification to style transfer and composition [BHP17].

Artificial Neural Networks

Artificial neural network models (also called neural networks or NNs) are machine learning models consisting of chains of composed highly-parametric functions called *layers*. Neural network parameters can be initialized randomly and trained from data with or without corresponding labels (i.e. supervised or unsupervised), depending on the specific model architecture and they can learn to approximate potentially arbitrarily complex functions [HSW89].

As shown in Figure 2.1, a neural network is composed of input, hidden,

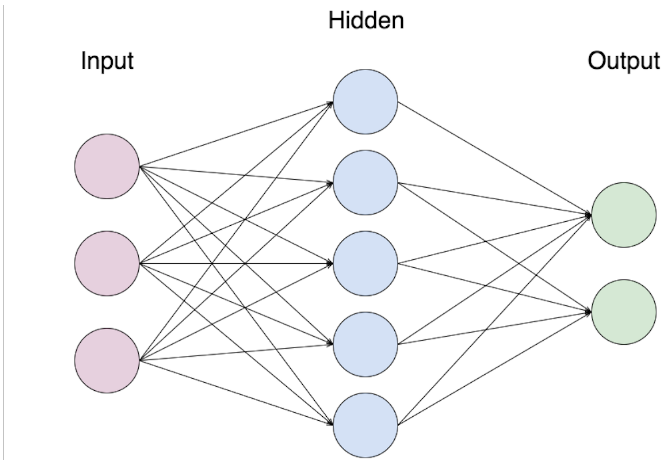


Figure 2.1: Artificial neural networks architecture.

and output layers — all of which are composed of nodes. Input layers take in a numerical representation of data (e.g., images with pixel specs), output layers output predictions, while hidden layers are correlated with most of the computation.

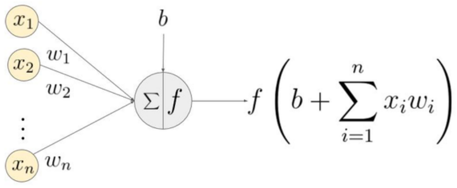


Figure 2.2: Neural network activation function.

As shown in Figure 2.2, each hidden node in the neural network has an activation function, consisted of weight and bias parameters — represented by

w and b respectively. These are essential to the actual learning process of a deep learning algorithm.

After the neural network passes its inputs all the way to its outputs, the network evaluates how good its prediction was (relative to the expected output) through something called a "loss function". As an example, the Mean Squared Error (MSE) loss function is calculated as follows:

$$\sum_{i=1}^{n} \left(Y_i - \hat{Y}_i\right)^2 \tag{2.6}$$

where $\hat{Y}$ represents the prediction, while Y represents the expected output. A mean is used if batches of inputs and outputs are used simultaneously (n represents sample count). The goal of the neural network is ultimately to minimize this loss by adjusting the weights and biases of the network. In using something called "back propagation" through gradient descent [MBBF00], the network backtracks through all its layers to update the weights and biases of every node in the opposite direction of the loss function — in other words, every iteration of back propagation should result in a smaller loss function than before [Nie15].

Neural network models use distributed representations of some fixed pre-specified dimensions as their sole means of representing information. Correspondingly, one of the major challenges in designing a neural network model architecture for some task is finding an appropriate way of transforming the task data into a fixed-size distributed form for input into the network. For some tasks, this is relatively straightforward: if the task involves images of a fixed-size, for example, each pixel of each image can be treated as one dimension in a distributed representation.

For text, though, there are two challenges: discrete symbols (generally words) must be transformed into continuous representations, and this transformation must be able to produce representations of a consistent dimension from sequences that vary in length. The former problem is called "word embedding", and the latter problem is called "sentence embedding"; they will be explained later in this chapter.

The deep part of deep learning refers to creating deep neural networks with a large amount of layers — with the addition of more weights and biases, the neural network improves its ability to perform more complex functions. And deep learning approaches can be categorized as follows: Supervised, semi-supervised or partially supervised, and unsupervised. In addition, there is another category of learning approach called Reinforcement Learning (RL) or Deep RL (DRL), which are often discussed under the scope of semi-supervised or sometimes under unsupervised learning approaches [ATY+19].

Deep Supervised Learning

Supervised learning is a learning technique that uses labeled data. In the case of supervised DL approaches, the environment has a set of inputs and corresponding outputs. There are different supervised learning approaches for deep learning, such as Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), including Long Short-Term Memory (LSTM) [HS97a], and Gated Recurrent Units (GRU) [CVMG+14]. These networks will be described in detail in the respective sections.

Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) or ConvNet [Sch15], is a popular choice of neural networks for different computer vision tasks such as image recognition [KSH12]. The name "convolution" is derived from a mathematical operation involving the convolution of different functions. CNNs consists of two main parts: feature extractors and a classifier. In the feature extraction layers, each layer of the network receives the output from its immediate previous layer as its input, and passes its output as the input to the next layer.

As shown in the example illustrated in Figure 2.3, there are two types of layers: convolution and max-pooling. The output nodes of the convolution and max-pooling layers are grouped into a 2D plane called "feature mapping". Each plane of a layer is usually derived from the combination of one or more planes of previous layers. The nodes of a plane are connected to a small region of each connected planes of the previous layer. Each node of the convolution layer extracts the features from the input by convolution operations on the input nodes.

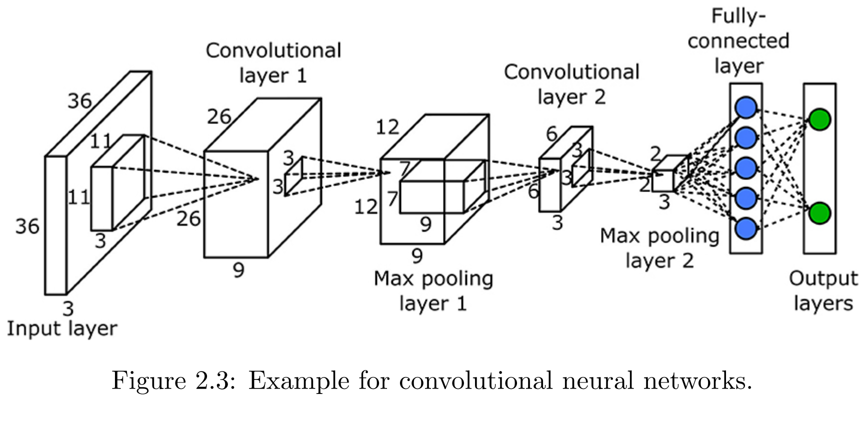


Figure 2.3: Example for convolutional neural networks.

Higher-level features are derived from features propagated from lower level layers. As the features propagate to the highest layer or level, the dimensions of features are reduced depending on the size of the kernel for convolutional and max-pooling operations respectively. The output of the last layer of the CNN is used as the input to a fully connected network which is called classification layer. In the classification layer, the extracted features are taken as inputs with respect to the dimension of the weight matrix of the final neural network.

Recurrent Neural Networks

Recurrent Neural Networks (RNNs) have been very popular in areas where the sequence in which the information is presented is important. As a result, they are used in many applications in real-world domains such as NLP, speech synthesis and machine translation. RNNs are called "recurrent" mainly because a uniform task is performed for every single element of a sequence, with the output dependent on the previous computations as well. These networks act as having a memory, where every calculated information is captured, stored and utilized to calculate the final outcome. Figure 2.4 illustrates a simple RNN framework.

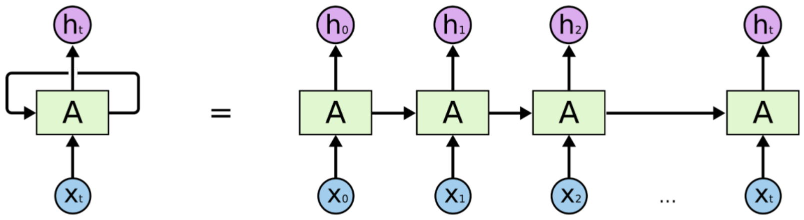


Figure 2.4: Recurrent neural networks architecture.

For the simple recurrent network, the recurrent kernel is implemented as:

$$h_t = \text{ReLU}\left(W_{ih} \cdot g_t + W_{hh} \cdot h_{t-1} + b_r\right) \tag{2.7}$$

where h_t and h_{t-1} are the hidden state of the RNN at step t and $t-1$ respectively, g_t is the glimpse feature extracted by the spatial glimpse network at step t, W_{ih} is the weighting matrix from input to hidden, W_{hh} is the weighting matrix from hidden to hidden, and b_r is the bias term for the recurrent module. Rectified linear unit (ReLU) is used as an activation function for the simple recurrent network.

RNNs have been used for many NLP applications such as: Word-level classification (e.g., Named Entity Recognition), Language modeling, Sentence-level classification (e.g., sentiment polarity), Semantic matching (e.g., match a message to candidate response in dialogue systems), Natural language generation (e.g., machine translation, visual QA, and image captioning). In the language modeling, it tries to predict the next word or set of words or some cases sentences based on the previous ones [ATY+19]. As it compares with a CNN model, an RNN model can be similarly effective or even better at specific natural language tasks but not necessarily superior [YKYS17]. This is because they model very different aspects of the data, which only makes them effective depending on the semantics required by the task at hand.

Over the years, quite a few varieties of RNNs have been researched and developed, such as bidirectional RNN. In bidirectional RNN, the output in this type of RNN depends not only on the past but also the future outcomes. Figure 2.5 shows an example of bidirectional RNN.

In NLP, the input expected by a RNN are typically one-hot encodings or word embeddings, but in some cases they are coupled with the abstract repre-

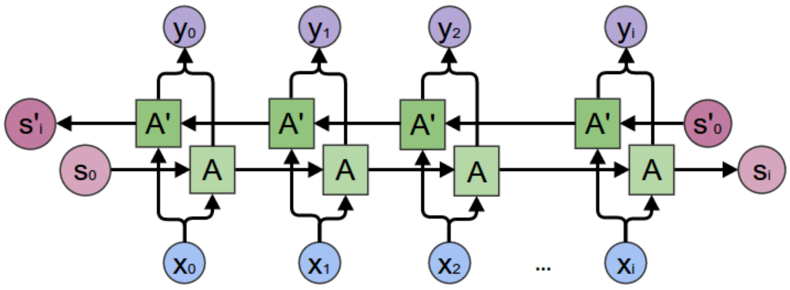


Figure 2.5: Bidirectional neural networks architecture.

sentations constructed by other deep learning model (e.g., a CNN model). Simple RNNs suffer from the *vanishing gradient problem* [HBF+01] which makes it difficult to learn and tune the parameters in the earlier layers. Other variants, such as long short-term memory (LSTM) networks, and gated-recurrent networks (GRU) were later introduced to overcome this limitation.

An LSTM consist of three gates (input, forget, and output gates), and calculate the hidden state through a combination of the three. GRUs are similar to LSTMs but consist of only two gates and are more efficient because they are less complex. Various LSTM-based models have been proposed for sequence to sequence mapping (via encoder-decoder frameworks) that are suitable for machine translation, text summarization, modeling human conversations, question answering, image-based language generation, among other tasks. LSTM and GRU are described in detail in the following sections.

Long Short-Term Memory (LSTM)

RNN has short-term memory, but with combining Long Short Term Memory (LSTM) gates, the network can have long term memory. Instead of the recurring section in RNN, LTSM is a small neural network consisting of four neural network layers. These are the recurring layer from the RNN with three networks acting as gates. An LSTM also has a cell state as well, alongside the hidden state. This cell state is the long-term memory. Rather than just returning the hidden state at each iteration, a tuple of hidden states is returned comprised of the cell state and hidden state.

LSTM has three gates: an Input gate, this controls the information input at each time step. An Output gate, this controls how much information is given in output to the next cell or upward layer. A Forget gate, this controls how much data to lose at each time step. It can be represented by the following equations:

$$\begin{aligned}
i_t &= \sigma\left(W_{xi} g_t + W_{hi} h_{t-1} + W_{ci} c_{t-1} + b_i\right) \\
f_t &= \sigma\left(W_{xf} g_t + W_{hf} h_{t-1} + W_{cf} c_{t-1} + b_f\right) \\
c_t &= f_t c_{t-1} + i_t \tanh\left(W_{xc} g_t + W_{hc} h_{t-1} + b_c\right) \\
o_t &= \sigma\left(W_{xo} g_t + W_{ho} h_{t-1} + W_{co} c_t + b_o\right) \\
h_t &= o_t \tanh\left(c_t\right)
\end{aligned} \tag{2.8}$$

where σ is the logistic function, and i, σ, o, and c are the four components inside the LSTM model, namely, input gate, forget gate, output gate, and cell. σ is the $softmax$ activation function. b is the bias term. The subscripts of weighting matrices W also indicate the input to the matrix as well as the component the matrix belongs to. For example, W_{xi} belongs to the input gate and takes glimpse feature g_t as input, while W_{ho} belongs to the output gate and takes previous hidden states h_{t-1} as input.

Gated Recurrent Unit (GRU)

Gated Recurrent Unit (GRU) is sometimes referred to as a gated recurrent network [CVMG$^+$14]. At the output of each iteration there is a small neural network with three neural networks layers implemented, consisting of the recurring layer from the RNN, a reset gate and an update gate. The update gate acts as a forget and input gate. The coupling of these two gates performs a similar function as the three gates forget, input and output in an LSTM. Compared to an LSTM, a GRU has a merged cell state and hidden state, whereas in an LSTM these are separate.

The reset gate takes the input activations from last layer, these are multiplied by a reset factor between 0 and 1. The reset factor is calculated by a neural network with no hidden layer (like a logistic regression), this performs a dot product matrix multiplication between a weight matrix and the concatenation of the previous hidden state and our new input. This is then put through the *sigmoid* function. This can learn to do different things in different situations, for example, to forget more information if there is a full stop token.

The update gate controls how much of the new input to take and how much of the hidden state to take. It is a linear interpolation. This is $1 - Z$ multiplied by the previous hidden state plus Z multiplied by the new hidden state. It controls to what degree we keep information from the previous states and to what degree we use information from the new state. The update gate is often represented as a switch in diagrams, although the gate can be in any position to create a linear interpolation between the two hidden states. Figure 2.6 shows the difference between LSTM and GRU architectures.

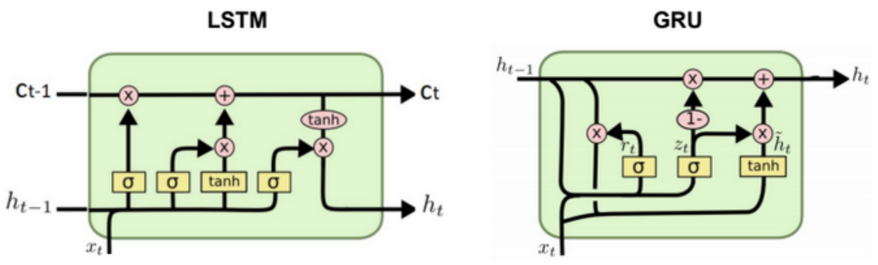


Figure 2.6: LSTM vs. GRU architectures.

Recursive Neural Networks

Similar to RNNs, recursive neural networks are natural mechanisms to model sequential data. Language could be seen as a recursive structure where words and sub-phrases compose other higher-level phrases in a hierarchy [FGS98]. In such a structure, a non-terminal node is represented by the representation of all its children nodes. Figure 2.7 shows an example for a recursive neural network. Recursive neural networks are used for various applications such as parsing, leveraging phrase-level representations, semantic relationships classification and sentence relatedness [LYLZ14].

Deep Unsupervised Learning

Unsupervised learning systems work without the presence of data labels. In this case, the network learns the internal representation or important features to discover unknown relationships or structure within the input data. There are several members of the deep learning family that are good at clustering and non-linear dimensionality reduction, including Auto Encoders (AE), Restricted Boltzmann Machines (RBM), and the recently developed Generative

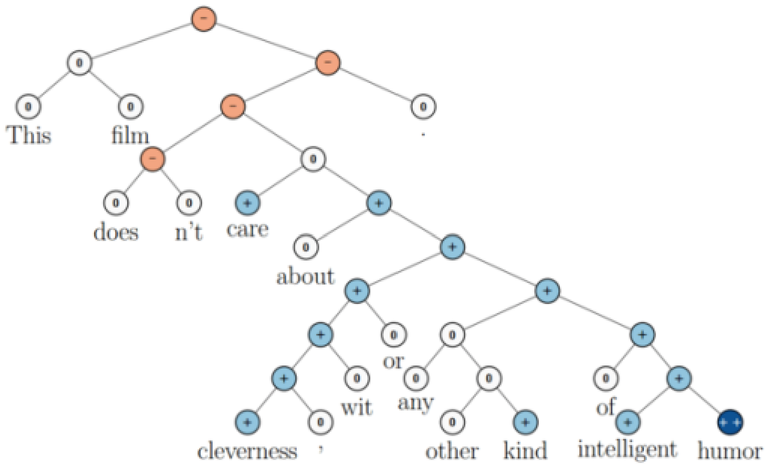


Figure 2.7: Example for recursive neural networks (from [SPW+13]).

Adversarial Networks (GANs), which are described in the following sections. In addition, RNNs, such as LSTM are also used for unsupervised learning in many application domains.

Autoencoder (AE)

Autoencoder (AE) is a deep neural network approach used for unsupervised feature learning with efficient data encoding and decoding. The main objective of autoencoder is to learn and represent (encoding) of the input data, typically for data dimensionality reduction, compression, and many more [BCV13]. Autoencoder technique consists of two parts: the encoder and decoder. In the encoding phase, the input samples are mapped usually in the lower dimensional features space with a constructive feature representation. This approach can be repeated until the desired feature dimensional space is reached. Whereas in the decoding phase, we regenerate actual features from lower dimensional features with reverse processing. The conceptual diagram of autoencoder with encoding and decoding phases is shown in Figure 2.8.

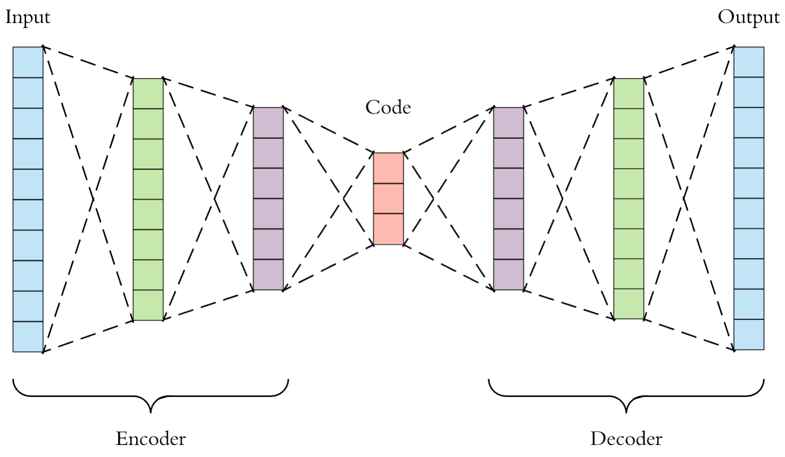


Figure 2.8: Autoencoder architecture.

Deep Belief Networks (DBNs)

Deep Belief Networks (DBNs) are composed of layers of Restricted Boltzmann Machines (RBMs) for the pre-train phase and then a feedforward network for the fine-tune phase [Hin09]. Figure 2.9 shows the network architecture of a DBN. The fundamental purpose of RBMs in the context of deep learning and DBNs is to learn these higher-level features of a dataset in an unsupervised training fashion.

