Contextual word embedding for multi-label classification task of scientific articles

Yi Ning Liang
University of Amsterdam
12076031

ABSTRACT

In Natural Language Processing (NLP), using word embeddings as input for training classifiers has become more common over the past decade. Training on a lager corpus and more complex models has become easier with the progress of machine learning techniques. However, these robust techniques do have their limitations while modeling for ambiguous lexical meaning. In order to capture multiple meanings of words, several contextual word embeddings methods have emerged and shown significant improvements on a wide rage of downstream tasks such as question answering, machine translation, text classification and named entity recognition. This study focuses on the comparison of classical models which use static representations and contextual embeddings which implement dynamic representations by evaluating their performance on multi-labeled text classification of scientific articles. Ten experiments are designed and implemented using different embedding methods to train a classifier and the results demonstrate that contextual word embeddings have outperformed the distributed word embeddings. Furthermore, the model that was fine-tuned on a scientific corpus has achieved better results than the one fine-tuned on general domain collections or other models that use pretrained embeddings.

KEYWORDS

contextual word embedding, text classification, scientific articles. multi-labeled classification

1 INTRODUCTION

Word representation learning [1] has been one of the main research areas in Semantics of the NLP domain and has proven to be effective in downstream applications. The main idea here is that every word can be transferred to a N-dimensional vector where similar words will have closer proximity to each other. The word representations are learned by exploiting a large amount of text corpora, most popular classic word embedding methods include word2vec, GloVe and fastText. Unlike count based method such as Bag-of-word and Tf-idf vectors which only take the frequencies into account, these methods use distributed vector representations that consider the local word order in sequence and try to capture the semantics meaning and similarity of words. These methods will be referred to classical models in this thesis.

However, despite their robustness and ability to understand semantic properties of words, distributed representations methods are not able to disambiguate accurate word senses in different content and fail to capture polysemy of words. A word can have one or multiple meanings, and different words also often have very similar or overlapping meanings. For instance, the word *bank* can refer to two different meanings according to the text: a financial institution or the land along the edge of a river. Hence, the word sense of *bank* is ambiguous and it shares the same representation regardless of the context it appears in.

In order to deal with the ambiguity of words, several methods have been proposed to model the individual meaning of words and an emerging branch of study has focused on directly integrating the unsupervised embeddings into downstream applications. These methods are known as contextualized word embedding. Most of the prominent contextual representation models implement a bidirectional language model (biLM) that concatenates the output vector of the left-to-right output vector and right-to-left using a LSTM model [2–5] or transformer model [6, 7].

The contextualized word embeddings have achieved state-of-the-art results on many tasks from the major NLP benchmarks with the integration of the downstream task. Nevertheless, these fine-tuned (please refer to section 2.3) experiments are mostly trained on supervised data, the fine-tuned method using a pretraining objective hasn't yet been thoroughly tested. Furthermore, the pretrained models are mostly trained on a large corpus of general domain. We investigate whether the model trained on the specific domain corpus can improve the multi-label classification of scientific articles provided by Elsevier. Our research questions are:

- RQ1 *What is the impact on the performance metrics (please refer to section 4.3) of contextual word embeddings compared to classical word embedding?*
- RQ2 *How do embedding models trained on different domains compared with each other in terms of performance metrics?*

The method of this research consists of literature review, analysis and experiments. Firstly, related literature of word embedding algorithms and multi-label classification are discussed. This is followed by a description of the selected models used for the experiments. Then, the exploratory data analysis and data preprocessing steps are covered. In order to truly understand the difference between distributed word embedding and contextualized word embedding, we conduct the experiments using both pretrained embedding and fine-tuning on own data. For answering RQ1, BERT is fine-tuned with an unsupervised method and compared with other classical models by the classification task. Then, we fine-tuned BERT and SciBERT (a variant of the BERT model pretrained on scientific articles) with a supervised approach. The evaluation results lead us to the answer of RQ2.

2 RELATED WORK

This section discusses previous study done on multi-label classification learning (2.1), distributed word representations (2.2), and contextualized word embeddings (2.3).