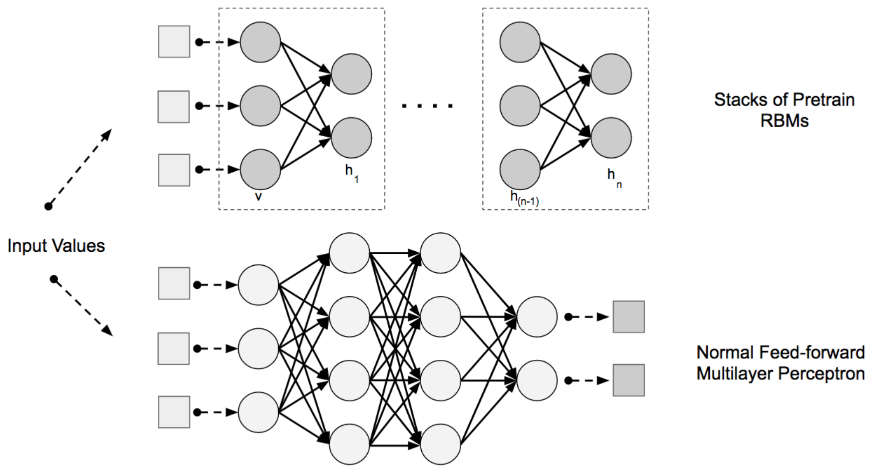


Figure 2.9: DBN architecture.

Generative Adversarial Networks (GANs)

Generative Adversarial Network (GAN) is a deep learning approach proposed by [GPAM+14] in 2014. GANs offer an alternative approach to maximum likelihood estimation. GAN is an unsupervised deep learning approach, it consists of two competing networks – a generator (G) and a discriminator (D). G generates synthetic data from some noise with the goal of fooling D into thinking it is real data. D has to discriminate whether a given sample is real or fake. It is this simple tussle between these two networks that makes GANs so powerful.

Deep Semi-Supervised Learning

Semi-supervised learning is learning that occurs based on partially labeled datasets [AZ05]. In some cases, Deep Reinforcement Learning (DRL) and Generative Adversarial Networks (GAN) are used as semi-supervised learning techniques. Additionally, RNNs, including LSTMs and GRUs, are used for semi-supervised learning as well [DL15].

2.3 Word Embeddings

Word embeddings are vector representation of the meaning of words. In practice, this usually means that word embeddings are placed in a high dimensional space where the embeddings of similar or related words are close to each other, while different word embeddings are placed far from each other. Word embeddings also acquire more complex geometric structures as a side effect of some algorithms. A typical example for this are real world analogies that can be discovered using simple vector arithmetic: $king - man + woman = queen$. Figure 2.10 shows an example of words in high dimensional space.

Word embeddings can be trained on knowledge graphs, but the most popular algorithms learn these vector representations just by scanning big corpora. All of these algorithms rely on a single assumption: words that appear in similar contexts have similar meanings. There are many state-of-the-art word embedding techniques, but the most popular are Word2vec [MSC+13], GloVe [PSM14a] and FastText [BGJM17]. Each of these techniques are described in detail in the following sections.

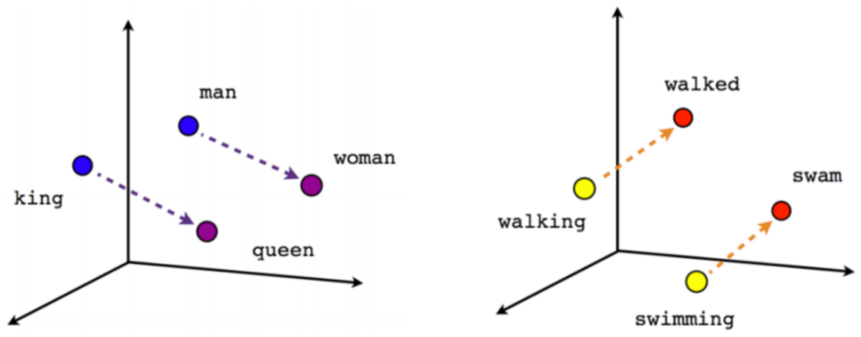


Figure 2.10: An example for word embeddings spaces.

Word2Vec

Word2vec is a two-layer neural net that processes text. Its input is a text corpus and its output is a set of vectors (i.e., feature vectors for words in the corpus). Word2vec starts with a set of word vectors that are initialized randomly. It scans the corpus sequentially, always keeping a context window around each word it looks at. The algorithm computes the dot product between the target word and the context words and tries to minimize this metric performing Stochastic Gradient Descent (SGD) [BB08]. The more evidence is found while scanning the corpus that two words are similar, the closer they will be.

Word2vec has two model architectures to produce a distributed representation of words: continuous bag-of-words (CBOW) or skip-gram. As shown in Figure 2.11, the CBOW architecture, the model predicts the current word from a window of surrounding context words. The order of context words does not influence prediction. While in the skip-gram architecture, the model uses the current word to predict the surrounding window of context words.

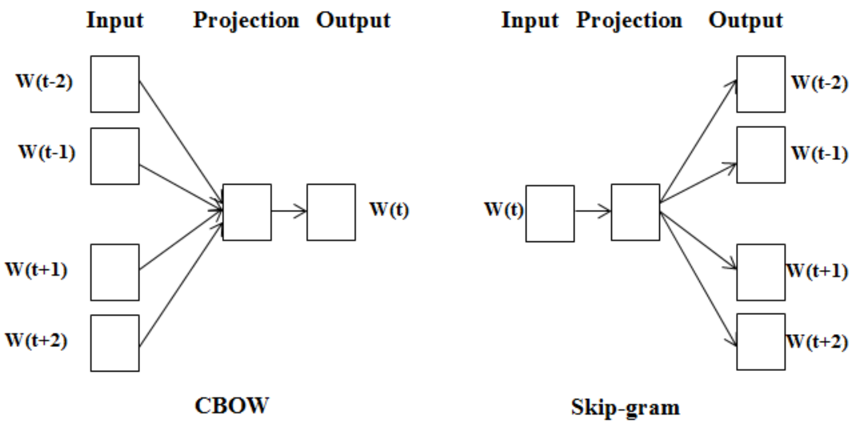


Figure 2.11: CBOW vs. Skip-gram architectures.

GloVe

GloVe is the second most well-known word embedding algorithm, after word2vec. They both perform similarly on most tasks, although the popular perception seems to be that GloVe is marginally faster to train. GloVe is based on matrix factorization (MF) techniques on the word-context matrix. It first constructs a large matrix of (words x context) co-occurrence information, i.e., for each word (the rows), it counts how frequently this word is seen in some context (the columns) in a large corpus. Then it factorizes this matrix to yield a lower-dimensional (word x features) matrix, where each row now yields a vector representation for each word. In general, this is done by minimizing a reconstruction loss. This loss tries to find the lower-dimensional representations which can explain most of the variance in the high-dimensional data.

FastText

FastText is another word embedding method that is an extension of the word2vec model. Instead of learning vectors for words directly, fastText represents each word as an n-gram of characters. This helps capture the meaning of shorter words and allows the embeddings to understand suffixes and prefixes. Once the word has been represented using character n-grams, a skip-gram model is trained to learn the embeddings.

So, for example, take the word, "science" with n = 3, the fastText representation of this word is <sc, sci, cie, ien, enc, nce, ce>, where the brackets indicate the beginning and end of the word. This model is considered to be a BOW model with a sliding window over a word because no internal structure of the word is taken into account. As long as the characters are within this window, the order of the n-grams does not matter.

FastText works well with rare words. So even if a word was not seen during training, it can be broken down into n-grams to get its embeddings. Word2vec and GloVe both fail to provide any vector representation for words that are not in the model dictionary. This is a huge advantage of this method.

2.4 Sentence Embeddings

Individual neural network layers require fixed-length inputs. Thus, if a neural network is to handle sentences of varying lengths, it needs to be accompanied by a component for converting these sentences into suitable fixed-length vectors. A simple such technique is to look up the embedding vectors for each of the words in a sequence and sum or average them, yielding a single vector of the same dimension. This strategy, often called *centroid of word embeddings* is effective in some simple tasks, but it ignores word order information which may be important in other tasks.

On the other hand, most modern neural network models that operate over word sequences include a special learned neural network component called *a sentence encoder*. This component has repeating parts that can be added or removed to fit the structure and size of the input sentence, and generally takes one of three basic forms:

- In recurrent or sequence-based networks, including LSTMs and GRUs, the input is fed into the network in sequential order (from left to right or vice versa for text) with the network updating a hidden state after each input is processed.

- In convolutional neural networks, information from all parts of a sentence are processed in parallel using a set of filters that look at fixed-size subsequences of words [KGB14].

- In recursive or tree-structured networks, the input is fed in according to a tree structure (generally produced by a parser), in which hidden representations are formed for increasingly large sub-spans of the input data following the principle of compositionality [SLMN11].

Of these, sequence-based models have been the most widely used in practice. While convolutional models have not been as widely used for sentence embeddings.

2.5 Attention Mechanism

The basic idea of attention mechanism is that when a sequence deep learning model predicts an output word, it only uses parts of an input where the most relevant information is concentrated instead of an entire sentence. In other words, it only pays attention to some input words [LPM15].

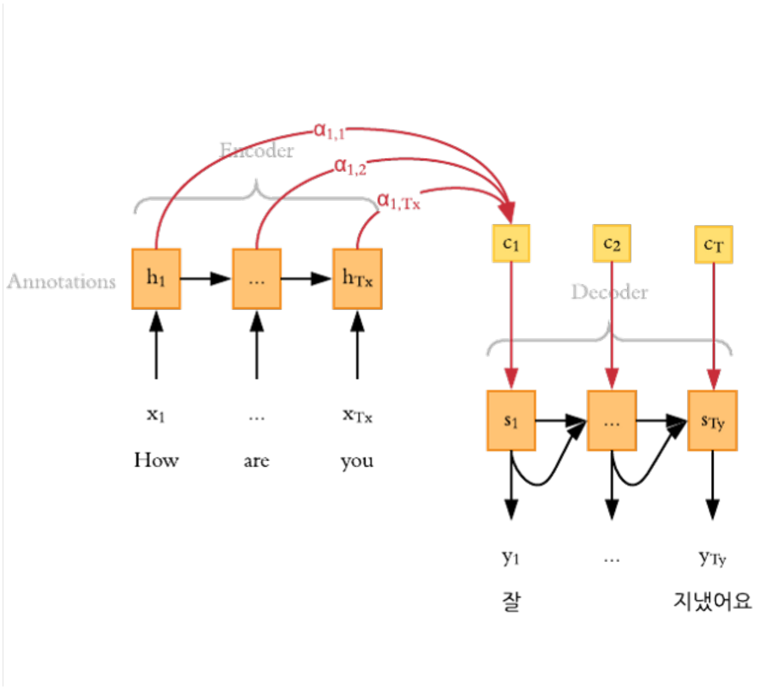


Figure 2.12: An example illustrating the attention mechanism.

As an example, in the encoder-decoder architecture, as shown in Figure 2.12, the decoder's hidden state is computed with a context vector, the previous output and the previous hidden state. We do not use a single context vector c, but a separate context vector c_i for each target word. These context vectors are computed as a weighted sum of annotations generated by the encoder. The weight of each annotation is computed by an alignment model which scores how well the inputs and the output match. An alignment model is a feedforward neural network, for instance. In general, it can be any other model as well. As a result, the alphas — the weights of hidden states when computing a context vector — show how important a given annotation is in deciding the next state and generating the output word. These are the attention scores.

Chapter 3

Research Paper Recommender Systems — A Survey

Recommender system (RS) approaches can be classified by their method of recommendation into three types: content-based filtering, collaborative filtering and hybrid approaches. Content-based filtering mainly uses the content of items which are highly rated by a user in order to find her preferences. On the other hand, collaborative filtering utilizes the similarity between user's preferences and other similar users' preferences in order to recommend new items. While hybrid RSs use a combination of content-based and collaborative filtering techniques in order to get the best of both. The authors of [BGLB16] have published a comprehensive survey about the research paper RSs proposed from 1998 to 2013. In this chapter we aim to extend part of this work by including a review of the recent work done in this area, highlighting the techniques used, especially the content-based and deep learning related ones. The use of deep learning models for NLP has recently received much attention, since they are usually trained on large amounts of data and they could provide high quality semantic representations. Finally, we discuss the challenges of the different recommendation approaches in the research paper domain.

3.1 Content-based Filtering

Content-based filtering is widely used in recommender systems. This approach provides recommendations by comparing candidate item's content representation with the target user's interest representation. This method has been applied mostly in textual domains such as news recommendation [PB07]. There have been some attempts to develop recommendation systems for scientific literature, and some work was done based on articles content.

In [YM07], the authors presented *PURE*, a content-based RS based on document titles and abstracts of the PubMed[1] dataset. It used the well-known TF-IDF method to train a probabilistic model for computing relevant documents based on selected documents added by the user. The idea of *PURE* was to automatically capture user preferences by using her response to the presented papers.

In [NLdSG11], the authors provided another example of a content-based filtering technique for scientific articles recommendation. Their proposed solution utilized n-grams models to generate queries from a particular article that is presented by the user, and then submit the generated queries on publicly available web sources of scientific papers. Their method used titles and abstracts of the articles, and the similarity of the papers was calculated through term frequency weighting scheme and cosine similarity method. The results gained by

[1]https://www.ncbi.nlm.nih.gov/pubmed/

this technique was fairly positive, demonstrating that it is enough to consider only the title and abstract of the articles for recommendation purposes.

In [FPT11], the authors proposed a content-based research paper RS which produces rich user profiles and resource descriptions by extracting keyphrases from scientific articles, mainly using part-of-speech (POS) tags and n-gram features.

Docear [BLGN13] is an academic literature suite to search, organize, and create research articles. Its recommender system uses content-based methods to recommend articles. It builds a user model using the mind maps created by the user, and matches it with *Docear* Digital Library in order to generate recommendations.

In [LLK13], a personalized academic research paper RS is presented. It recommends articles relevant to the research held by the users, supposing that researchers like their own articles. Based on this assumption papers similar to the ones previously written by users are recommended as relevant to them. This system used a web crawler to retrieve research papers from IEEE Xplore[2] and ACM digital library[3]. It measures text similarity using bag-of-words (BOW) and k-nearest neighbors (KNN) methods to determine the similarity between two research papers, and it uses collaborative filtering methods to recommend the items.

In [HBH+15], the authors proposed a solution to recommend scientific articles to non-profiled users. This methodology is meant to avoid the problems of collaborative filtering for users for whom there is not enough data available to build their user profile. They take a content-based approach in extracting

[2]https://ieeexplore.ieee.org
[3]https://dl.acm.org/

both short and long queries from a single paper provided as an input. The long queries are taken from the abstract and sections similar to the title whereas the short queries are commonly occurring phrases in the paper as well as words from the title. These queries are weighted and used to filter candidate papers from the corpus. The recommendations are made using a simple cosine similarity between the target paper and the filtered papers.

In [AARK16], the authors presented *Science Concierge*, an open source Python library that implements a RS for literature search. The library uses a scalable vectorization of documents through online Latent Semantic Analysis (LSA) [LD97] to discover groups of words that are equivalent in their meaning.

In [APSF16], the authors proposed a fully content-based approach to the recommendation of scientific papers based on the researchers corpus. The researcher profile is built upon the topics generated by LDA algorithm on the researcher's publications corpus.

In [DNT14], the authors proposed a novel content-based RS technique based on a network, rather than vector. User models were built upon sets of concepts automatically extracted from documents. By using concepts as features, they developed a concept-based RS that suggests the papers related to the concepts of interest for the active user. More specifically, concepts are identified as keyphrases automatically extracted from scientific papers.

The authors in [AAUE17] presented a recommender system for research papers which used a Dynamic Normalized Tree of Concepts (DNTC) user modelling technique. The system utilized the ontology of 2012 ACM Computing Classification System (CCS)[4]. The user profiling phase creates a user profile as a dynamic normalized tree of concepts, which is used with a dynamic tree

[4]https://www.acm.org/publications/class-2012

edit distance method, to compare between the user profile and the new unseen research papers, that are also represented as tree of concepts.

The authors of [BABG17], introduced *Mr. DLib*, a recommendations-as-a-service, which allows third parties to easily integrate a RS into their products. *Mr. DLib* indexes the metadata of the partner's documents (title, authors, abstract, venue, keywords), and it uses mainly Apache Lucene/Solr's More-Like-This function to calculate document relatedness.

3.2 Collaborative Filtering

Collaborative filtering is one of the most successful recommendation approaches that works by recommending items to target users based on what other similar users have previously preferred. This method has been used in e-commerce sites such as Amazon.com, Ebay and so on. However, it suffers from the *cold-start problem* in which it cannot generate accurate recommendations without enough initial ratings from users [HKTR04].

In the research paper domain, some authors suggested using collaborative filtering and ratings. Ratings can be generated by considering citations as ratings. They can also be implicitly inferred by monitoring user's actions such as downloading or bookmarking a paper. Citation databases such as CiteSeerX[5] apply citation analysis in order to identify papers that are similar to an input paper. Scholarly search engines such as Google Scholar[6] focus on classic text mining and citation counts. The research of [WSW16] has also presented a RS based on citations.

[5]https://citeseerx.ist.psu.edu
[6]https://scholar.google.com/

The authors of [SK13] applied collaborative filtering to discover potential citation papers that help model target papers to recommend. The proposed technique significantly outperformed state-of-the-art recommendation baselines as measured by Normalized Discounted Cumulative Gain (nDCG) and Mean Reciprocal Rank (MRR).

In [HID$^+$17], the authors utilized the publicly available contextual metadata to leverage the advantages of collaborative filtering approach in recommending a set of related papers to a researcher based on paper-citation relations. The approach mined the hidden associations between a research paper and its references and citations using paper-citation relations. The rationale behind the approach is that, if two papers are significantly co-occurring with the same citing paper(s), then they should be similar to some extent.

3.3 Hybrid Approaches

In order to improve the performance of the RSs, there have been successful efforts to combine the collaborative filtering approaches with content-based filtering approaches [DCFLHRM10]. In the research paper domain, some work was done based on hybrid approaches.

The system proposed in [GBH09], called *Scienstein*, combines different methods for providing literature recommendation. *Scienstein* integrated the traditional keyword-based search with citation analysis, author analysis, source analysis, implicit and explicit ratings. Instead of entering just keywords for searching documents, a user may provide entire documents as an input, include reference lists, and provides implicit and explicit ratings in order to improve the recommendation process.

In [WB11] the authors have proposed an extension of LDA for recommending scientific articles called collaborative topic regression (CTR). This hybrid approach combines collaborative filtering based on latent factor models and content analysis based on topic models. A matrix factorization and LDA are merged into a single generative process, where item latent factors are obtained by adding an offset latent variable to the document-topic distribution. Like collaborative filtering approaches, this method is able to predict articles already rated by similar users.

The authors of [PN11] proposed *PubRec*, that make the recommendation based on the author's research interests using content-based, collaborative and global relevance approaches. The proposed approach recommends a research paper considering the paper similarity, author rating score and the number of times an article stored in their personal libraries. The content similarity relies on a word-correlation matrix [KN06] to determine the similarity between any two tags assigned to their respective publications, which capture and represent their contents.

The work of [MLA14] has addressed the issue of overcoming cold-starts in the collaborative filtering approach by first categorizing each document in the corpus using LDA as the topic model. A scraper and parser collected all the documents publicly available and categorized them into various fields. For a new user, a common stereotype set of recommendations were made. The prototype then logged user actions in order to build a user profile over time. This user profile was used in collaborative filtering to create new recommendations for the user.