2.1 Multi-label classification

In general, the classification task in machine learning aims to explore knowledge that can be used to predict the unknown class of an instance based on the features of the input instances. A classification problem can be categorized into two groups: single-label classification and multi-label classification. Single classification tasks concerned with learning from a set of examples that are associated with a single label from a set of disjoint labels L [8]. In multi-label classification, the examples can be associated with two or more concept labels. In other words, each instance belongs to a subset of classes from L. Over the past years, multi-label classification tasks appears in a wide range of real-world situations and applications, and the existing multi-label classification methods can be grouped into three categories: a) problem transformation methods, b) algorithm adaptation methods and c) Ensemble methods [9, 10]. Problem transformation methods transform the multi-label classification problem into either one or more single-label classification or regression problems, and an algorithm adaptation approach aims to extend specific learning algorithms in order to handle multi-label data directly without requiring any preprocessing. The ensemble based multi-label classifiers are developed on top of the problem transformation and algorithm adaptation methods.

2.2 Distributed Word Representations

For capturing the semantic and syntactic regularities of words in an NLP task, it is important that we know which words or expressions can be used in a similar way based on a certain content. One of the earliest work dates back to Rumelhart et al.,1986 [11], who use the statistical language modeling to generate representations from large unlabeled corpora by using the preceding words to predict the next word in a sentence. The other approach to derive features is by grouping together the words which appear in the same content. For example, the clustering model by Brown et al. (1992) [12] automatically categorizes words in a large corpus based on their neighboring words. More recently, neural network models had been proposed for modeling the probability distribution of the next word prediction. These models were first introduced in Bengio et al.,2003 [13] and Bengio et al., 2006 [14] who use the concatenation of previous word vectors as an input representation of a feed-forward neural model and aims to predict the next word in the sequence. Each word trained in the model can be mapped into a low dimensional vector where semantically similar words have similar vector representations such as “increase” and “grow”. Mikolov et al., 2013 [15] proposed two new model architectures: Continuous Bag-Of-Words (CBOW) and Skip-gram, which was popularized as the word2vec package that can be used for learning high-quality word vectors from larger corpus. In the skip-gram model, surrounding words are predicted based on the source word given, while in the CBOW model, it predicts the target word according to its content. Quoted directly from the w2v paper: ’Somewhat surprisingly, these questions can be answered by performing simple algebraic operations with the vector representation of words’. A famous example using arithmetic operations is as follow: *vector(”King”) - vector(”Man”) + vector(”Woman”) = vector(”Queen”)*.

Most of the approaches generate the vector on the word level without parameter sharing. However, the internal structure of words is ignored and pose limitation of morphology for rich languages. A new branch emerged to leverage the word vector on subwords units, which is a word can be split into arbitrary character sequences and encode into n-gram vector [16–18].

2.3 Contextualized Word Embeddings

In the distributed word representation, each word type is projected as a single point in the semantic space no matter which content it appears in. However, in the real world, words tend to have different meanings in different contexts. Capturing the multiple meanings of a word and conflates all the meaning into one single vector has become the biggest limitation in word embedding learning. In order to adapt to dynamic polysemy language model, contextualized word embedding has emerged to analyze the context of the target word and generate its dynamic embedding. Context2vec [2] is one of the earliest work that employ contextualized representations. Inspired by the word2vec’s CBOW architecture, Context2vec proposes a bi-directional LSTM language model to replace the averaged word embeddings in a fixed window and contributes to crucial foundation for a lot of subsequent research. In the case of a unidirectional model, a word representation is generated from left to right. For example, in the sentence “I have the bank account”, the model would represent “bank” based on “I have the” but not “account". As for the bidirectional contextual model, the "bank" word would be represented using both its previous and following context. More computationally efficient models have been proposed for language modeling later on including a feed forward self-attention method for machine translation which is also known as Transformer [19] and Embeddings from Language Models (ELMo) [4] where each token is the concatenation of the left-to-right and right-to-left representations.

The methods for training the word embedding can be categorized into two groups: pretrained methods and fine-tune methods. Pretrained embeddings can be obtained by training a large corpus of unsupervised data which is usually time consuming and requires lots of computation power. In order to integrate to several downstream task easily, a fine-tune method has emerged that utilizes the pretrained embedding to initialize the model and fine-tunes on domain specific representations or a supervised downstream task. The advantage of this approach is that few parameters need to be learned from scratch. Due to this progress, several models including ELMo, OpenAI GPT [20], ULMFiT [5] and BERT [6] have achieved state-of-the-art results on many sentence level tasks from the major NLP benchmarks.