The research of [MKKG16] proposed a novel method for integrating structural and contextual information to build a context specific network for generating recommendations for similar PubMed articles. The proposed method integrates graph-based models, statistical techniques, and NLP-based approaches that utilized word2vec for computing individual word similarities.

The authors of [WWY15] proposed collaborative deep learning (CDL) that integrates stacked denoising autoencoder (SDAE) [VLL$^+$10] into probabilistic matrix factorization (PMF), generating more accurate latent model in terms of the rating prediction accuracy.

Finally, the authors of [BBM16] have proposed a method leveraging deep recurrent neural networks to encode the text sequence into a latent vector, specifically gated recurrent units (GRUs) trained end-to-end on the collaborative filtering task. For the task of scientific paper recommendation, this yields models with significantly higher accuracy. Performance is further improved by multi-task learning, where the text encoder network is trained for a combination of content recommendation and item metadata prediction.

3.4 Challenges

Some of the proposed RS methods have drawbacks, which limit their ability to deliver effective recommendations. For example, collaborative filtering in the research paper RS domain would be ineffective as there is a huge number of papers compared with the number of users, and only few users rated the same papers. In domains such as movie recommendations, there are few items and many users such as in MovieLens[7] RS, and most movies have been watched and rated by at least some users. Therefore, like-minded users can be found and recommendations can be given effectively.

[7]https://movielens.org/

In the citation-based approaches, not all research papers are cited and, hence, cannot be recommended. Also, reference lists can contain irrelevant citations just because the author believes that well-known papers should be cited, or in order to promote other publications although they are irrelevant for the citing paper.

Thus, we believe that content-based filtering should be part of any research paper RS. However, most of the current content-based research paper RSs cannot identify related papers if different terms are used; they are based on traditional BOW models, that represent the number of times each word occurs in a document. The semantic similarity between words is not considered. In addition, these techniques do not take the context of the words into consideration.

3.5 Summary

In this chapter, we have provided a literature survey of research paper RSs, which can be categorized to content-based filtering, collaborative filtering and hybrid approaches. In addition, we have discussed the main challenges of those approaches. From which, it has been found that current methodologies for content-based filtering have limited capabilities that need to be addressed for further advancements in research paper RSs. Those difficulties will be investigated in the upcoming chapters by proposing deep learning models that aim to learn the semantics of papers.

Chapter 4

Semantic-based Tag Prediction for Research Papers

Recently, tagging has become a common way for users to organize and share digital content. Thus, tag prediction has become a very important research topic. Most of the prediction approaches based on text embedding have utilized bag-of-words (BOW) techniques. On the other hand, proposed deep learning methods for capturing semantic meanings in text, have been proved to be effective in various natural language processing (NLP) applications. In this work, we present a tag prediction method that adopts deep recurrent neural networks to encode titles and abstracts of scientific articles into semantic vectors, more specifically a hierarchical bidirectional gated recurrent units (bi-GRUs) with attention mechanism. The experimental evaluation is performed on a real dataset from CiteULike. The overall findings show that the proposed model is effective in extracting research paper semantic features for the tag prediction task.

4.1 Introduction

Tagging has become a common service on Web 2.0 applications. This kind of service allows users to share and annotate interesting web resources. In the scientific community, tagging allows users to share research papers based on their interest. It also allows the users to create annotations or tags attached to the research papers. By means of tags, users can conceptually organize and summarize information, but also search for it. In most cases, tags emphasize important keywords that may, or may not, be present in the document.

In many domains, text items are often associated with tags, such as microposts, news articles and research papers. Tag prediction in such systems help users find appropriate tags for resources and help consolidate annotations across all users and resources. Tag prediction not only improves user experience but also enriches the quality of generated tags and improves the quality of the information retrieval (IR) services and recommender systems (RSs) that rely on tags as source of information.

Existing approaches to automatic tag recommendation range from collaborative filtering (CF) [IHJS07] to traditional content-based techniques including BOW features (e.g., TF-IDF feature) or Naïve Bayes and unsupervised learning methods (e.g., topic models [SZG11]). Most of these methods rely on counting statistics that ignore specific features, such as keyword order. Therefore, they suffer from sparsity and poor generalization performance, and cannot effectively encode semantic information.

On the other hand, neural models have recently shown great potential for learning effective representations and delivered state-of-the-art performance on various NLP tasks. For example, semantic sentence embedding methods effectively map text content into vector spaces, obtaining relevant performances in sentence classification tasks. Among those methods, the long short-term memory (LSTM) and Gated Recurrent Units (GRUs), variant of recurrent neural networks (RNNs), are widely used due to their capability of capturing long-term dependencies in learning sequential representations.

In this work, we propose an approach that adopts bidirectional gated recurrent units (bi-GRUs) with attention mechanism to capture important patterns and semantic representations of summaries of scientific papers (i.e., titles and abstracts) for the tag prediction task. We consider the tag prediction as a multi-label classification problem. Additionally, we compare the proposed method with several baselines commonly used in similar works. The experimental results on a real dataset extracted from CiteULike[1], an online scientific articles bookmarking system, prove the validity of our approach.

The main contributions of this work can be summarized as follows:

- A proposed approach that takes advantage of deep neural network architectures and bi-GRUs with attention mechanism applied on paper titles and abstracts for the tag prediction task.
- Experimental evaluation on a dataset collected from CiteULike service to comparatively prove the benefits of the approach in terms of prediction accuracy with respect to some state-of-the-art methods.

4.2 Related Work

In this section, we briefly review some related work on multi-label text classification, and then we review some of the work that utilized attention mechanism

[1]http://www.citeulike.org/

for textual feature extraction.

Multi-label Text Classification

The most popular learning method for multi-label text classification is the vector space model (VSM), which represents each document through a vector of all word occurrences weighted by their TF-IDF, or statistical topic modeling techniques, such as Latent Dirichlet Allocation (LDA) [BAG17, KF09, KFN09, HKL17]. However, those methods do not utilize information such as text order and semantic of words.

There are some work that has utilized deep learning techniques for sentence representation and text-based multi-label classification. For example, in [NKM$^+$14] the authors showed that a simple neural networks architecture with deep learning techniques such as rectified linear units (ReLUs), learning rate adaptation, and regularization using dropout training, can outperform the traditional multi-label classification techniques. They experimented on different datasets including scientific articles dataset.

In [Ber15], the author has demonstrated that both a CNN and a GRU using semantic word embeddings significantly outperform the Binary Relevance method with BOW features on a large scale multi-label classification problem. The models have been tested on scientific abstracts from PubMed[2]. The authors of [LCWY17] have utilized CNNs [Kim14] for extremely large label collection. [GSC18] also applied deep CNN on research papers with abstracts and titles for each article for multi-label text classification of PubMed MeSH (Medical Subject Heading).

[2]www.ncbi.nlm.nih.gov/pubmed

Attention Mechanism

Attention-based models have demonstrated success in a wide range of NLP tasks. Attention is a mechanism for selecting the reference part of context information, which can facilitate global learning. It was originally proposed in machine translation tasks to deal with the issue for encoder-decoder approaches that all the necessary information should be compressed into fixed length encoding vector [BCB14]. Then, an attention model is leveraged for generating image descriptions [XBK+15]. Other attention-based work includes sentence summarization [RCW15].

In [PTDU16], the authors proposed an attention model based on a convolutional neural network (CNN) for natural language inference. In [WCdML16], the authors presented an approach for relation classification via multilevel attention CNNs. In [YSXZ15], the authors proposed an attention-based CNN for modeling sentence pairs. Yang *et al.* [YYD+16] exploited attention in neural networks, enabling it to attend differentially to more and less important content when constructing the document representation, by capturing hierarchical patterns of documents from word to sentence and finally to the whole document.

To the best of our knowledge, this is the first study utilizing hierarchical bidirectional gated recurrent units (bi-GRUs) with attention mechanism to represent research papers (i.e., their titles and abstracts) for a tag prediction task.

4.3 Methodology

In this work, we model the tag prediction task as a multi-label classification problem, where document level classification aims to predict the tag distributions according to their text information. We use concatenated titles and abstracts of papers to represent documents.

Problem Definition

Given a set of N documents $\{d_i\}$ and a vocabulary of M tags $\{t_j\}$, we consider a dataset $D = \{(x_i, y_j)\}$, where x_i is the keyword-based representation of the document d_i, and y_j assumes 1 or 0 value if the tag t_j is associated with d_i or not, respectively. The representation x_i consists of a sequence of d-dimensional embeddings of consecutive words grouped into sentences, as follows:

$$x_i = \{\{w_{1,1}, w_{1,2}, ..., w_{1,N_T}\}, \cdots, \{w_{N_K,1}, w_{N_K,2}, ..., w_{N_K,N_T}\}\} \tag{4.1}$$

being N_T the maximum number of words in a sentence, and N_K the maximum number of sentences in a document. The network takes as input a document x_i and outputs a document vector u_i. The output u_i is used by the classification layer to determine $\{y_j\}$, that is, the tags related to d_i.

The Proposed Approach

To solve the aforementioned problem, we adopt a hierarchical attention architecture for document representation, as shown in Figure 4.1. The proposed approach consists of the following components:

- **Input layer.** It takes the concatenated titles and abstracts of research papers.
- **Word embeddings.** Each document is assigned to a word sequence representation, where each position corresponds to a word vector from pre-trained word embeddings.
- **Word sequence encoder.** It encodes high-level representation of the sequence of words using bi-GRUs.
- **Word attention.** It applies the attention mechanism to get the important words from a sentence.
- **Sentence sequence encoder.** It encodes high-level representation of the sequences of sentences using bi-GRU.
- **Sentence attention.** It incorporates the attention model into the network and derive the final document representation.
- **Output layer.** The representations of the documents are concatenated, with the *softmax* function which gives in output the predicted tags.

The detailed descriptions of all the different components are given in the following sections.

Word Embeddings

Titles and abstracts of the scholarly documents are extracted, concatenated, and subjected to tokenization and stop-word removal. Each word in the document is represented as a fixed-size vector from pre-trained word embeddings. We used GloVe [PSM14b] word embeddings for this purpose. More details

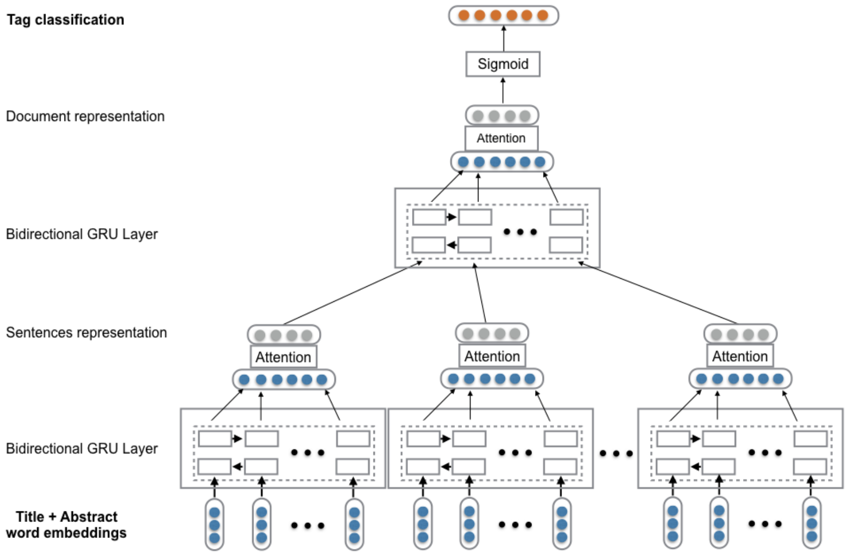


Figure 4.1: Proposed approach for tag prediction.

about the selection of the word embeddings model is given in the "Experimental Settings" Section.

Bidirectional GRU Encoders

As stated in Chapter 2, recurrent neural network (RNN) [RHW86] is a kind of feed-forward neural network that is useful for modeling sequences. Traditional RNN models suffer from the problem of vanishing and exploding gradients, preventing them from learning long-term dependencies. There have been several modifications to the RNNs proposed to overcome this problem, among which the most popular are long short-term memory units (LSTMs) [HS97b] and the more recent gated recurrent units (GRUs).

GRUs encoders are simpler than LSTMs, in terms of number of parameters, and give competitive performance. GRU uses reset and update gate vectors at each position to control the information flow along the sequence, thus improving the modeling of long-range dependencies. Bidirectional GRU [SP97] is another version of GRU. Unlike standard GRUs, which only capture information from the current and past states, bidirectional GRU determine outputs also considering inputs from the future.

At word level, we embed each word in a sentence into a low dimensional semantic space using a bi-GRUs. At sentence level, we also feed the sentence embeddings into bi-GRUs in order to obtain the document representation.

At word level, the function g_w encodes the sequence of input words $\{w_{l,t}|t = 1, ..., N_T\}$ for each sentence l of the document, that is:

$$h_w{}^{(l,t)} = \{g_w(w_{l,t})|t = 1, ..., N_T\} \tag{4.2}$$

At sentence level, after combining the intermediate word vectors $\{h_w{}^{(l,t)}|t = 1, ..., N_T\}$ to a sentence vector s_l, the g_s function encodes the sequence of sentence vectors $\{s_l|l = 1, ..., N_K\}$, that is, $h_s{}^{(l)}$.

The g_w and g_s functions are bidirectional GRUs with parameters H_w and H_s respectively, obtained from the forward GRU, $\overrightarrow{g_w}$, and the backward GRU, $\overleftarrow{g_w}$:

$$h_w{}^{(l,t)} = [\overrightarrow{g_w}(h_w^{(l,t)}); \overleftarrow{g_w}(h_w^{(l,t)})] \tag{4.3}$$

The same concatenation procedure is applied to the hidden state representation of a sentence $h_s{}^{(l)}$.

Attention Layers

A typical way of assigning a representation to a given word sequence at each level is by taking the last hidden-state vector output by the encoder. However, it is hard to encode all the relevant input information needed in a fixed-length vector, which may limit the performance of these networks, especially when long input sentences are considered. In addition, not all the input words contribute equally to the representation of the sentence meaning, and not all the sentences contribute equally to the document representation. This problem is addressed by introducing an attention mechanism at each level, denoted by α_w and α_s, that estimates the importance of each hidden state vector with respect to the sentence or document meaning, respectively. The sentence vector $s_l \in R^{dw}$, where dw is the dimension of the word encoder, is thus obtained as follows:

$$\sum_t \alpha_w^{(l,t)} h_w^{(l,t)} = \frac{\exp\left(u_{l,t}^\top u_w\right)}{\sum_t \exp\left(u_{l,t}^\top u_w\right)} h_w^{(l,t)} \tag{4.4}$$

where $v_{l,t} = f_w(h_w^{(l,t)})$ is a fully-connected neural network with W_w parameters. Similarly, the document vector $u \in R^{ds}$, where ds is the dimension of the sentence encoder, is obtained as follows:

$$\sum_l \alpha_s^{(l)} h_s^{(l)} = \frac{\exp\left(u_l^\top u_s\right)}{\sum_l \exp\left(u_l^\top u_s\right)} h_s^{(l)} \tag{4.5}$$

We feed the output vector to a linear layer whose output length is M, the cardinality of the tag vocabulary. We believe that such an attention model could adaptively assign an importance score to words according to their semantic relatedness with the tag.

Classification Layer

Finally, we formulate the tag prediction task as a multi-label classification problem. We train our model in a supervised manner by minimizing the cross-entropy error of the tag classification. We adopted a *sigmoid* classifiers of predefined classes on top for classification, with a loss based on the cross-entropy between gold and predicted tags, and the input is a combination of the features generated from the document level attention. The *sigmoid* layer is added to give in output the probability distributions of all candidate tags. The *sigmoid* function is calculated as follows:

$$\phi(z) = \frac{1}{1 + e^{-z}} \tag{4.6}$$

4.4 Experimental Evaluation

In this section, we give more details about the experimental settings and empirical results on the tag prediction task.

Dataset

CiteULike was an online platform which allowed registered users to create personal reference libraries by saving papers that are of interest to them [BVdB08]. The dataset is built upon this service and consists of papers in the user libraries, user provided tags on papers, and titles and abstracts of them. We used *citeulike-a* dataset from [WCL13], which consists of 16,980 papers with 46,391 tags. We selected the top-20 common tags from the dataset, and then we extracted the papers that were tagged with at least one of the top tags identified in the previous step. From this process, we got 8,386 titles and abstracts. Some statistics about the dataset after preprocessing are provided in Table 4.1. We then separated our data into 90% and 10% for training and testing, respectively. Therefore, we had 5747 and 839 documents in the training and test set, respectively. Figure 4.2 lists the most common 20 tags that we have selected for our experiments and their frequency.

Table 4.1: Statistics of a subset of *citeulike-a* dataset used for evaluating the performance of tag prediction (*Tag cardinality is the average number of tags assigned to a document).

#Documents	8,386
#Tags	20
Tag Cardinality*	4
Titles Vocabulary	11,638
Abstracts Vocabulary	61,363

Preprocessing

In the preprocessing step, titles and abstracts of each research paper were concatenated, tokenized, and subjected to stop-word removal before the lemmatization performed by the NLTK[3] Python package. The maximum number of sentences per document was set to 10, and the maximum number of words per sentence to 50 with zero-pad the beginning of sentences and documents, if necessary.

[3]https://www.nltk.org/

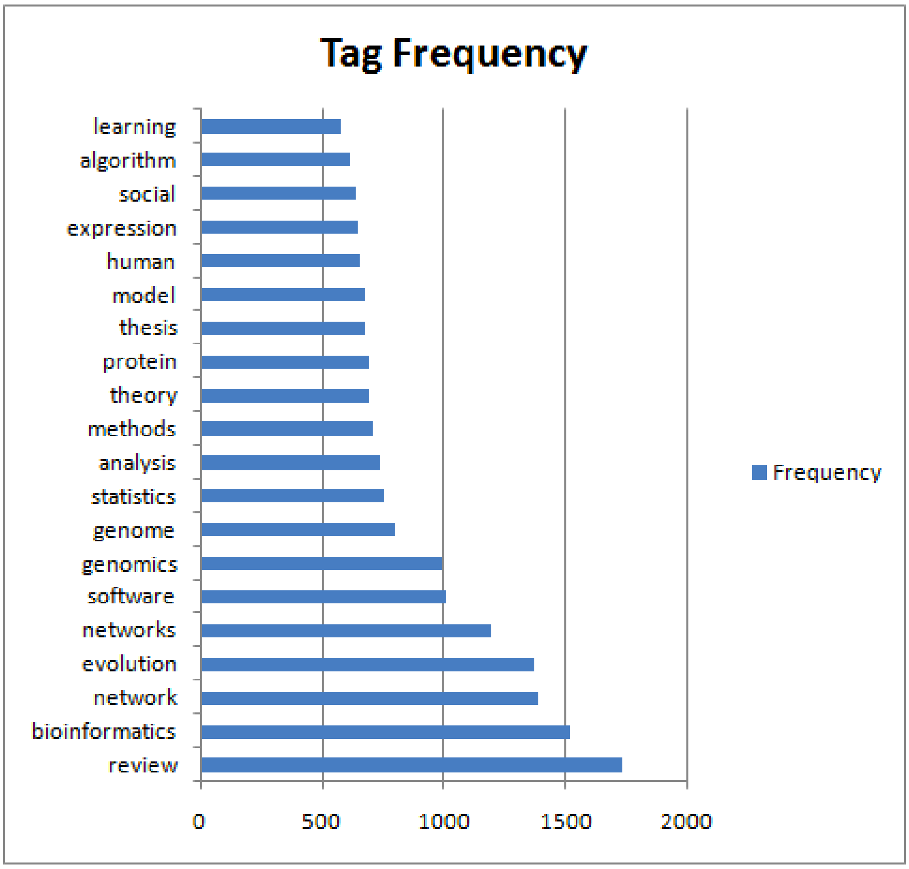


Figure 4.2: Top-20 tags from *citeulike-a* and their frequency.

Experimental Settings

In order to decide which word embedding model to use, we experimented with different embedding models in order to explore which of them has less out-of-vocabulary (OOV) words, i.e., words that do not exist in the vocabulary. If the occurrence of words that are actually important is not covered, this may reduce the F-measure performance of text categorization on scientific articles. Table 4.2 lists the different state-of-the-art word embedding models,

Table 4.2: Word embedding models and OOV statistics

Embedding Model	Vocab.	Titles OOV	Abstracts OOV
Word2Vec (Google News)[4]	3M	23.68%	51.92%
GloVe (Wikipedia 2014)[5]	400K	49.98%	55.82%
GloVe (Common Crawl - 42B tokens)[6]	1.9M	44.15%	45.11%
GloVe (Common Crawl - 840B tokens)[7]	2.2M	11.71%	34.20%
FastText (Wikipedia 2017)[8]	1M	49.63%	55.54%

the vocabulary size of the dataset used for training each model, the percent of the OOV words in both titles and abstracts of research papers in our dataset. As per these statistics, we decided to use the 840B tokens GloVe version that is pre-trained on Common Crawl as our word-level embeddings, since it has less OOV words in both titles and abstracts.

We implemented our model using the Python open source library Keras[9] with a TensorFlow[10] backend. Our method achieved the best performance when the dimension of hidden state of bi-GRU networks was set to 100. In this case, a combination of forward and backward GRU gave us 200 dimensions for word/sentence annotation. We added an additional dense layer after the

[9]https://keras.io/
[10]https://www.tensorflow.org/

document attention layer, in order to increase the complexity of the network. We also added dropout layers before and after the dense layer in order to prevent overfitting.

Additional hyperparameters were 50 hidden states for the dense layer and 0.2 as dropout value. A mini-batch stochastic gradient descent (SGD) algorithm was used to train each model [PVG$^+$11]. Batch size was set to 64 to minimize the loss function of a categorical cross entropy. Moreover, we manually set the prediction threshold to 0.5. Finally, the epoch was set to depend on an early stop, which relied on a validation set to decide when to stop the training. In our case, the early stop condition is when F1 value starts to decrease for three epochs. In our experiments, it took around 50 epochs to stop the training process.

Baselines

Multi-label text classification can generally be divided into two sub-tasks: text feature extraction and multi-label classification. In order to perform an empirical evaluation of our proposed method (i.e., Bi-GRU+Attention), we made a comparison with many baseline methods in text feature extraction. And we experimented with two different linear classification methods, namely Logistic Regression (LR) and Support Vector Machine (SVM) classifiers. These classifiers are suitable for high dimensional and sparse data (text data is high dimensional and sparse). For each classification method, we trained a set of classifiers, one for each label, using the OneVsRestClassifier available in the popular scikit-learn library [PVG$^+$11]. We experimented the classification methods when research paper titles only were used, also when titles and abstracts are concatenated.

The following is a detailed description for the classifiers we have used:

- **Support Vector Machine (SVM).** A widely used supervised text classifier [IHJS07]. SVMs have been explored systematically for text categorization [Joa98]. An SVM classifier finds a hyperplane that separates examples into two classes with maximal margin [CV95] (Multi-classes are handled by multi one-vs-rest classifiers). Examples that are not linearly separable in the feature space are mapped to a higher dimension using kernels. In our experiments, we used Linear SVC (Support Vector Classifier) implemented in scikit-learn library.

- **Logistic Regression (LR).** Logistic regression is also widely used for text classification [ZJYH03, GLM07, YHL11]. Logistic regression classifies data by using a decision boundary, determined by a linear function of the features. For the implementation of the algorithm, we used scikit-learn.

With both classifiers, we used different unsupervised text feature baselines as follows:

- **TF-IDF.** We used TF-IDF as baseline for representing text features. As mentioned in Chapter 2, TF-IDF is a BOW method based on the concept of term frequency.

- **Latent Dirichlet Allocation (LDA).** LDA is based on a Bayesian probabilistic model where each topic has a discrete probability distribution of words and each document is composed of a mixture of topics. In LDA, the topic distribution is assumed to have a Dirichlet prior which gives a smoother topic distribution per document. LDA has been utilized widely in tag recommendation [JWY+19]. Similar to [SZG11], we trained an N-topic LDA model [BNJ03], where N is the number of tags.

- **Paragraph-Vector.** Paragraph-Vector or Doc2Vec [LM14] is a state-of-the-art performer on several benchmark datasets. For complex text classification algorithms, the BOW would not be suitable as it lacks the capability to capture the semantics and syntactic order of words in the text. Thus using them as feature input to machine learning algorithm will not yield significant performance. On the other hand, Doc2Vec is able to detect the relationships among words and understands the semantics of the text. Doc2Vec is an unsupervised algorithm that learns fixed-length feature vectors for texts. The goal of Doc2Vec is to create a numeric representation of a document, regardless of its length. They have used the word2Vec model, and added another vector (Paragraph ID) as shown in Figure 4.3. So, when training the word vectors W, the document vector D is trained as well, and in the end of training, it holds a numeric representation of the document. Doc2Vec architecture also has

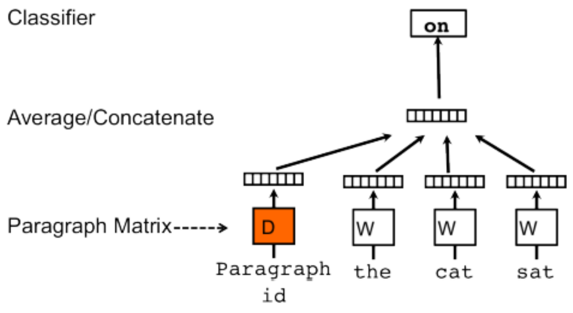


Figure 4.3: Doc2vec PV-DM model.

two algorithms like word2Vec and they are the corresponding algorithms for those two algorithms, namely, Continuous bag-of-words (CBOW) and Skip-Gram (SG). One of the algorithms in doc2vec is called Paragraph Vector - Distributed bag-of-words (PV-DBOW) which is similar to SG model in word2vec. Here the neural network is trained to predict the probability distribution of words in the given paragraph based on the words in the paragraph. The second algorithm as shown in Figure 4.3, which is Distributed Memory version of Paragraph Vector (PV-DM) that is similar to CBOW in word vector. It acts as a memory that remembers what is missing from the current context — or as the topic of the paragraph. While the word vectors represent the concept of a word, the document vector intends to represent the concept of a document.

- **Sent2Vec.** Sent2Vec [PGJ17] presents a simple but efficient unsupervised objective to train distributed representations of sentences. It can be thought of as an extension of fastText and word2vec (CBOW) to sentences. The sentence embedding is defined as the average of the source word embeddings of its constituent words. This model is furthermore augmented by also learning source embeddings for not only unigrams but also n-grams of words present in each sentence, and averaging the n-gram embeddings along with the words.

- **Centroid of Word Embeddings.** The centroid-based method for extractive summarization was introduced by [ER04]. The centroid represents a pseudo-document which encodes the meaningful information of a document. In the simplest case, the centroid of a text t is the sum of the embeddings of the tokens of t divided by the number of tokens in t. Previous work on hierarchical biomedical document classification [KAP15] reported improved performance when the IDF scores of the tokens are also taken into account. In this work, we used the 300-dimensional word embeddings obtained by applying GloVe word embeddings for calculating word embeddings centroid.

We also experimented with the following deep learning models:

- **CNN.** We used a word based CNN model similar to [Kim14]. In our experiments, the model is constructed from a total of 128 filters with size 5 and max pooling. Similar to our proposed model, we used a mini-batch size of 64 and an SGD to train the model.
- **Bi-GRUs.** We constructed a model with the same settings as our proposed approach, but without the attention layers.

Evaluation Methodology

To evaluate the performance of the proposed approach and the baselines, we report precision, recall, and F1, commonly used measures in information retrieval and essential in any classification task such as tag prediction, that involves imbalanced classes. The definitions of such evaluation metrics are as follows:

Precision (also called positive predictive value) is the fraction of relevant instances among the retrieved instances.

$$\text{Precision} = \frac{T_P}{T_P + F_P} \tag{4.7}$$

Recall (also known as sensitivity) is the fraction of relevant instances that have been retrieved over the total amount of relevant instances.

$$\text{Recall} = \frac{T_P}{T_P + F_N} \tag{4.8}$$

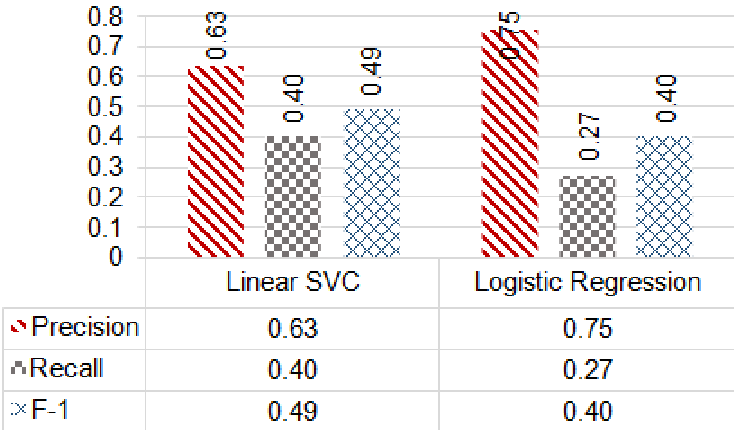
F1 is defined as the harmonic mean of precision and recall for all of the tags.

$$\text{F1} = \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.9}$$

where T_P is the number of true positive labels, F_P the number of false positive labels, and F_N the number of false negative labels.

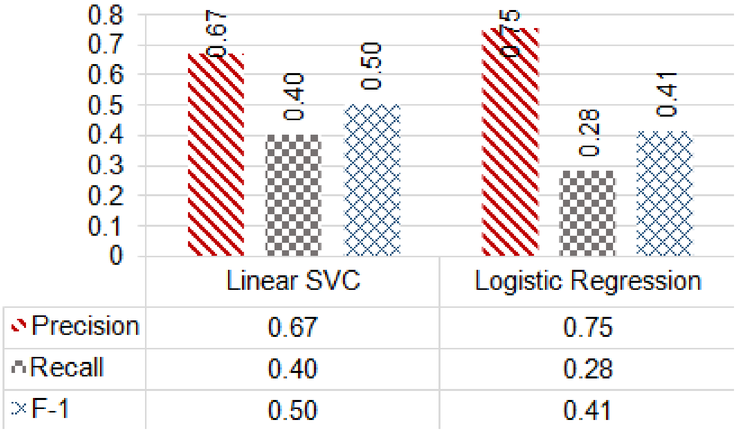
4.5 Results and Discussion

Figures 4.4, 4.6, 4.8, 4.10, and 4.12 show the results of TF-IDF, LDA, Doc2Vec, Sent2Vec and GloVe centroid when paper titles were used. While, Figures 4.5, 4.7, 4.9, 4.11, and 4.13 show the results when titles and abstracts were combined. SVM achieved the best results with all the unsupervised baseline techniques, except when LDA is used as feature extractor. In this case LR performed better.



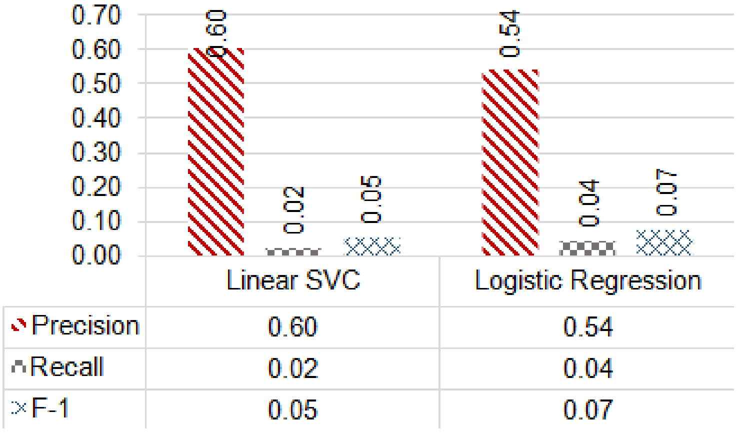
	Linear SVC	Logistic Regression
▨ Precision	0.63	0.75
▦ Recall	0.40	0.27
▩ F-1	0.49	0.40

Figure 4.4: Tag prediction performance using TF-IDF with paper titles.



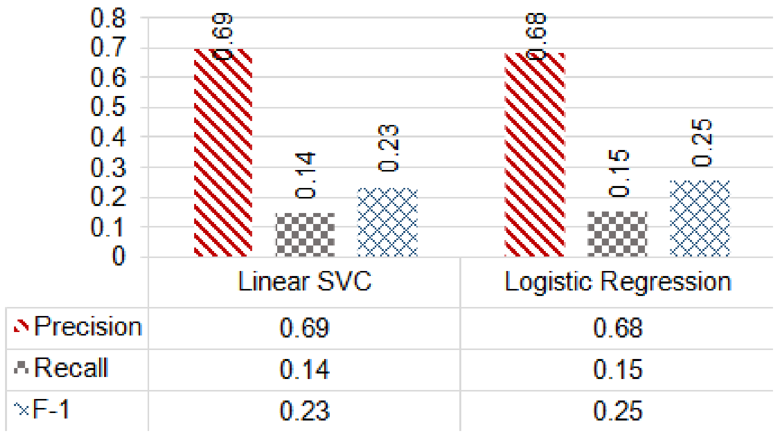
	Linear SVC	Logistic Regression
Precision	0.67	0.75
Recall	0.40	0.28
F-1	0.50	0.41

Figure 4.5: Tag prediction performance using TF-IDF with paper titles & abstracts.



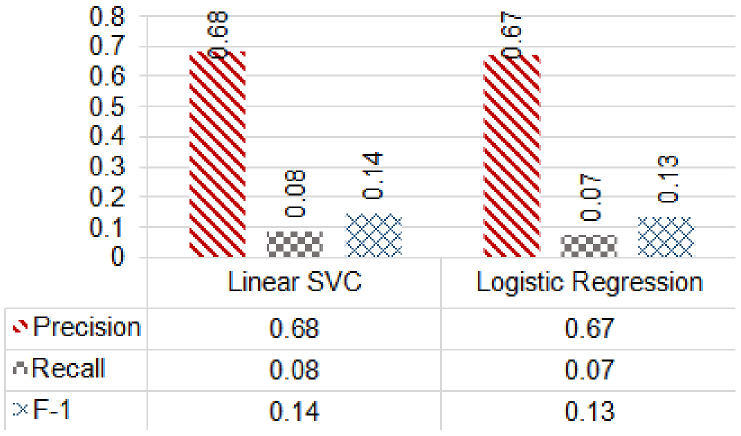
	Linear SVC	Logistic Regression
Precision	0.60	0.54
Recall	0.02	0.04
F-1	0.05	0.07

Figure 4.6: Tag prediction performance using LDA with paper titles.



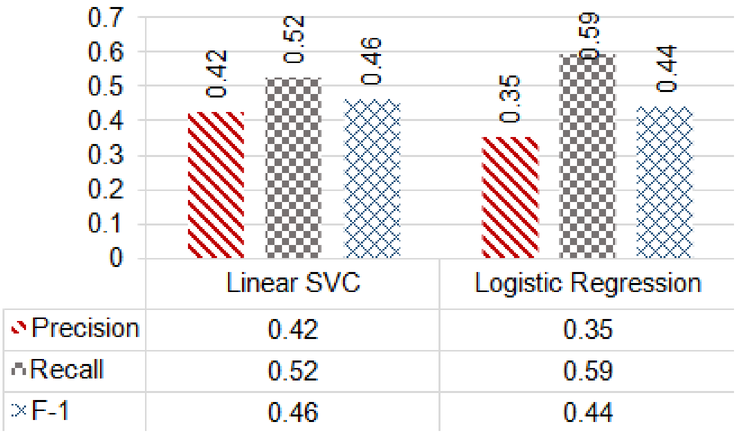
	Linear SVC	Logistic Regression
Precision	0.69	0.68
Recall	0.14	0.15
F-1	0.23	0.25

Figure 4.7: Tag prediction performance using LDA with paper titles & abstracts.



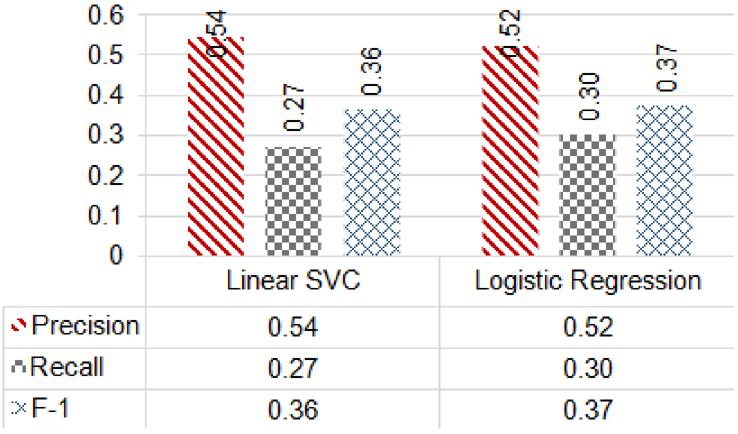
	Linear SVC	Logistic Regression
Precision	0.68	0.67
Recall	0.08	0.07
F-1	0.14	0.13

Figure 4.8: Tag prediction performance using Doc2Vec with paper titles.



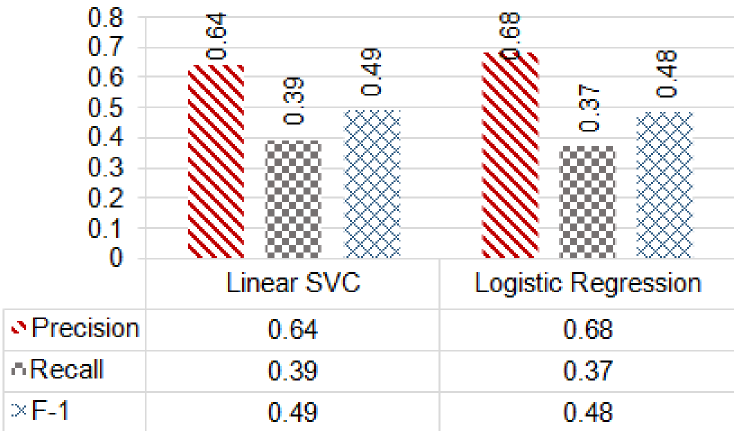
	Linear SVC	Logistic Regression
▨ Precision	0.42	0.35
▦ Recall	0.52	0.59
▩ F-1	0.46	0.44

Figure 4.9: Tag prediction performance using Doc2Vec with paper titles & abstracts.