3 MODELS

This section describes the details of selected classical models (3.1) and contextual embedding models (3.2).

3.1 Classical models

3.1.1 Tf-idf model. Tf-idf (Term frequency-inverse document frequency) is an popular method in information retrieval to determine how important a word is to a document. The term frequency simply means how many times a word occurred in the document and inverse document frequency evaluates how much information the word carries, which tends to mitigate the importance of a term

if it tends to appear in many documents. A simple classifier can be created by treating each text as a vector of word counts and multiply the Tf-idf weights of each word. This method only takes the word count into consideration instead of the word order and the surrounding content. There are various methods to determine the exact values of both statistics, in this case we use sklearn [1], an open source machine learning package in python to calculate the Tf-idf scores. The equation is defined as following:

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \cdot (\log \frac{N}{\text{df}_t} + 1) \quad (1)$$

whereby N is the total number of documents in the corpus, $tf - idf_{t,d}$ is the frequency of term t in document d, and df_t denotes the number of documents where term t appears.

3.1.2 Word2vec model. Word2vec is a shallow, two-layer neural net that processes linguistic contexts of words. It takes large text corpus as input and reconstruct a vector space with each word token to a N-dimensional vector in the semantic space. The vectors are created using distributed numerical representations of word features where similar words are located in close proximity each other in the vector space. Word2vec consists of two architectures: Continuous Bag-of-Words model (CBOW) and Skip-gram model. The Continuous Bag-of-Words which has an architecture similar to feed-forward neural network language model but with the non-linear hidden layer removed and the projection layer shared for all words. The model predicts the current word according to a window of surrounding context and the order of context words doesn't influence prediction. However, unlike standard bag-of-words model, continuous distributed representation is used of the context. The model architecture is shown at Figure 1. For all word positions, the weight between the input and the projection layer is shared then the embedding is averaged before feeding into the objective function.

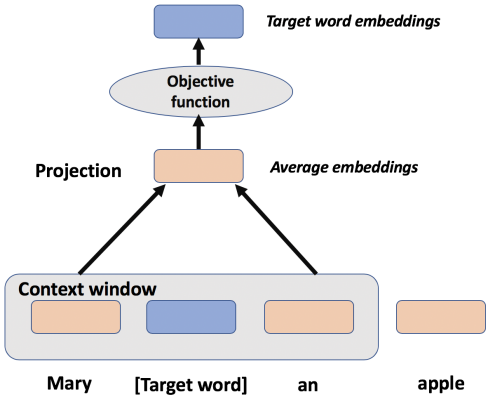


Figure 1: Model architecture of CBOW

Instead of predicting the target word based on its surrounding context, skipgram model predicts the context given a word. In addition, more distant words are given less weight by randomly sampling them. The model architecture is shown at Figure 2.