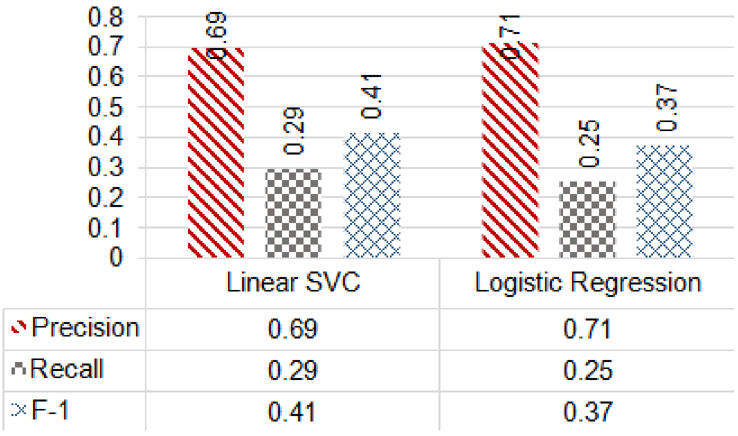
	Linear SVC	Logistic Regression
▨ Precision	0.54	0.52
▦ Recall	0.27	0.30
▩ F-1	0.36	0.37

Figure 4.10: Tag prediction performance using Sent2Vec with paper titles.



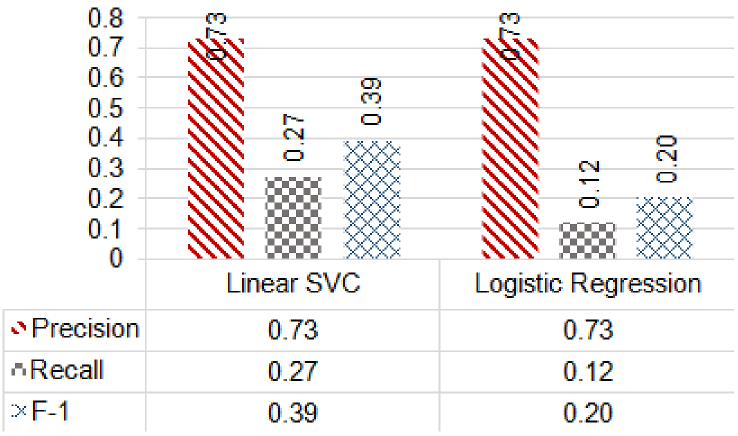
	Linear SVC	Logistic Regression
▨ Precision	0.64	0.68
▦ Recall	0.39	0.37
▩ F-1	0.49	0.48

Figure 4.11: Tag prediction performance using Sent2Vec with paper titles & abstracts.



	Linear SVC	Logistic Regression
▨ Precision	0.69	0.71
▦ Recall	0.29	0.25
▩ F-1	0.41	0.37

Figure 4.12: Tag prediction performance using GloVe centroid with paper titles.



	Linear SVC	Logistic Regression
▨ Precision	0.73	0.73
▨ Recall	0.27	0.12
▨ F-1	0.39	0.20

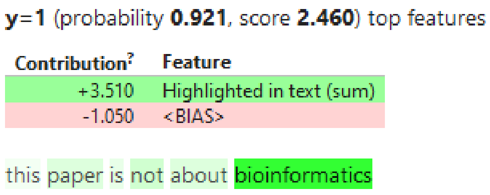
Figure 4.13: Tag prediction performance using GloVe centroid with paper titles & Abstracts.

A summary of the experimental results of the baselines in addition to our proposed technique is shown in Table 4.3. The following observations can be drawn by analyzing the obtained outcomes:

- Research paper titles combined with abstracts led to better classification performance in comparison with titles only.
- Traditional TF-IDF techniques performs quite good compared to LDA and other recent unsupervised techniques such as Sent2Vec, Doc2Vec, and Word embedddings centroid.
- LDA and Doc2Vec performed poorly when only research paper on research paper titles were used.
- TF-IDF even gave better results than CNN when the titles were used.

While CNN outperformed TF-IDF when titles and abstracts were concatenated.

- Bi-GRU outperformed centroid of word embeddings and CNN models. Hence, it is important to consider words sequence for the representation of research papers. Figures 4.14 and 4.15 show an example of classifying a research paper title to "bioinformatics" tag using centroid of word embeddings model and Bi-GRU model, respectively. Bi-GRU could assign more accurate weights to the words in the title, considering words sequence. The probability of the title being related to "bioinformatics" is less in Bi-GRU model, since the word "not" presents before "bioinformatics".

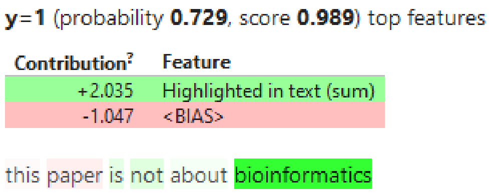


y=1 (probability **0.921**, score **2.460**) top features

Contribution?	Feature
+3.510	Highlighted in text (sum)
-1.050	<BIAS>

this paper is not about bioinformatics

Figure 4.14: Example of classifying a research paper title to "bioinformatics" tag using word embedding centroid.



y=1 (probability **0.729**, score **0.989**) top features

Contribution?	Feature
+2.035	Highlighted in text (sum)
-1.047	<BIAS>

this paper is not about bioinformatics

Figure 4.15: Example of classifying a research paper title to "bioinformatics" tag using Bi-GRU.

Table 4.3: Performance of the proposed approach and other baselines for tag prediction.

	Titles			Titles + Abstracts		
Methods	Precision	Recall	F1	Precision	Recall	F1
TF-IDF + SVM	0.63	0.40	0.49	0.67	0.40	0.50
LDA + LR	0.54	0.04	0.07	0.68	0.15	0.25
Doc2Vec + SVM	0.68	0.08	0.14	0.42	0.52	0.46
Sent2Vec + SVM	0.54	0.27	0.36	0.64	0.39	0.49
GloVe centroid + SVM	0.69	0.29	0.41	0.73	0.27	0.39
CNN	0.55	0.41	0.47	0.63	0.45	0.53
Bi-GRU	0.55	0.45	0.50	0.60	0.49	0.54
Bi-GRU+Attention	0.63	0.44	**0.52**	0.64	0.50	**0.56**

- Finally, our proposed model, Bi-GRU+Attention, has significantly outperformed the Bi-GRU model.

4.6 Summary

In this chapter, we proposed an attention-based bidirectional gated recurrent unit (bi-GRU) model for the tag prediction task in scholarly materials. We formulated our problem as a multi-label classification task, and utilized hierarchical word and sentence level attention networks for aggregating important words and sentences, in order to increase the general representation and visualization of the key concepts in research papers. The results of our experiments on a CiteULike dataset showed that the proposed approach is able to outperform state-of-the-art methods for text feature representation in tag prediction.

Chapter 5

Tag-Aware Matrix Factorization for Research Paper Recommendation

Finding relevant scientific articles has become difficult due to the increasing number of publications. Thus, scientific paper recommendation has become a very important research topic. Collaborative filtering (CF) is a successful recommendation approach, which uses the ratings given to items by users as a source of information for learning to make recommendations. However, the ratings are often very sparse as in the research paper domain, due to the huge number of publications growing every year. Therefore, more attention has been drawn to hybrid methods that consider both ratings and content information. Nevertheless, most of the hybrid recommendation approaches that are based on text embedding have utilized bag-of-words (BOW) techniques, which ignores word order and semantic meaning. In this chapter, we propose a hybrid approach that leverages deep semantic modeling technique and collaborative filtering to improve the performance of tag-aware personalized recommendation.

The experimental evaluation is performed on a real dataset from CiteULike. The results show that the proposed model is effective in recommending research papers when the rating data is very sparse.

5.1 Introduction

Recently, researchers can access growing archives of scientific articles. However, it became difficult for them to find articles relevant to their interests. One way that researchers find articles is by following citations in other articles that they are interested in, but this way limits the researchers to specific citation communities. Therefore, recommender systems (RSs) are becoming important to make effective use of available information.

Collaborative filtering is one of the most popular approaches for RSs. Matrix factorization (MF) is a collaborative filtering based technique, which has become a dominant solution for personalized recommendation and has been reported to be superior to memory-based techniques [KBV09]. However, there exists a “cold start” problem in MF: many users only give very few ratings, resulting in a very sparse user-item rating matrix, and making it difficult to summarize users' preferences. A widely adopted solution is to incorporate additional sources of information about items.

Since scientific article content is large, a good representation based on text is essential. As mentioned in Chapter 3, approaches based on document modeling methods such as Latent Dirichlet Allocation (LDA) and Stacked Denoising Auto Encoder (SDAE) based on TF-IDF have been proposed to represent research papers content. However, those methods are based on the BOW model that ignores information such as word order and semantic meaning in the textual content. Therefore, there is a need to utilize advanced deep learning NLP techniques.

In this system, we adopt the model we proposed in Chapter 4 for social tag prediction, which is based on bidirectional gated recurrent units (bi-GRUs) and attention networks, to capture semantic vector representation of items, then we incorporate those tag-aware item representations into an MF technique in order to derive item rankings for users.

5.2 Methodology

Matrix Factorization

The idea of Matrix Factorization (MF) is to characterize both items and users through vectors of factors inferred from item rating patterns. High correspondence between item and user factors leads to a recommendation. These methods have become popular in recent years by combining good scalability with predictive accuracy. One strength of MF is that it allows for incorporation of additional information.

RSs rely on different types of input data, which are often placed in a matrix with one dimension representing users and the other dimension representing items of interest. In RSs, the most convenient rating is the explicit feedback, which includes explicit input by users regarding their interest in items. For example, Netflix[1] collects star ratings for movies. We refer to explicit user feedback as ratings. Usually, explicit feedback comprises a sparse matrix, since any single user is likely to have rated only a small percentage of possible items as in the research paper domain.

[1]https://www.netflix.com

When explicit feedback is not available, RSs can infer user preferences using implicit feedback [HKV08], which indirectly reflects opinion by observing user behavior including purchase history, browsing history, search patterns, or even mouse movements. Implicit feedback usually denotes the presence or absence of an event, so it is typically represented by a densely filled matrix.

As illustrated in Figure 5.1, MF models map both users and items to a joint latent factor space of dimensionality k, such that user-item interactions are modeled as inner products in that space. Accordingly, each item j is associated with a vector $p_j \in k$, and each user i is associated with a vector $u_i \in k$. For a given item j, the elements of p_j measure the extent to which the item possesses those factors, positive or negative. For a given user i, the elements of u_i measure the extent of interest the user has in items that are high on the corresponding factors, again, positive or negative. The resulting dot product, $p_j^T u_i$, captures the interaction between user i and item j — the user's overall interest in the item's characteristics. This approximates user i's rating of item j, which is denoted by $\hat{r_{ij}}$, leading to the following estimate:

$$\hat{r_{ij}} = p_j^T u_i \tag{5.1}$$

The major challenge is computing the mapping of each item and user to factor vectors $p_j, u_i \in k$. After the recommender system completes this mapping, it can easily estimate the rating a user will give to any item.

Tag-Aware Matrix Factorization

To alleviate the cold start problem in traditional MF, a widely adopted solution is to incorporate additional sources of information about items or users to achieve additional information-based MF. Therefore, inspired by the recent

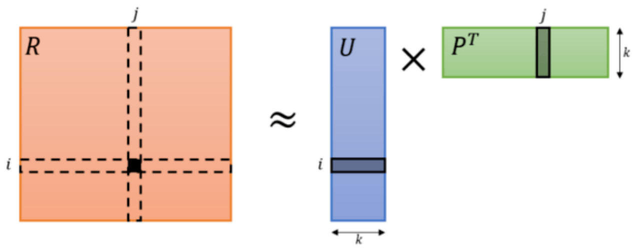


Figure 5.1: Main idea of matrix factorization.

development of deep-semantic modeling, we propose a hybrid deep semantic MF model to tackle these problems and to further enhance the performance of tag-aware personalized recommendation, by integrating the techniques of deep semantic modeling, hybrid learning, and MF.

We extend the MF model by combining item textual semantic representation, which we obtain from training a deep learning model for social tag recommendation as proposed in Chapter 4. We extract the document representations from the last layer in the model (i.e., the layer used for classification) as item vector representations. In this case, the item vector length is equal to the number of tags. Figure 5.2 illustrates the proposed approach.

We used LightFM[2], a Python library for implementing hybrid RSs, where user and item metadata can be incorporated into the traditional MF algorithms [Kul15]. A user (or item) vector is the sum of the vectors associated to its constituent features. Similarly, a user (or item) bias term is just the sum of the bias terms associated to its features. The probability $\hat{r_{ij}}$ of an interaction between a user i and an item j is modelled as the *sigmoid* of the dot product

[2]https://github.com/lyst/lightfm

of the user vector and the item vector, along with the bias terms associated with the user and the item. In our case, we have only item related features.

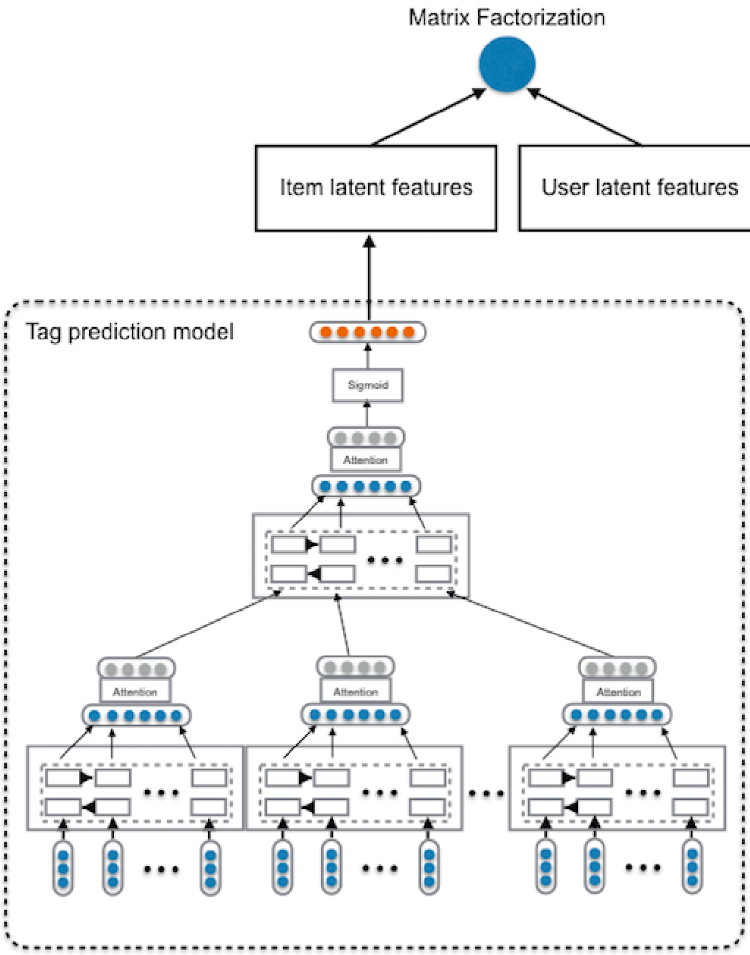


Figure 5.2: The proposed deep tag-aware matrix factorization approach.

5.3 Experimental Evaluation

Dataset

We used *citeulike-a* dataset as in Chapter 4. It has 5551 users, 16980 items, 46391 tags and ratings in 0.22% of its user-item matrix entries. In our experiments, we extracted only the top 300 tags and their 13356 related items to train the model proposed in Chapter 4 and to extract document representations. While, we used the whole user-item entries for validating the hybrid approach that we propose in this chapter.

Evaluation Methodology

For the *citeulike-a* dataset, we randomly selected P items associated with each user to form the training set and we used all the rest of the dataset as the test set. In our experiments, we set P to 10. As in [WB11, WWY15], we used recall as the performance measure since the rating information is in the form of implicit feedback. Which means a zero entry may be due to the fact that the user is not interested in the item, or that the user is not aware of its existence. Therefore, precision is not a suitable performance measure. And like most of the RSs, we sort the predicted ratings of the candidate items and recommend the top K items to the target user. The recall@K for each user is then defined as:

$$\text{Recall@}K = \frac{\text{number of items that the user likes among the top-K}}{\text{total number of items that the user likes}} \tag{5.2}$$

The final result reported the average recall over all users.

Baselines

In order to perform an empirical evaluation of the proposed method (MF-GRUTags), we made a comparison with the following baseline methods:

- **BPR**. Bayesian Personalized Ranking [RFGST09], a popular pair-wise ranking method for recommendations. It maximises the prediction difference between a positive example and a randomly chosen negative example. It is useful when only positive interactions are present as in our case (implicit feedback).

- **WARP**. Weighted Approximate-Rank Pairwise [WYW13], another commonly used pair-wise ranking method for recommendations. It maximises the rank of positive examples by repeatedly sampling negative examples until rank violating one is found. It is useful when only positive interactions are present and optimising the top of the recommendation list (precision@K) is desired.

- **MF-Tags**. A hybrid approach that incorporates item tags as features in an MF technique using the LightFM library.

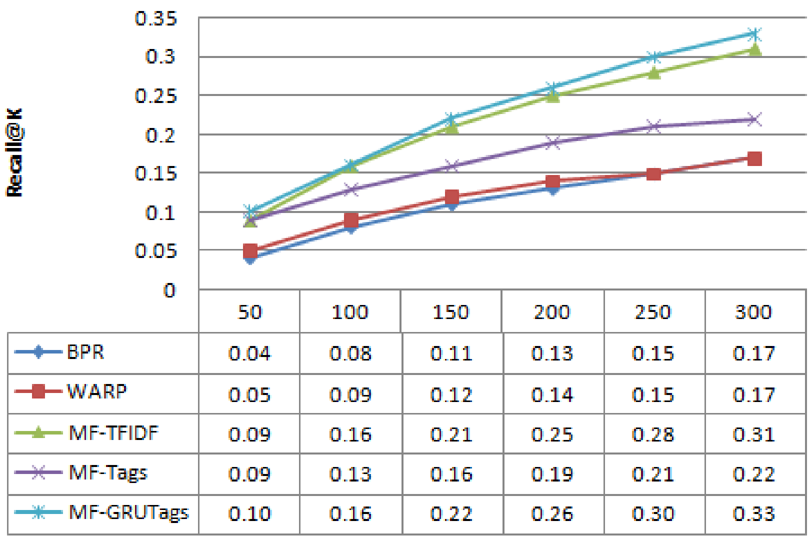
- **MF-TFIDF**. A hybrid MF model based on LightFM, representing items by their content features using TF-IDF.

- **MF-GRUTags**. Is our proposed model.

We used WARP as loss function for all the experimented hybrid approaches.

5.4 Results and Discussion

The performance of MF-GRUTags has been validated with the baselines based on the exploited dataset. As shown in Figure 5.3, WARP has slightly outperformed BPR. Incorporating TF-IDF as item features increased the recall almost to the double compared with the basic MF techniquies (BPR and WARP). Incorporating tags information also gave better recall but still less than the TF-IDF based one. The recall increased over the TF-IDF approach using our proposed model, which shows the effectiveness of using deep semantic representation of research papers based on social tags for this task.



	50	100	150	200	250	300
◆ BPR	0.04	0.08	0.11	0.13	0.15	0.17
■ WARP	0.05	0.09	0.12	0.14	0.15	0.17
▲ MF-TFIDF	0.09	0.16	0.21	0.25	0.28	0.31
× MF-Tags	0.09	0.13	0.16	0.19	0.21	0.22
✱ MF-GRUTags	0.10	0.16	0.22	0.26	0.30	0.33

Figure 5.3: Performance comparison between BPR, WARP, MF-TFIDF, MF-Tags and MF-GRUTags based on recall@K.