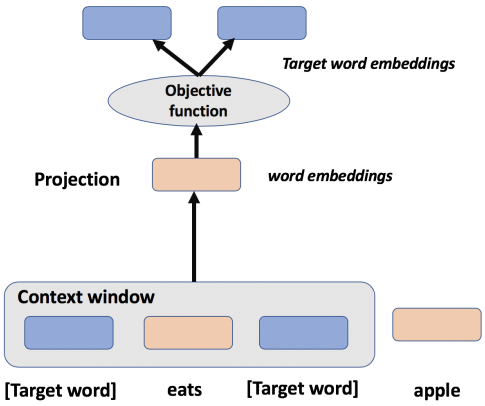


Figure 2: Model architecture of skipgram

3.1.3 fastText model. Inspired by the work of Mikolov et al.,2013 [15] and Levy et al.,2015 [21], fastText[2] is a library provided by Facebook's AI Research (FAIR) lab for learning of word representations [17] and text classification [22]. It supports continuous bag of words (CBOW) and Skip-gram models to train a word embedding model. Taking word order into consideration, fastText uses n-grams as additional features to capture the information of the local word order and also provides a solution for Out-Of-Vocabulary (OOV) problems. Word2vec and Glove fail to give vector representation if the word doesn't exist in the dictionary, however a word can be split into character n-grams in fastText, utilizing these can give more meaningful vectors for morphologically similar words, e.g., compact v.s. compactification. These words would have overlapping n-grams and would be embedded closer to each other. Additionally the "ion" has linguistic meaning which can also be captured by taking this approach. And for most sentences, it is possible to reconstruct the order of the words by just looking at a bag of n-grams. A word is represented as the sum of the n-gram vectors, and the text representation is obtained by averaging the word representations in the document. The classifier architecture in fastText is similar to the CBOW model, where the middle word is replaced by a label. For classifying a text, fastText implements a simple linear model with rank constraint, which makes the parameters sharing among features and classes possible and increase the generalization power of the context where some classes have very few examples. The loss function for a set of N documents in fastText is defined as followed:

$$-\frac{1}{N}\sum_{n=1}^{N} y_n \cdot \log(f(BAx_n)) \quad (2)$$

where f is the softmax function that computed the probability distribution over the predefined classes, y_i is the label, x_i is the normalized bag of features of the nth document, A is the lookup of

[1] https://scikit-learn.org/stable/

[2] https://fasttext.cc/

word to vector combined with averaging which is then passed to a linear model B.

3.2 Contextual models

3.2.1 BERT model. One of the latest milestones for contextual word embedding is the release of BERT, a deep bidirectional transformer model, in the late 2018. BERT builds upon several robust pre-training contextual embeddings including Semi-supervised Sequence Learning [23], Generative Pre-Training [7], ELMo [4], and ULMFit [5]. For standard language model, training a bidirectional sequence seems impossible since it would allow each word indirectly to "see itself" and the target word can be trivially predicted based on the context. Unlike these previous models, which looked at a text sequence either from left to right or combined left-to-right and right-to-left, BERT proposed a new language modeling method called *Masked language modeling* to achieve the deep bidirectional training. When a sequence of text is fed into the model, BERT tokenizes the words using WordPiece. WordPiece implements the greedy longest-match-first algorithm which is similar to byte pair encoding [24] that builds up a vocabulary of its subword units. An example for tokenizing the word "unaffable" using WordPiece would generate the following output: "un", "##aff", "##able", which also covers a wider spectrum of Out-Of-Vocabulary (OOV) words.

At pretrain time, after the sequence is tokenized into a list of words and subwords, around 15% of input tokens in each sequence are chosen at random, and 80% of those would be replaced with a special token [MASK], while 10% would remain unchanged and the rest 10% would be altered by a random token. Not masking all the chosen words is to mitigate the downside of mismatch between pre-training and fine-tuning method since the [MASK] token does not appear during fine-tuning. In order to improve the understanding of sentence relationships of the model, BERT pretrained a binarized next sentence prediction task along with the masked language model. While choosing sentences A and B for the prediction task, each pretraining example are designed that 50% of the time B is the actual next sentence that follows A. The masked sequence would first be embedded into vectors which is the sum of the token embeddings, the segmentation embeddings and the position embeddings and then processed in the Transformer encoder. A classification layer would be added on top of the encoder output, and by multiplying the output vectors to the embedding matrix would transform it into the vocabulary dimension. The probability of each word and next sentence is then calculated by the softmax function. The masked language model and next sentence prediction are trained together, with the goal of minimizing the combined loss function.

3.2.2 SciBERT. SciBERT [25] is a variant of BERT which focuses training on scientific articles instead of general domain corpora such as news articles and Wikipedia. The corpus of SciBERT consists of 18% papers from the computer science domain and 82% from the broad biomedical domain of 1.14M papers in total. The model is trained on the full text, not just the title and abstract. Considering the frequently occurring words might differ between scientific articles and general domain texts, SciBERT builds a new WordPiece vocabulary using SentencePiece library and sets the vocabulary size to about 30K. In the paper, the author reports the overlap of the token between BERT and SciBERT is 42%.

4 EXPERIMENT

This section describes the dataset used in the study (4.1), models designed to answer the research questions (4.2), and the model performance metrices used for multi-labeled evaluation (4.3).

4.1 Dataset

The dataset used has been provided by Elsevier's own ScienceDirect and Omniscience taxonomy that contains scientific content and topics of science of the article they belong to. ScienceDirect is a platform that enables users to search and access scientific articles of different domains and Omniscience taxonomy provides hierarchical relations of the science topics. For the classification task, the title and abstract are used to represent an article and associates with one or multiple topics it belongs to. In other words, an article can have more than one science topic which makes it a multi-labeled classification task.

In the dataset given, there are 1,031 classes in total and many of the classes only have few examples. In order to mitigate class imbalance effect, classes that have fewer than 500 examples are not considered. In Figure 3, the bar chart on top shows the class distribution of the original dataset which included classes that only have few examples and hence not significant enough to represent a class in the training dataset. The lower bar chart, the class distribution, is shown after dropping certain observations which makes the dataset more equally distributed.

There are two main characteristics for evaluating a multi-labeled dataset: label cardinality and label density [26]. The Label cardinality is the mean of the number of labels of the classes, defined by Eq. 3 and the label density is the mean of the number of labels of the classes divided by the number of classes in the dataset, defined by Eq. 4 .

$$Cardinality = \frac{1}{N}\sum_{i=1}^{N}|Y_i| \tag{3}$$

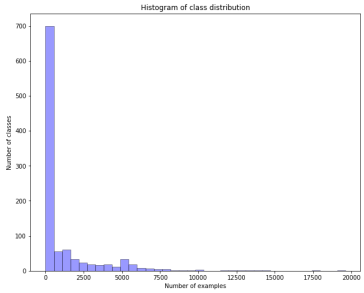
$$Density = \frac{1}{N}\sum_{i=1}^{N}\frac{|Y_i|}{|L|} \tag{4}$$

where N is the number of instances, Y_i is the set of labels of the ith instance and L is the set of total distinct labels in the dataset.

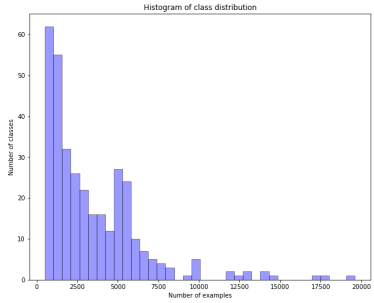
In table 1, two label statistics are recorded at two stages: Before and after the data is processed. Label cardinality slightly decreased after the data is processed since some observations are dropped due to the imbalanced nature of the dataset while the label density increased marginally due to the drastic decrease of number of classes, which is from 1,031 to 338 classes. In general, the measures can be used to evaluate differences between datasets, if the label cardinality is identical between two collections but with distinct label density can cause different behavior to the training classifier [26]. In our case, the properties of the data does not seem to vary too much before and after the processed stage.

	Original dataset	New dataset
Label cardinality	2.042	1.982
Label density	0.002	0.006

Table 1: Description of the label statistics before and after data processing



(a) Original dataset



(b) New dataset

Figure 3: class distribution for scientific articles

4.1.1 Data pre-processing. Data processing is a critical step that can reduce the complexity of the data and make it more understandable. The subsequent analysis could be better off through decreasing the incomplete and inconsistent nature of real-world data. For the scientific dataset, there is punctuation and other marks in the text that need to be removed. Firstly, the title and abstract are concatenated to one piece of text and punctuation like single/double quotes, spaces are normalized. Secondly, the copyright marks and the IEEE citation references are eliminated. Instead of performing stop word removal and stemming, we tried to keep the text as intact as possible. Removing stop words can be beneficial in some cases in NLP applications, however some semantic information might be lost and increase ambiguity which could weaken the performance of the contextual embedding learning. As for stemming, BERT uses WordPiece tokenization which tokenizes a string by its suffix (please refer to 3.2) which would be impossible if stemming is performed beforehand. The data is split into training and testing set by each class with 80% and 20% to make sure the two datasets have the same class distribution. The size of training and testing data is 416,334 and 168,458 respectively.