5.5 Summary

In this chapter, we proposed a personalized research paper recommendation approach, which integrates MF technique and semantic-aware document representations as items metadata. The document representations were extracted from a social tag prediction model that utilizes bidirectional GRUs and attention mechanism for aggregating important words and sentences, in order to increase the general representation and visualization of the key concepts in research papers. The results of our experiments on the CiteULike dataset show that the proposed approach is able to outperform state-of-the-art collaborative filtering based techniques.

Chapter 6

Reranking Research Paper Recommendations using Pre-trained Sentence Encoders

The task of content-based matching is challenging, mainly due to the problem of determining the semantic similarity of texts. Nowadays, there exist many sentence embedding models that learn deep semantic representations by being trained on huge corpora, aiming to provide transfer learning to a wide variety of NLP tasks. In this chapter, we present a comparative evaluation among five well-known pre-trained sentence encoders deployed in the pipeline of title-based research paper recommender system (RS). The experimented encoders are USE, BERT, InferSent, ELMo, and SciBERT. In this study, we propose a methodology for evaluating such models in reranking BM25-based recommendations. The experimental results show that the sole consideration of semantic information from these encoders does not lead to improved recommendation performance over the traditional BM25 technique, while their integration enables the retrieval of a set of relevant papers that may not be retrieved by the BM25 ranking function.

6.1 Introduction

Currently, deep learning (DL) research trends have emerged in text representation learning. DL for NLP leads towards increasingly powerful and complex models, such as RNNs, LSTMs, and attention models architectures. Several among these models use, for example, labelled datasets of paraphrase pairs to obtain sentence embeddings in a supervised manner [WBGL16, CKS+17] to learn sentence embeddings. The increased complexity of these models makes them slower to train on larger datasets. Here comes the importance of Transfer Learning.

Transfer Learning stores the knowledge gained from solving source tasks (usually with abundant annotated data), and apply it to other tasks (usually suffer from insufficient annotated data to train complex models), to solve the inadequate supervision problem, which has become prevalent in many applications, such as the ones related to the research paper domain.

Pre-trained sentence encoders such as Google's BERT and USE, Facebook's InferSent, and AllenAI's SciBERT and ELMo, are considered applications of Transfer learning and they have received significant attention in recent years. These pre-trained models can encode a sentence into deep contextualized embeddings. They have been reported to outperform previous state-of-the-art approaches such as traditional word embeddings, for many NLP tasks [CKS+17, DC19]. Such tasks also include the calculation of semantic similarity and relatedness, which is key in developing effective research paper RSs. If sentence encoders performed for calculating relatedness of research papers as good as for other tasks, this would mean a great advancement for research paper RSs.

In the research literature, there are some works on using document embeddings for ranking research papers based on semantic relatedness (e.g., [CB19]), but - as far as we know - no work on exploiting pre-trained sentence embeddings for the same task. The only works we are aware of are related to utilizing BERT for document retrieval. However, those works are focused on different domains such as social media posts [YZL19], news [DC19, YZL19], and web pages [DC19].

Therefore, our goal is to find how well some of the most common sentence encoders perform when the task is the research paper recommendation, more precisely the identification of related research papers for non-profiled users such as in [NLdSG11] and [BABG17] mentioned in Chapter 3. To the best of our knowledge, we are the first to conduct such an evaluation in the field of research paper recommendation.

6.2 Methodology

Problem Definition

We focus on the task of related-article recommendations, where a RS receives one paper as input, and returns a list of related papers. In our experiments, research papers are represented by their title only. While this may not be ideal to leverage the full potential of sentence encoders, using only the title is a realistic scenario as many research paper RSs do not use full-texts but only the title (and sometimes the abstract) [BGLB16].

Recommendation Pipeline

Our recommendation pipeline follows the classical workflow of content-based recommendation framework. It can be split into four steps:

- Given a set of research papers R, each $r \in R$ is indexed using Apache Lucene/Solr.
- Given a query title q, provided by a user, a set of N similar research paper titles are retrieved by Apache Lucene/Solr BM-25 ranking function.
- For each $n \in N$, and q, sentence embeddings are calculated using one of the pre-trained sentence encoders that we experiment.
- Recommendations are calculated by exploiting classic similarity measures: items are ranked according to their decreasing similarity, and top-N recommendations are returned to the user.

Figure 6.1 shows the overall architecture of the proposed approach.

Sentence Embeddings

We experimented with five pre-trained sentence encoders to transform the input paper and candidates papers in the corpus into sentence embeddings. The encoders are described in the following sections.

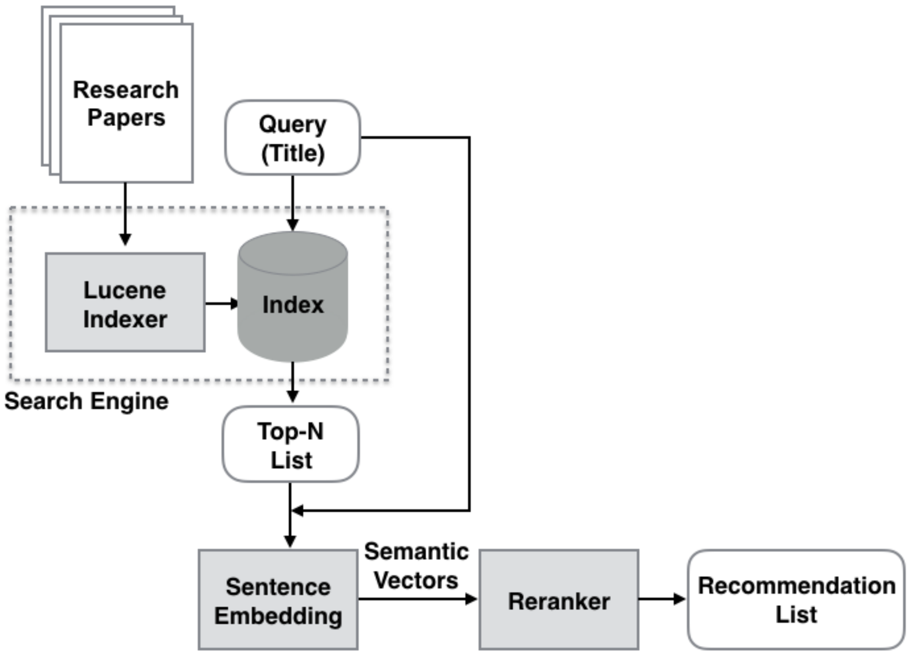


Figure 6.1: The proposed approach for reranking recommendations using sentence encoders.

Universal Sentence Encoder (USE)

It has two models available to download from Tensorflow Hub[1]: the former trained with a Deep Averaging Network (DAN)[2], the latter with a Transformer[3] network. The original Transformer model constitutes an encoder and decoder, but only encoder part is used. The encoder is composed of a stack of $N = 6$ identical layers. Each layer has two sub-layers. The first is a multi-head self-

[1]https://www.tensorflow.org/hub
[2]https://tfhub.dev/google/universal-sentence-encoder/2
[3]https://tfhub.dev/google/universal-sentence-encoder-large/3

attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. They also employ a residual connection around each of the two sub-layers, followed by layer normalization. The transformer based encoder achieves the best overall transfer task performance. However, this comes at the cost of computing time and memory usage scaling dramatically with sentence length.

Deep Averaging Network (DAN) is much simpler where input embeddings for words and bi-grams are first averaged together and then passed through a feedforward DNN to produce sentence embeddings. The primary advantage of the DAN encoder is that its computation time is linear in the length of the input sequence.

Both Transformer and DAN models are trained on a variety of web sources, such as Wikipedia, web news, web question-answer pages, and supervised data from the Stanford Natural Language Inference (SNLI) corpus[4], a set of 570k pairs of sentences labelled with 3 categories: neutral, contradiction and entailment. Both models return vectors of 512 dimensions as output [CYK+18]. Figure 6.2 clarifies the differences between Transformer and DAN architectures.

InferSent

It adopts a bidirectional LSTM (biLSTM) completed with a max-pooling operator as sentence encoder. InferSent is trained on the SNLI corpus. Both sentences are encoded using the same encoder while the classifier is trained on a pair representation constructed from the two sentence embeddings as shown in Figure 6.3. We experimented with two models of InferSent[5]: the former

[4]https://nlp.stanford.edu/projects/snli/

[5]https://github.com/facebookresearch/InferSent

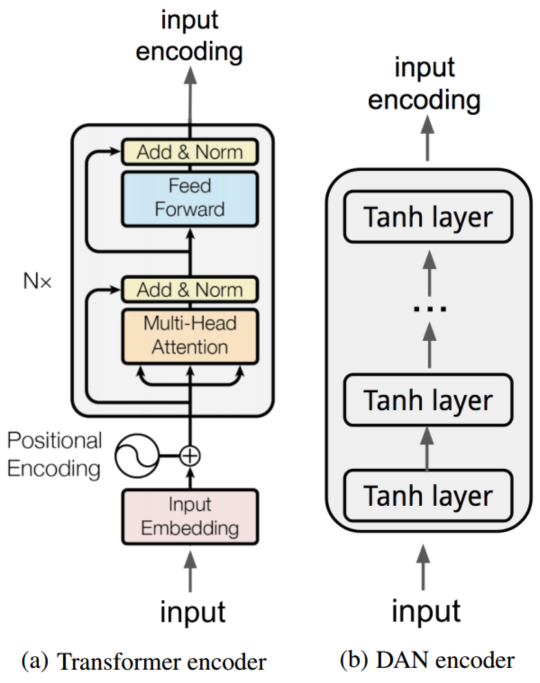


Figure 6.2: USE's Transformer vs. DAN architectures.

trained using GloVe word embeddings, the latter using fastText word embeddings. The output of InferSent is an embedding of 4096 dimensions [CKS+17].

ELMo (Embedding from Language Models)

ELMo's inputs are characters rather than words. They can, thus, take advantage of sub-word units to compute meaningful representations even for out-of-vocabulary words (like FastText). ELMo uses biLSTM in training, so that its language model not only understands the next word, but also the previous

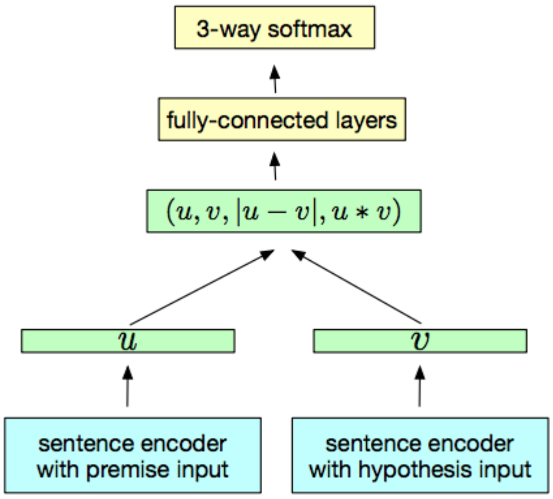

3-way softmax
fully-connected layers
$(u, v, |u - v|, u * v)$
u
v
sentence encoder with premise input
sentence encoder with hypothesis input

Figure 6.3: InferSent architecture (from [CKS+17]).

word in the sentence. It contains a 2-layer biLSTM backbone. ELMo are concatenations of the activations on several layers of the biLSTMs as shown in Figure 6.4. Different layers of a language model encode different kinds of information on a word (e.g., Part-Of-Speech tagging is well predicted by the lower level layers of a biLSTM while word-sense disambiguation is better encoded in higher-levels). Concatenating all layers allows to freely combine a variety of word representations for better performances on downstream tasks.

We used TensorFlow Hub implementation of ELMo[6], trained on the 1 Billion Word Benchmark. ELMo returns a representation of 1024 dimensions [PNI+18].

[6]https://tfhub.dev/google/elmo/2

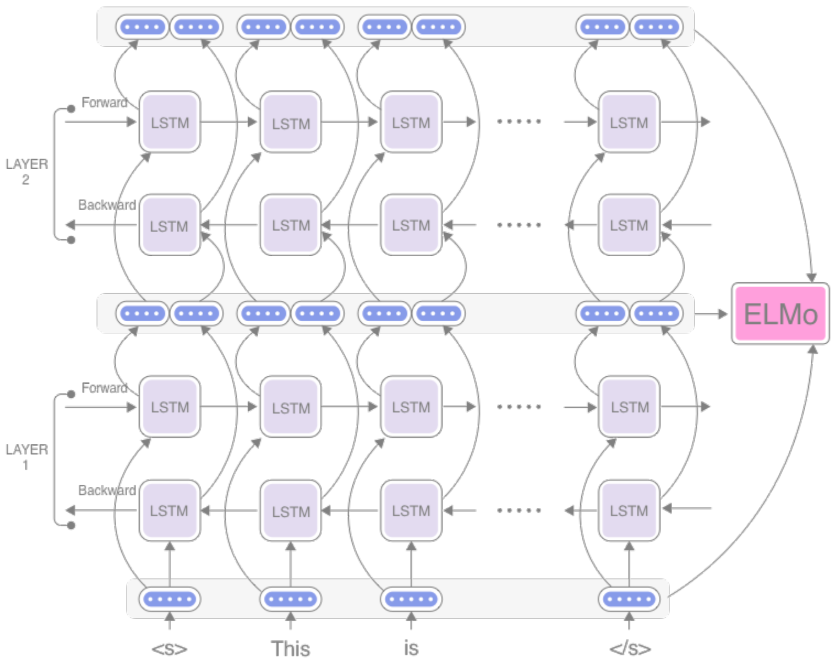


Figure 6.4: ELMo architecture.

BERT (Bidirectional Encoder Representations from Transformer)

It is a sentence embedding model that learns vector representations by training a deep bidirectional Transformer network. BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. In its vanilla form, Transformer includes two separate mechanisms — an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT's goal is to generate a language model, only the encoder mechanism is necessary.

We used the uncased BERT-Base model[7], which consists of 12 hidden layers and 768 attention heads, and trained on Wikipedia[8], through bert-as-service[9] to obtain a vector of 768 dimensions [DCLT18]. Bert-as-service uses BERT as a sentence encoder and hosts it as a service via ZeroMQ[10], allowing us to map sentences into fixed-length representations in just two lines of code. BERT-base's high-level architecture is shown in Figure 6.5.

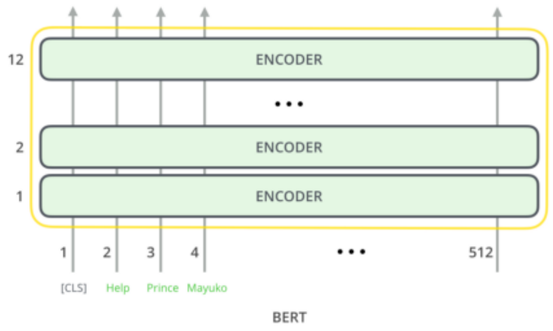


Figure 6.5: BERT-Base architecture.

SciBERT

It is a BERT model, but trained on a corpus of 1.14M scientific papers [BCL19]. We used the recommended uncased scibert-scivocab version[11]. Similar to BERT, we obtain a vector of 768 dimensions using bert-as-service.

[7]https://github.com/google-research/bert
[8]https://www.wikipedia.org/
[9]https://github.com/hanxiao/bert-as-service
[10]https://zeromq.org/
[11]https://github.com/allenai/scibert

Reranking

In the reranking step, we calculate the cosine similarity between the sentence embedding of the input paper title and embeddings of all the candidate paper titles. This similarity metric expresses the semantic similarity. We perform a linear combination between the initial scores from BM25 after being normalized, and the semantic similarity scores from sentence embeddings, by summing up the scores (with uniform weights set to 0.5) to generate the final ranked recommendations.

Baseline and Hybrid Approaches

As strong baseline we use BM25, a common approach for document ranking [BABG17]. Today's sentence encoders have a major drawback, that is, the high execution time on large corpora. Using sentence embeddings on large corpora seems hardly feasible in a production RS, which needs to return recommendations within a few seconds or less. Hence, we do not only compare the embeddings with BM25, but we additionally experiment with hybrid approaches in which we first use Apache Lucene's BM25 to retrieve a list of top-20, 50 or 100 recommendation candidates, and then we rerank that list instead of the entire corpus, with the sentence encoders.

6.3 Offline Evaluation

Dataset

For the evaluation, we used the CiteULike dataset [WB11]. It contains paper collections of 5,551 researchers, i.e., lists of which documents researchers added

Table 6.1: Statistics of *citeulike-a* dataset.

#Users	#Papers	User-Paper
5,551	16,980	204,986

to their personal document collection. Table 6.1 shows the statistics of the dataset.

Evaluation Methodology

Using *5*-fold cross-validation, we split the data by randomly selecting one of the research paper titles that a user has in her library as input paper, and all the remaining paper titles are used for evaluating if the recommended papers were actually in the user's library.

As evaluation metrics, we calculated precision, recall, and Mean Average Precision (MAP) at rank 10.

Precision measures the capability of the system to reclaim as much relevant research papers as possible in response to the target paper request.

$$\text{Precision} = \frac{\sum(\text{relevant papers }) \cap \sum(\text{retrieved papers})}{\sum(\text{retrieved papers})} \tag{6.1}$$

Recall measures the capability of the system to reclaim as few irrelevant research papers as possible in response to the target paper request.

$$\text{Recall} = \frac{\sum(\text{relevant papers}) \cap \sum(\text{retrieved papers})}{\sum(\text{relevant papers})} \tag{6.2}$$

As users often scan only documents presented at the top ranked of the recommendation list, we feel it is imperative to also measure the system's ability to provide useful recommendations at the top of the recommendation list using the one of the most widely used ranked information retrieval evaluation measures: Mean Average Precision (MAP).

Average Precision (AP) is the average precision values at all ranks where relevant research papers are found, and MAP is the average of all APs. The Average Precision (AP) is calculated as follows:

$$\text{AP} = \frac{1}{m} \sum_{k=1}^{N} P\left(R_k\right) \tag{6.3}$$

where for a user u, m is the number of relevant papers to u, N is the whole number of the papers in recommended list, $P(R_k)$ represents the precision of retrieved results from the top result until get to paper k.

MAP gives an average of each user's AP value:

$$\text{MAP} = \frac{1}{U} \sum_{k=1}^{U} AP(k) \tag{6.4}$$

We also assess the efficiency of the reranking process by reporting data related to the recommendation processing time in this scenario. Our tests were performed on a computer equipped with Intel Core i7-6700 CPU and 16GB RAM.

Results and Discussion

We report the performance of standalone techniques (i.e., only sentence encoder scores are used for reranking), and hybrid techniques (i.e., BM25 scores are combined with sentence encoder similarity scores for reranking). BM25 without any reranking is used as baseline in both cases. The following observations can be drawn by analyzing the obtained outcomes:

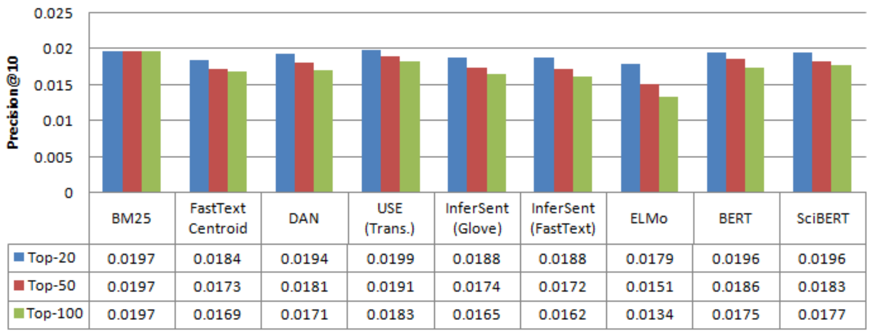
- Figures 6.6, 6.7, and 6.8 show that none of the sentence embedding models alone is able to outperform BM25 in terms of precision, recall, and F1.