4.2 Model implementation details

4.2.1 Word2vec training. Three experiments are designed to evaluate the word2vector model. For the pretrained vector representation, Word2vec provide a pretrained model that was trained on Google News corpus with CBOW model, and the model contains approximately 3 billion words and phrase. And we also trained the CBOW and skipgram model on our own scientific data, with around 7 million words in the corpus. All models have 300-dimensional vectors with the context window set to 10 and minimum word occurrences of 5.

4.2.2 fastText training. The pretrained model [27] provided by fastText was trained on 1 million word vectors on Wikipedia 2017, UMBC webbase corpus and statmt.org news dataset (16B tokens) with CBOW model. In practice, training CBOW model is much faster than a skipgram model due to the nature of the architecture. However, skipgram is better for infrequent words since it is designed to predict the context of a target word so the common words and rare words are treated the same [3]. We selected a skipgram model for generating the embeddings on our own data. The embeddings are trained with bigram, the concatenation of 2 consecutive tokens or words, so that the model can trivially capture the local order in the sentence. On the character level of n-grams, minimum and maximum length of char n-gram is set to 3 and 6 respectively. Minimal number of word occurrences is set to 5.

4.2.3 BERT training. BERT provides two options to fine tune the classifier: BERT-Base and BERT-Large. Considering the computation time, we chose BERT-Based, which is a smaller model with 12 layer, 768 hidden size and 12 self-attention. Suggested by the paper, "uncased" model of BERT-Based model is selected which means the text has been lowercased before WordPiece tokenization and accent markers have been stripped out since the case information is not important for our task. The pretrained BERT-based uncased model provided by BERT was trained on BooksCorpus with about 800 million words and English Wikipedia with 2,500 million words. We fine-tuned BERT with both the unsupervised and the supervised method. All the parameters are set to default which are suggested by BERT in the fine-tuning stage, only the batch size has changed from 32 to 8 to deal with the out of memory error during training. We obtained the embedding by concatenating the last four hidden layers of the transformer since the method shows the highest F1 score compared to other approaches in the BERT paper. The BERT model is implemented by using the pytorch library [4] and run on a GPU with 16GB of RAM.

4.2.4 SciBERT training. SciBERT model is fine-tuned in a supervised approach in order to compare with the supervised BERT model. The purpose of this experiments is to figure out whether training on a different domain will affect the performance result so the parameters are set to the same as in fine-tuning BERT in order to have a fair comparison.

4.2.5 Classifier. In this case, we chose multinomial logistic regression (softmax regression) which is a generalization of logistic

[3] https://code.google.com/archive/p/word2vec/
[4] https://github.com/huggingface/pytorch-pretrained-BERT

regression for multi-class classification. Only the fastText experiments were trained on its own classifier, all the other experiments were trained on this classifier. Standard logistic regression can only do binary classification which assumes the labels are either 0 or 1. However, multinomial logistic regression allows one to handle multiple classes. The model predicts the probabilities of the different possible outcomes with a set of attributes and these class probabilities are summed up to one. With 338 labels in our dataset, training could take up a lot of time. Hence, the max iteration of classifier is set to 30 steps. That is, the model will converge at 30 epoch even if it doesn't reach its optimal stage. Due to time constraints and the goal of this studying isn't building a strong classifier, parameter tuning isn't applied.

4.3 Model Performance Metrics

Traditional single-label classification task is concerned with learning from examples that are related to a single label, while multi-label learning is concerned with examples that can be associated with multiple labels [28]. Therefore, the prediction in multi-label task can be fully correct, partially correct or fully incorrect and thus requires different metrics than those used in traditional single-label classification [10]. Evaluation measures for multi-label learning can be categorized into two groups: example based and label based. Example based tend to calculate the average difference between the predicted labels and the true labels for each test example, and then average over all examples in the test set in order to capture partially correct information. On the other hand, label based measures evaluate each label first and then averaged over all labels.

4.3.1 Example-based method. [29] In the definitions below, N is the number of examples in the dataset and Q is the number of classes. y_i denotes the set of true labels and x_i is the label predicted by the classifier.