- USE, BERT and SciBERT outperform ELMo and InferSent, on average. One possible reason is that USE, BERT, and SciBERT are trained on corpora that contain technical and scientific terms (i.e., USE and BERT on Wikipedia, SciBERT on scientific papers), whereas ELMo and InferSent are trained on a news crawl and a natural language inference corpus, respectively.

- Figures 6.9, 6.10, and 6.11 show that the hybrid approaches (BM25 + sentence embeddings) outperforms BM25 and all other standalone approaches (results are statistically significant with $p < 0.05$ in ANOVA test). Apparently, in some cases, BM25 fails to assign the right ranking scores to papers, while sentence embeddings could capture the semantic similarity between them. In this case, the ranking performance increases with the top-N number of papers retrieved by BM25, which means that more relative papers can be found.

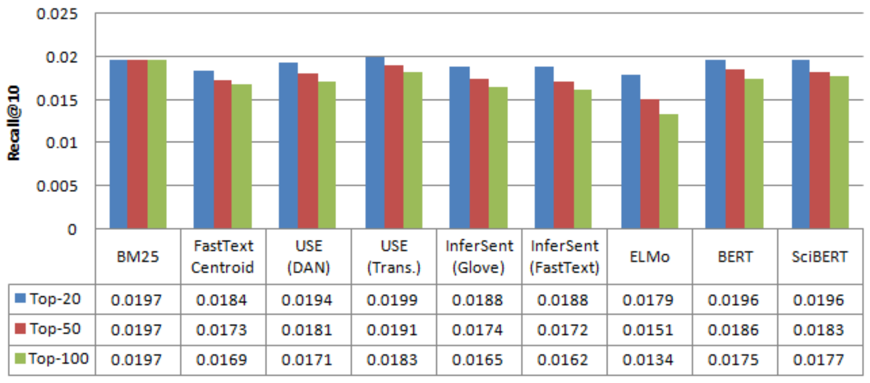
- BM25 + USE (Transformer) performs best. Compared to BM25, it relatively increases MAP@10 by +5.29% when reranking 20 titles, by +6.47% when reranking 50, and by +7.35% when reranking 100 titles.

In our experiments, BM25 queries took around 5 milliseconds to retrieve up to 100 results. The extra time taken to calculate embeddings and reranking 20, 50 and 100 titles through the different models is shown in Figure 6.12. The results can be concluded as follows:



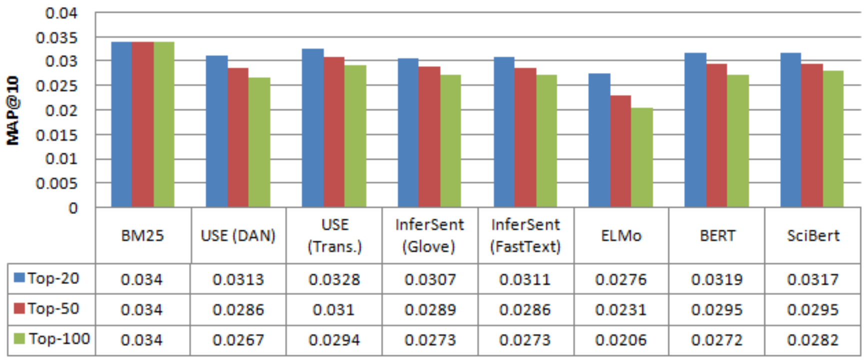
	BM25	FastText Centroid	DAN	USE (Trans.)	InferSent (Glove)	InferSent (FastText)	ELMo	BERT	SciBERT
🟦 Top-20	0.0197	0.0184	0.0194	0.0199	0.0188	0.0188	0.0179	0.0196	0.0196
🟥 Top-50	0.0197	0.0173	0.0181	0.0191	0.0174	0.0172	0.0151	0.0186	0.0183
🟩 Top-100	0.0197	0.0169	0.0171	0.0183	0.0165	0.0162	0.0134	0.0175	0.0177

Figure 6.6: Precision@K of standalone reranking techniques.



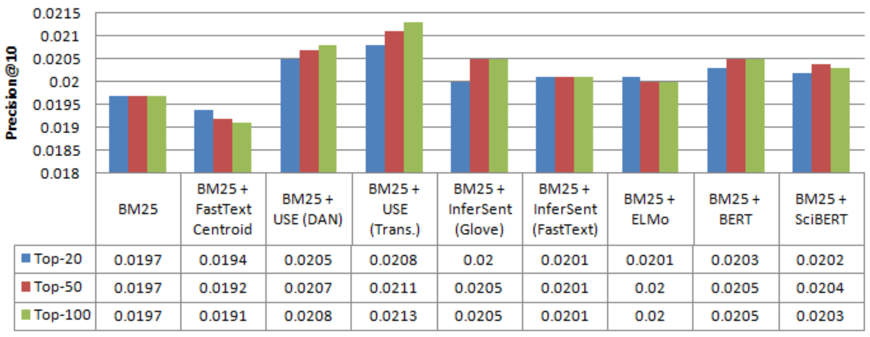
	BM25	FastText Centroid	USE (DAN)	USE (Trans.)	InferSent (Glove)	InferSent (FastText)	ELMo	BERT	SciBERT
🟦 Top-20	0.0197	0.0184	0.0194	0.0199	0.0188	0.0188	0.0179	0.0196	0.0196
🟥 Top-50	0.0197	0.0173	0.0181	0.0191	0.0174	0.0172	0.0151	0.0186	0.0183
🟩 Top-100	0.0197	0.0169	0.0171	0.0183	0.0165	0.0162	0.0134	0.0175	0.0177

Figure 6.7: Recall@K of standalone reranking techniques.



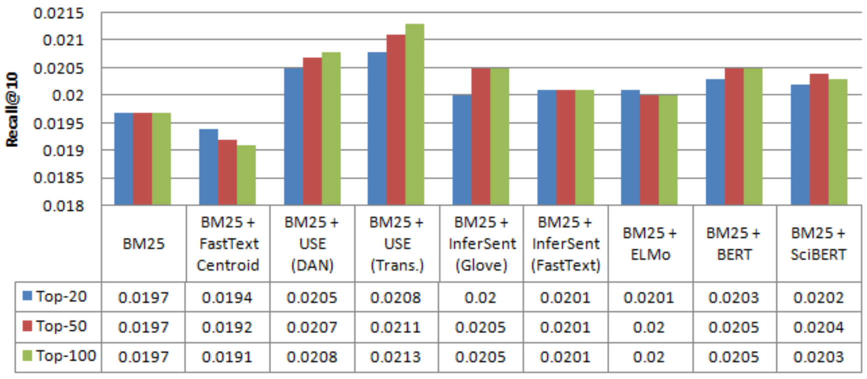
	BM25	USE (DAN)	USE (Trans.)	InferSent (Glove)	InferSent (FastText)	ELMo	BERT	SciBert
🟦 Top-20	0.034	0.0313	0.0328	0.0307	0.0311	0.0276	0.0319	0.0317
🟥 Top-50	0.034	0.0286	0.031	0.0289	0.0286	0.0231	0.0295	0.0295
🟩 Top-100	0.034	0.0267	0.0294	0.0273	0.0273	0.0206	0.0272	0.0282

Figure 6.8: MAP@K of standalone reranking techniques.



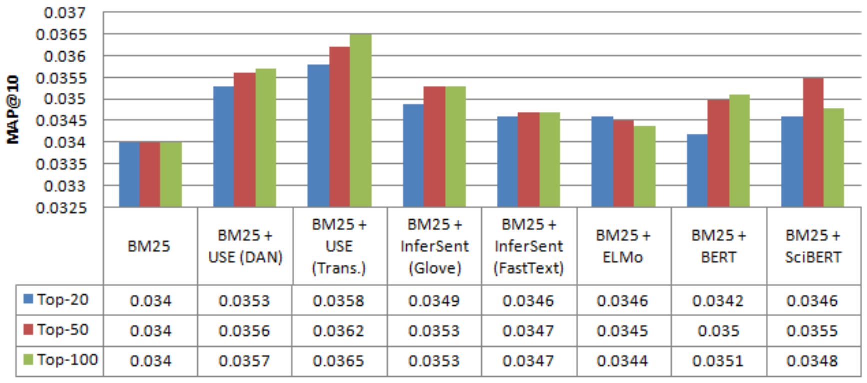
	BM25	BM25 + FastText Centroid	BM25 + USE (DAN)	BM25 + USE (Trans.)	BM25 + InferSent (Glove)	BM25 + InferSent (FastText)	BM25 + ELMo	BM25 + BERT	BM25 + SciBERT
🟦 Top-20	0.0197	0.0194	0.0205	0.0208	0.02	0.0201	0.0201	0.0203	0.0202
🟥 Top-50	0.0197	0.0192	0.0207	0.0211	0.0205	0.0201	0.02	0.0205	0.0204
🟩 Top-100	0.0197	0.0191	0.0208	0.0213	0.0205	0.0201	0.02	0.0205	0.0203

Figure 6.9: Precision@K of hybrid reranking techniques.



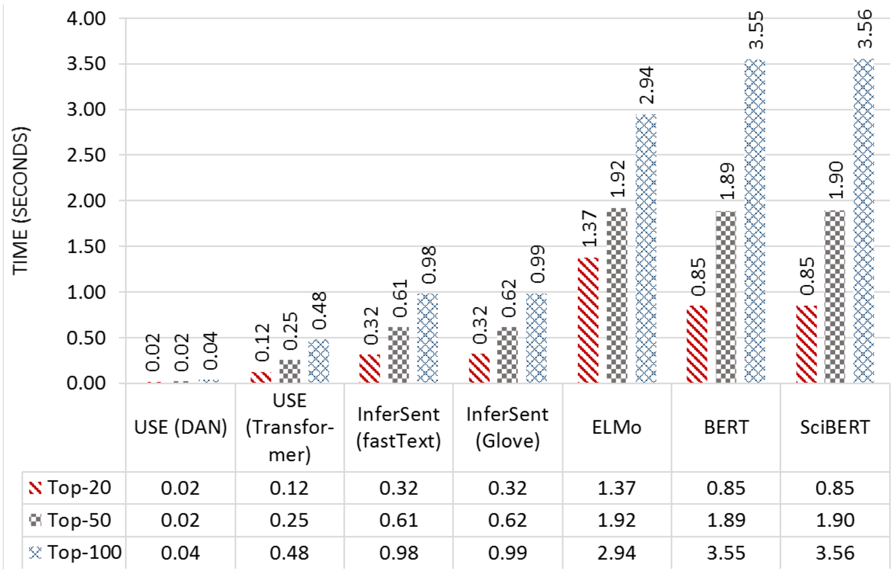
	BM25	BM25 + FastText Centroid	BM25 + USE (DAN)	BM25 + USE (Trans.)	BM25 + InferSent (Glove)	BM25 + InferSent (FastText)	BM25 + ELMo	BM25 + BERT	BM25 + SciBERT
🟦 Top-20	0.0197	0.0194	0.0205	0.0208	0.02	0.0201	0.0201	0.0203	0.0202
🟥 Top-50	0.0197	0.0192	0.0207	0.0211	0.0205	0.0201	0.02	0.0205	0.0204
🟩 Top-100	0.0197	0.0191	0.0208	0.0213	0.0205	0.0201	0.02	0.0205	0.0203

Figure 6.10: Recall@K of hybrid reranking techniques.



	BM25	BM25 + USE (DAN)	BM25 + USE (Trans.)	BM25 + InferSent (Glove)	BM25 + InferSent (FastText)	BM25 + ELMo	BM25 + BERT	BM25 + SciBERT
🟦 Top-20	0.034	0.0353	0.0358	0.0349	0.0346	0.0346	0.0342	0.0346
🟥 Top-50	0.034	0.0356	0.0362	0.0353	0.0347	0.0345	0.035	0.0355
🟩 Top-100	0.034	0.0357	0.0365	0.0353	0.0347	0.0344	0.0351	0.0348

Figure 6.11: MAP@K of hybrid reranking techniques.



	USE (DAN)	USE (Transfor-mer)	InferSent (fastText)	InferSent (Glove)	ELMo	BERT	SciBERT
▨ Top-20	0.02	0.12	0.32	0.32	1.37	0.85	0.85
▦ Top-50	0.02	0.25	0.61	0.62	1.92	1.89	1.90
▩ Top-100	0.04	0.48	0.98	0.99	2.94	3.55	3.56

Figure 6.12: Reranking time in seconds using the different sentence encoders.

- USE (DAN) is the fastest, taking around 0.02 seconds to rerank 20 or 50 titles, and 0.03 seconds to rerank 100 titles.

- ELMo is the slowest in reranking 20 or 50 titles.

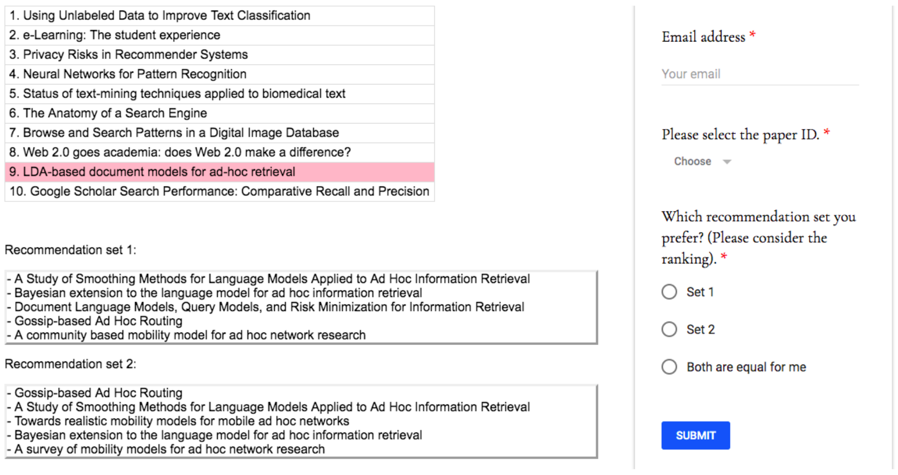
- Finally, BERT and SciBERT using bert-as-service are the slowest in reranking 100 titles, taking around 4.0 seconds. This means that they could not be used for real-time reranking recommendations, unless higher computing resources (e.g., GPU or TPU) were provided.

6.4 User Study

We conducted a user study in order to validate the results obtained from the offline evaluation. Nine volunteers were selected to participate in the user study. Their area of expertise covered various topics in computer science including Machine Learning, RSs, and NLP. The user study participants were two academics, three PhD students, three master students and one software developer, all with computer science specialty.

The general objective of the study was to receive expert feedback and opinion about the correctness of the conclusion received from the offline evaluation, which concludes that the best performing algorithm is "BM25 + USE (Transformer)" in comparison to BM25 as baseline. Figure 6.13 shows the structure of the user study survey. We provided the same list of ten papers from different domains in computer science to each participant, with the title and abstract of those papers. The participants were asked to select a paper in order to receive two recommendation sets, each of them with five recommended papers generated by different algorithms randomly, i.e., the baseline (BM25) and the best performing as per the offline evaluation (BM25 + Transformer).

Using a questionnaire, the participants were asked to anonymously fill out with their opinions and evaluate which recommendation set they think is better (Set 1 or Set 2) and if both look equal to her, considering the semantic similarity to the selected paper and the ranking of the recommended papers. We asked the participants to submit their responses for at least one out of the ten papers.



1. Using Unlabeled Data to Improve Text Classification
2. e-Learning: The student experience
3. Privacy Risks in Recommender Systems
4. Neural Networks for Pattern Recognition
5. Status of text-mining techniques applied to biomedical text
6. The Anatomy of a Search Engine
7. Browse and Search Patterns in a Digital Image Database
8. Web 2.0 goes academia: does Web 2.0 make a difference?
9. LDA-based document models for ad-hoc retrieval
10. Google Scholar Search Performance: Comparative Recall and Precision

Recommendation set 1:

- A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval
- Bayesian extension to the language model for ad hoc information retrieval
- Document Language Models, Query Models, and Risk Minimization for Information Retrieval
- Gossip-based Ad Hoc Routing
- A community based mobility model for ad hoc network research

Recommendation set 2:

- Gossip-based Ad Hoc Routing
- A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval
- Towards realistic mobility models for mobile ad hoc networks
- Bayesian extension to the language model for ad hoc information retrieval
- A survey of mobility models for ad hoc network research

Email address *

Your email

Please select the paper ID. *

Choose

Which recommendation set you prefer? (Please consider the ranking). *

○ Set 1

○ Set 2

○ Both are equal for me

SUBMIT

Figure 6.13: User study graphical user interface.

Results and Discussion

The results of the user study are shown in Figure 6.14. We received 63 responses for the questionnaire, the majority which is around 59% of the answers were Set 1 (BM25 + Transformer). Only around 24% of the answers were that Set 2 (BM25) is a good recommendation set. While 17% of the answers were that both sets look similar to the participant. This result confirms that the proposed “BM25 + USE (Transformer)” could effectively capture the semantic relatedness between research paper titles.

In the user study, we also provided the tester with an option to view the research paper abstract corresponding to the selected paper title. We gathered an information on the perceived utility of showing abstracts. In more details, we have tracked how many clicks on abstracts were performed in order to collect

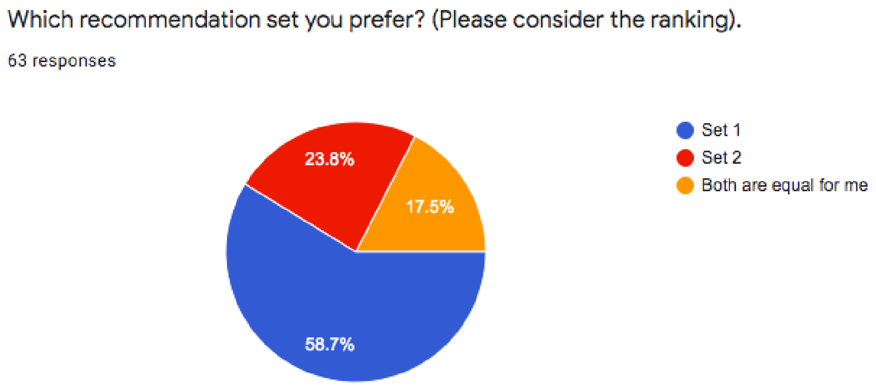


Figure 6.14: Results of the user study.

statistics of how many users decided to view an abstract. We found out that only 1.6% of the displayed paper abstracts were viewed. Which means that the title in this case was enough to understand the recommendation provided and decide about the recommendation set.

6.5 Summary

In this chapter, we experimented five pre-trained sentence encoders for reranking research paper recommendations. Our results show that the sentence encoders – which perform so well in other domains – are not performing well in the domain of research paper recommendations. When combined in a hybrid approach with BM25, they performed better than BM25 alone or any of the encoders alone. However, the improvement of up to 7.35% is still small compared to the exceptional results sentence encoders achieved in other domains.