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N} |\frac{x_i \cap y_i}{x_i \cup y_i}| \tag{5}$$

Accuracy computes the percentage of correctly classified labels among all predicted and true labels.

$$Precision = \frac{1}{N}\sum_{i=1}^{N} |\frac{x_i \cap y_i}{x_i}| \tag{6}$$

Precision calculates the percentage of correctly classified labels among all predicted labels.

$$Recall = \frac{1}{N}\sum_{i=1}^{N} |\frac{x_i \cap y_i}{y_i}| \tag{7}$$

Recall shows the percentage of correctly classified labels among all true labels.

$$F1 - measure = \frac{1}{N}\sum_{i=1}^{N} 2 \cdot \frac{|x_i \cap y_i|}{|x_i| + |y_i|} \tag{8}$$

F1-measure is the harmonic mean of precision and recall, it will penalizes classifiers with imbalanced precision and recall scores.

4.3.2 Label-based method. [29] Label-based metrics consider each label as a binary classifier and can be further divided into Macro-averaged and Micro-averaged measure. Here we chose to use Macro-averaged method to evaluate the model.

$$Macro - precision = \frac{1}{Q}\sum_{i=1}^{Q} |\frac{tp_i}{tp_i + fp_i}| \tag{9}$$

where tp_i and fp_i are the number of true positives and false positives for the label.

$$Macro - recall = \frac{1}{Q}\sum_{i=1}^{Q} |\frac{tp_i}{tp_i + fn_i}| \tag{10}$$

tp_i and fp_i are defined above and fn_i denotes the number of the false negatives for the label.

$$Macro - F1 - measure = \frac{1}{Q}\sum_{i=1}^{Q} \frac{2 \cdot tp_i}{2 \cdot tp_i + fp_i + fn_i} \tag{11}$$

In macro-averaged F1 measure average of the precision and recall for each set is calculated.

5 RESULTS

This section covers the results of the experiments mentioned in the model implementation details section and answers to the research questions mentioned in the introduction section.

5.1 Answer to RQ1

We conducted several experiments to compare classical embeddings and contextual embeddings. The classification performance for the different embeddings is shown in Table 2. In the first experiment, we built the Tf-idf model to compare the performance between the count-based method and the word embedding method. The result shows that tf-idf performs worse than most of the embedding methods. Tf-idf is an improved version of bag-of-words model and it does not capture the semantic meaning and the position of words. Word2vec, fastText and BERT provide pretrained embeddings that were trained on a general domain. The results indicate that Word2vec has the best result compared to fastText and BERT among pretrained embeddings. In this case, using the fastText classifier with a pretrained embedding performed worse than the tf-idf method. We also trained word2vec, fastText and BERT using our own dataset. The word2vec models were trained using two architectures: skipgram and CBOW. The fastText model was trained using skipgram and we fine-tuned the BERT model in an unsupervised way to learn the new embeddings. The word2vec skipgram model obtained the best result in this classification task and fastText with the pretrained embedding performed the worse. In the pretrained embeddings experiments, not only the model architectures are distinct, but also the dataset used is different. On the other hand, training BERT from scratch is not realistic due to the limitation of time and computation power, hence BERT can only be fine-tuned on the general domain dataset. We suspect that these are the most contributing factors to explain that contextual embeddings do not outperform the classical embedding while training

Model	Accuracy	Precision@1	Recall@1	F1-measure	Macro-precision	Macro-recall	Macro-F1-measure
Tf-idf	0.074	0.328	0.088	0.139	0.259	0.104	0.109
Word2vec pretrained embedding	0.115	0.489	0.131	0.207	0.401	0.168	0.180
fastText pretrained embedding	0.052	0.235	0.063	0.099	0.13	0.079	0.081
BERT pretrained embedding	0.081	0.353	0.093	0.147	0.272	0.118	0.204
Word2vec CBOW embedding	0.126	0.531	0.142	0.224	0.437	0.196	0.213
Word2vec skipgram embedding	0.133	0.554	0.148	0.234	0.455	0.201	0.217
fastText skipgram embedding	0.114	0.483	0.130	0.204	0.264	0.152	0.152
Unsupervised fine-tuning BERT	0.088	0.384	0.103	0.162	0.297	0.135	0.149
Supervised fine-tuning BERT	0.145	0.599	0.160	0.253	0.355	0.218	0.230
Supervised fine-tuning SciBERT	0.159	0.649	0.174	0.275	0.473	0.249	0.271

Table 2: Measurements of prediction quality on test data using different embedding

without a downstream task purpose. However further investigation is needed to verify these statements.

5.2 Answer to RQ2

The results in table 2 show that embeddings that are trained on the scientific dataset outperformed those using the pretrained embedding. The word2vec skipgram model, the fastText skipgram model and the unsupervised fine-tuned BERT model show an increase of the accuracy by 13.5%, 54% and 8% respectively. We also fine-tuned BERT which was trained on a large corpus of general domain data and SciBERT which was trained on scientific domain data to compare their performances. The accuracy and F1 score both increase by 8% for the supervised SciBERT model and it obtained the best result in our experiments. We can conclude that training the corpus on the scientific domain improves the performance of the classification task. Furthermore, associating the training progress with labels improves the performance significantly.

6 CONCLUSION

In this study, we generated a training dataset consisting of the title and abstract of scientific articles that can be used as input to a logistic regression classifier. We conducted experiments with different embeddings including tf-idf, word2vec, fastText and BERT. The performances of the models are then evaluated using various evaluation metrics. The results show that the classical embeddings outperformed unsupervised contextual embeddings in the classification task, and training a model on a corpus of the scientific domain improves the accuracy score. We have also shown that the classification task can be improved significantly by training supervised contextual embeddings. However, no conclusive result is obtained to explain why contextual embeddings performed worse in the unsupervised way. Further research would need to be done to reach a more comprehensive answer on the influence of the contextual word embeddings approach.

ACKNOWLEDGEMENT

I would like to take this opportunity to thank Elsevier for letting me work on this interesting project. In particular, I am grateful to Philip Tillman and Jan Jaap Meijerink for their constant advice and keen insight on how to improve the thesis. Furthermore, I would love to thank Anna Sepliarskaia for the helpful feedback and suggestions during the project.

REFERENCES

[1] J. Camacho-Collados and M. Taher Pilehvar. From word to sense embeddings: A survey on vector representations of meaning. *Journal of Artificial Intelligence Research 63*, pages 743–788, 2018.
[2] Oren Melamud, Jacob Goldberger, and Ido Dagan. context2vec: Learning generic context embedding with bidirectional lstm. pages 51–61, 01 2016.
[3] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. 2018.
[4] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *CoRR*, abs/1802.05365, 2018.
[5] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *CoRR*, abs/1801.06146, 2018.
[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.
[7] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.
[8] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3:1–13, 09 2009.
[9] Passent El Kafrawy, Amr Mausad, and Heba Esmail. Experimental comparison of methods for multi-label classification in different application domains. 2016.
[10] Mohammad S Sorower. A literature survey on algorithms for multi-label learning. 2010.
[11] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA, 1986.
[12] Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479, December 1992.
[13] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003.
[14] Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. Neural probabilistic language models. *Innovations in Machine Learning*, 194:pp 137–186, 2006.
[15] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781v3, 2013.

[16] Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramón Fermandez, Silvio Amir, Luís Marujo, and Tiago Luís. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[17] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *CoRR*, abs/1607.04606, 2016.

[18] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[20] Alec Radford. Improving language understanding by generative pre-training. 2018.

[21] Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. 2015.

[22] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *CoRR*, abs/1607.01759, 2016.

[23] Andrew M. Dai and Quoc V. Le. Semi-supervised sequence learning. *CoRR*, abs/1511.01432, 2015.

[24] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909, 2015.

[25] Iz Beltagy, Arman Cohan, and Kyle Lo. Scibert: Pretrained contextualized embeddings for scientific text. *CoRR*, abs/1903.10676, 2019.

[26] Flavia Bernardini, Rodrigo Barbosa da Silva, Rodrigo Magalhães Rodovalho, and Edwin Mitacc Meza. Cardinality and density measures and their influence to multi-label learning methods. *Learning and Nonlinear Models*, 12:53–71, 01 2014.

[27] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[28] Tsoumakas G. and Katakis I. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 2007.

[29] Passent El Kafrawy, Amr Mausad, and Heba Esmail. Experimental comparison of methods for multi-label classification in different application domains. *International Journal of Computer Applications*, 114, 2015.