Chapter 7

Conclusions

7.1 Summary of Results

This thesis has focused on investigating possible deep learning techniques that may capture semantic textual similarity between research papers, in order to develop more accurate and effective recommender systems. Since tagging is one of the ways to organize, find and recommend research papers, we proposed a tag predection approach for research papers. This approach has adopted a supervised deep recurrent network with attention mechanism, in order to focus on important words in paper titles and abstracts, during the feature extraction phase. Then, we exploited the same model trained for tag prediction in extracting semantic and textual representations of research papers. We incorporated those representations into a matrix factorization technique, in order to obtain personalized research paper recommendations. Finally, we experimented five well-known pre-trained sentence encoders that claimed to capture generic semantic representations. The encoders were experimented in the pipeline of a non-personalized research paper recommender system that is based on Apache Lucene BM25 ranking function

Overall, the main findings of the research introduced in this thesis can be summarized as follows:

1. In the research paper domain, traditional techniques such as TF-IDF still performs quite good compared to most of the word embeddings based techniques, due to the lack of scientific and multi-domain vocabulary in word embedding models.

2. The proposed approach for representing research paper titles and abstracts, using bidirectional GRUs with attention mechanism, could effectively focus on important words and sentences, and it led to better tag prediction accuracy compared to TF-IDF and state-of-the-art techniques in text feature extraction. In this study, we also showed that using titles concatenated with abstracts in this scenario yields better performance compared to using only the titles.

3. Social tags as metadata can enhance the performance of research paper recommendation. We showed that tags can also be utilized for extracting semantic document representations, using the tag prediction model. And incorporating those representations with few number of ratings, can enhance the recommendations. A comparison was made, which showed the difference between using matrix factorization (MF) with tags or documents TF-IDF as metadata, and using our proposed model of tag prediction to extract text features as metadata. The comparison showed an improved performance of the proposed technique.

4. Traditional search engine based techniques such as the ones based on

Apache Lucene/Solr BM25 ranking function, still gives better recommendation performance for non-personalized research paper recommendation, in comparison to state-of-the-art pre-trained sentence encoders such as USE, BERT, ELMo, InferSent and SciBERT. Differently, the integration of these encoders may enhance the ranking of recommendations.

7.2 Future Directions

There are several possibilities to extend the research presented in this thesis. Among the others,

- For the tag prediction work, more extensive experiments can be conducted to further evaluate the performance of the proposed approach on larger number of tags. In addition, the proposed approach could also be validated for research paper keyword prediction. Moreover, considering the hierarchical structure of tags could also be investigated.
- For the tag-aware research paper recommendation, possible extension could be to study how to select the number of top tags that could effectively train a model, which can further be used to extract research paper representations. Another possible future work is to explore more advanced MF techniques in this scenario rather than WARP.
- For the proposed technique of reranking recommendations using sentence encoders, a limitation of our research is that we only worked with the titles of papers. An extension could be to include the abstracts. However, this will probably further increase the running time, and hence decrease the applicability of sentence encoders in systems that require

recommendations in real-time. Another possible extension is to fine-tune the sentence encoders with annotated datasets. However, unfortunately, research paper domain lacks such kind of paper similarity and relatedness annotation. Finally, one interesting direction would be exploring meta-learning techniques that could automatically select or integrate different sentence encoders based on the domains of research papers.

Publications

1. Hebatallah A. Mohamed Hassan, Giuseppe Sansonetti, Fabio Gasparetti, and Alessandro Micarelli. BERT, ELMo, USE and InferSent sentence encoders: The panacea for research paper recommendation? In Proceedings of the 13th ACM Conference on Recommender Systems Late-Breaking Results, pages 6–10. ACM, 2019.

2. Hebatallah A. Mohamed Hassan, Giuseppe Sansonetti, Fabio Gasparetti, and Alessandro Micarelli. Semantic-based tag recommendation in scientific bookmarking systems. In Proceedings of the 12th ACM Conference on Recommender Systems, pages 465–469. ACM, 2018.

3. Hebatallah A. Mohamed Hassan. Personalized research paper recommendation using deep learning. In Proceedings of the 25th ACM Conference on User Modeling, Adaptation and Personalization, pages 327–330. ACM, 2017.

Bibliography

[AARK16] Titipat Achakulvisut, Daniel E Acuna, Tulakan Ruangrong, and Konrad Kording. Science concierge: A fast content-based recommendation system for scientific publications. *PloS one*, 11(7):e0158423, 2016.

[AAUE17] Modhi Al Alshaikh, Gulden Uchyigit, and Roger Evans. A research paper recommender system using a dynamic normalized tree of concepts model for user modelling. In *2017 11th International Conference on Research Challenges in Information Science (RCIS)*, pages 200–210. IEEE, 2017.

[APSF16] Maha Amami, Gabriella Pasi, Fabio Stella, and Rim Faiz. An lda-based approach to scientific paper recommendation. In *International conference on applications of natural language to information systems*, pages 200–210. Springer, 2016.

[ATY$^+$19] Md Zahangir Alom, Tarek M Taha, Chris Yakopcic, Stefan

Westberg, Paheding Sidike, Mst Shamima Nasrin, Mahmudul Hasan, Brian C Van Essen, Abdul AS Awwal, and Vijayan K Asari. A state-of-the-art survey on deep learning theory and architectures. *Electronics*, 8(3):292, 2019.

[AZ05] Rie Kubota Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853, 2005.

[BABG17] Joeran Beel, Akiko Aizawa, Corinna Breitinger, and Bela Gipp. Mr. dlib: recommendations-as-a-service (raas) for academia. In *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 1–2. IEEE, 2017.

[BAG17] Fabiano M Belém, Jussara M Almeida, and Marcos A Gonçalves. A survey on tag recommendation methods. *Journal of the Association for Information Science and Technology*, 68(4):830–844, 2017.

[BB08] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pages 161–168, 2008.

[BBM16] Trapit Bansal, David Belanger, and Andrew McCallum. Ask the gru: Multi-task learning for deep text recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 107–114. ACM, 2016.

[BCB14] D. Bahdanau, K. Cho, and Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *ArXiv e-prints*, September 2014.

[BCL19] Iz Beltagy, Arman Cohan, and Kyle Lo. Scibert: Pretrained contextualized embeddings for scientific text. *arXiv preprint arXiv:1903.10676*, 2019.

[BCV13] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[Ber15] Mark J. Berger. Large Scale Multi-Label Text Classification with Semantic Word Vectors. Technical report, Stanford University, 2015.

[BGJM17] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[BGLB16] Joeran Beel, Bela Gipp, Stefan Langer, and Corinna Breitinger. Research paper recommender systems: A literature survey. *International Journal on Digital Libraries*, (4):305–338, 2016.

[BGO19] J Beel, A Griffin, and C O'Shea. Darwin & goliath: Recommendations-as-a-service with automated algorithm-selection and white-labels. In *Proceedings of the 13th ACM Conference on Recommender Systems*, 2019.

[BHP17] Jean-Pierre Briot, Gaëtan Hadjeres, and François Pachet. Deep learning techniques for music generation-a survey. *arXiv preprint arXiv:1709.01620*, 2017.

[BLGN13] Joeran Beel, Stefan Langer, Marcel Genzmehr, and Andreas Nürnberger. Introducing docear's research paper recommender system. In *JCDL*, pages 459–460, 2013.

[BNJ03] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[BVdB08] Toine Bogers and Antal Van den Bosch. Recommending scientific articles using citeulike. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 287–290. Citeseer, 2008.

[CB19] Andrew Collins and Joeran Beel. Document embeddings vs. keyphrases vs. terms: A large-scale online evaluation in digital library recommender systems. In *Proceedings of the ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL)*, 2019.

[CKS$^+$17] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*, 2017.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[CVMG+14] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[CYK+18] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

[DC19] Zhuyun Dai and Jamie Callan. Deeper text understanding for ir with contextual neural language modeling. *arXiv preprint arXiv:1905.09217*, 2019.

[DCFLHRM10] Luis M De Campos, Juan M Fernández-Luna, Juan F Huete, and Miguel A Rueda-Morales. Combining content-based and collaborative recommendations: A hybrid approach based on bayesian networks. *International Journal of Approximate Reasoning*, 51(7):785–799, 2010.

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[DL15] Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079–3087, 2015.

[DNT14] Dario De Nart and Carlo Tasso. A personalized concept-driven recommender system for scientific libraries. *Procedia Computer Science*, 38:84–91, 2014.

[ER04] Günes Erkan and Dragomir R Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479, 2004.

[FGS98] Paolo Frasconi, Marco Gori, and Alessandro Sperduti. A general framework for adaptive processing of data structures. *IEEE transactions on Neural Networks*, 9(5):768–786, 1998.

[FPT11] Felice Ferrara, Nirmala Pudota, and Carlo Tasso. A keyphrase-based paper recommender system. In *Italian research conference on digital libraries*, pages 14–25. Springer, 2011.

[GBH09] Bela Gipp, Jöran Beel, and Christian Hentschel. Scienstein: A research paper recommender system. In *Proceedings of the international conference on emerging trends in computing (icetic'09)*, pages 309–315, 2009.

[GLM07] Alexander Genkin, David D Lewis, and David Madigan. Large-scale bayesian logistic regression for text categorization. *Technometrics*, 49(3):291–304, 2007.

[GM09] Evgeniy Gabrilovich and Shaul Markovitch. Wikipedia-based semantic interpretation for natural language processing. *Journal of Artificial Intelligence Research*, 34:443–498, 2009.

[GPAM$^+$14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[GSC18] Francesco Gargiulo, Stefano Silvestri, and Mario Ciampi. Deep convolution neural network for extreme multi-label text classification. In *HEALTHINF*, pages 641–650, 2018.

[Has17] Hebatallah A Mohamed Hassan. Personalized research paper recommendation using deep learning. In *Proceedings of the 25th conference on user modeling, adaptation and personalization*, pages 327–330. ACM, 2017.

[HBF$^+$01] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[HBH$^+$15] Damien Hanyurwimfura, Liao Bo, Vincent Havyarimana, Dennis Njagi, and Faustin Kagorora. An effective academic research papers recommendation for non-profiled users. *International Journal of Hybrid Information Technology*, 8(3):255–272, 2015.

[HID$^+$17] Khalid Haruna, Maizatul Akmar Ismail, Damiasih Damiasih, Joko Sutopo, and Tutut Herawan. A collaborative approach for research paper recommender system. *PloS one*, 12(10):e0184516, 2017.

[Hin09] Geoffrey E Hinton. Deep belief networks. *Scholarpedia*, 4(5):5947, 2009.

[HKL17] Beomseok Hong, Yanggon Kim, and Sang Ho Lee. An efficient tag recommendation method using topic modeling approaches. In *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, RACS '17, pages 56–61, New York, NY, USA, 2017. ACM.

[HKTR04] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[HKV08] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272. Ieee, 2008.

[HS97a] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[HS97b] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[HSGM18] Hebatallah A Mohamed Hassan, Giuseppe Sansonetti, Fabio Gasparetti, and Alessandro Micarelli. Semantic-based tag recommendation in scientific bookmarking systems. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 465–469. ACM, 2018.

[HSGM19] Hebatallah A Mohamed Hassan, Giuseppe Sansonetti, Fabio Gasparetti, and Alessandro Micarelli. BERT, ELMo, USE and InferSent sentence encoders: The panacea for research-paper recommendation? In *Proceedings of the 13th ACM Conference on Recommender Systems Late-Breaking Results*, pages 6–10. ACM, 2019.

[HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[IHJS07] Jens Illig, Andreas Hotho, Robert Jäschke, and Gerd Stumme. A comparison of content-based tag recommendations in folksonomy systems. In *Knowledge Processing and Data Analysis*, pages 136–149. Springer, 2007.

[Joa98] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

[JWY$^+$19] Hamed Jelodar, Yongli Wang, Chi Yuan, Xia Feng, Xiahui Jiang, Yanchao Li, and Liang Zhao. Latent dirichlet allocation (lda) and topic modeling: models, applications, a survey. *Multimedia Tools and Applications*, 78(11):15169–15211, 2019.

[KAP15] Aris Kosmopoulos, Ion Androutsopoulos, and Georgios Paliouras. Biomedical semantic indexing using dense word vectors in bioasq. *J BioMed Semant Suppl BioMedl Inf Retr*, 3410:959136040–1510456246, 2015.

[KBV09] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[KF09] Ralf Krestel and Peter Fankhauser. Tag recommendation using probabilistic topic models. In *Proceedings of the 2009th International Conference on ECML PKDD Discovery Challenge - Volume 497*, ECMLPKDDDC'09, pages 131–141, Aachen, Germany, Germany, 2009. CEUR-WS.org.

[KFN09] Ralf Krestel, Peter Fankhauser, and Wolfgang Nejdl. Latent dirichlet allocation for tag recommendation. In *Proceedings of the third ACM Conference on Recommender Systems*, RecSys '09, pages 61–68, New York, NY, USA, 2009. ACM.

[KGB14] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[Kim14] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[KMTD14] Mikael Kågebäck, Olof Mogren, Nina Tahmasebi, and Devdatt Dubhashi. Extractive summarization using continuous vector space models. In *Proceedings of the 2nd Workshop on*

Continuous Vector Space Models and their Compositionality (CVSC), pages 31–39, 2014.

[KN06] Jonathan Koberstein and Yiu-Kai Ng. Using word clusters to detect similar web documents. In *International Conference on Knowledge Science, Engineering and Management*, pages 215–228. Springer, 2006.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[Kul15] Maciej Kula. Metadata embeddings for user and item cold-start recommendations. *arXiv preprint arXiv:1507.08439*, 2015.

[LCWY17] Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 115–124, New York, NY, USA, 2017. ACM.

[LD97] Thomas K Landauer and Susan T Dumais. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211, 1997.

[LLK13] Joonseok Lee, Kisung Lee, and Jennifer G Kim. Personalized academic research paper recommendation system. *arXiv preprint arXiv:1304.5457*, 2013.

[LM14] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014.

[LPM15] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[LYLZ14] Shujie Liu, Nan Yang, Mu Li, and Ming Zhou. A recursive recurrent neural network for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1491–1500, 2014.

[MBBF00] Llew Mason, Jonathan Baxter, Peter L Bartlett, and Marcus R Frean. Boosting algorithms as gradient descent. In *Advances in neural information processing systems*, pages 512–518, 2000.

[MKKG16] Shahin Mohammadi, Sudhir Kylasa, Giorgos Kollias, and Ananth Grama. Context-specific recommendation system for predicting similar pubmed articles. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pages 1007–1014. IEEE, 2016.

[MLA14] Kun Ma, Tingting Lu, and Ajith Abraham. Hybrid parallel approach for personalized literature recommendation system.

In *2014 6th International Conference on Computational Aspects of Social Networks*, pages 31–36. IEEE, 2014.

[MSC$^+$13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[Nie15] Michael A Nielsen. *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, USA:, 2015.

[NKM$^+$14] Jinseok Nam, Jungi Kim, Eneldo Loza Mencía, Iryna Gurevych, and Johannes Fürnkranz. Large-scale multi-label text classification — revisiting neural networks. In *Proceedings of the 2014th European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part II*, ECMLPKDD'14, pages 437–452, Berlin, Heidelberg, 2014. Springer-Verlag.

[NLdSG11] Cristiano Nascimento, Alberto HF Laender, Altigran S da Silva, and Marcos André Gonçalves. A source independent framework for research paper recommendation. In *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries*, pages 297–306. ACM, 2011.

[PB07] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[PGJ17] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507*, 2017.

[PKCK12] Deuk Hee Park, Hyea Kyeong Kim, Il Young Choi, and Jae Kyeong Kim. A literature review and classification of recommender systems research. *Expert systems with applications*, 39(11):10059–10072, 2012.

[PN11] Maria Soledad Pera and Yiu-Kai Ng. A personalized recommendation system on scholarly publications. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 2133–2136. ACM, 2011.

[PNI$^+$18] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

[PSM14a] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[PSM14b] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[PTDU16] Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2249–2255, 2016.

[PVG$^+$11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[RCW15] Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015.

[RFGST09] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press, 2009.

[RHW86] David E. Rumelhart, Geoff E. Hinton, and R. J. Wilson. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[RWJ$^+$95] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. Okapi at trec-3. *Nist Special Publication Sp*, 109:109, 1995.

[S$^+$01] Amit Singhal et al. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43, 2001.

[SB88] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[Sch15] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[SK13] Kazunari Sugiyama and Min-Yen Kan. Exploiting potential citation papers in scholarly paper recommendation. In *Proceedings of the 13th ACM/IEEE-CS joint conference on Digital libraries*, pages 153–162. ACM, 2013.

[SLMN11] Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 129–136, 2011.

[SP97] M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681, November 1997.

[SPW$^+$13] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on* *empirical methods in natural language processing*, pages 1631–1642, 2013.

[SZG11] Yang Song, Lu Zhang, and C Lee Giles. Automatic tag recommendation algorithms for social recommender systems. *ACM Transactions on the Web (TWEB)*, 5(1):4, 2011.

[TLPP$^+$14] Álvaro Tejeda-Lorente, Carlos Porcel, Eduardo Peis, Rosa Sanz, and Enrique Herrera-Viedma. A quality based recommender system to disseminate information in a university digital library. *Information Sciences*, 261:52–69, 2014.

[VDDP18] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.

[VLL$^+$10] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(Dec):3371–3408, 2010.

[WB11] Chong Wang and David M Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 448–456. ACM, 2011.

[WBGL16] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Charagram: Embedding words and sentences via character n-grams. *arXiv preprint arXiv:1607.02789*, 2016.

[WCdML16] Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. Relation classification via multi-level attention cnns. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1298–1307. Association for Computational Linguistics, 2016.

[WCL13] Hao Wang, Binyi Chen, and Wu-Jun Li. Collaborative topic regression with social regularization for tag recommendation. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[WSW16] Ian Wesley-Smith and Jevin D West. Babel: a platform for facilitating research in scholarly article discovery. In *Proceedings of the 25th international conference companion on world wide web*, pages 389–394. International World Wide Web Conferences Steering Committee, 2016.

[WWY15] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1235–1244. ACM, 2015.

[WYW13] Jason Weston, Hector Yee, and Ron J Weiss. Learning to rank recommendations with the k-order statistic loss. In *Pro-*

ceedings of the 7th ACM conference on Recommender systems, pages 245–248. ACM, 2013.

[XBK$^+$15] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In Francis R. Bach and David M. Blei, editors, *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2048–2057. JMLR.org, 2015.

[XMS16] Caiming Xiong, Stephen Merity, and Richard Socher. Dynamic memory networks for visual and textual question answering. In *International conference on machine learning*, pages 2397–2406, 2016.

[YHL11] Hsiang-Fu Yu, Fang-Lan Huang, and Chih-Jen Lin. Dual coordinate descent methods for logistic regression and maximum entropy models. *Machine Learning*, 85(1-2):41–75, 2011.

[YKYS17] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*, 2017.

[YM07] Takashi Yoneya and Hiroshi Mamitsuka. Pure: a pubmed article recommendation system based on content-based filtering. *Genome informatics*, 18:267–276, 2007.

[YSXZ15] Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. ABCNN: attention-based convolutional neural network for modeling sentence pairs. *CoRR*, abs/1512.05193, 2015.

[YYD$^+$16] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. Hierarchical attention networks for document classification. In *HLT-NAACL*, 2016.

[YZL19] Wei Yang, Haotian Zhang, and Jimmy Lin. Simple applications of bert for ad hoc document retrieval. *arXiv preprint arXiv:1903.10972*, 2019.

[ZJYH03] Jian Zhang, Rong Jin, Yiming Yang, and Alexander G Hauptmann. Modified logistic regression: An approximation to svm and its applications in large-scale text categorization. In *ICML*, volume 3, pages 888–895, 2003.