REPUBLIC OF TURKEY

AKDENIZ UNIVERSITY

NEURAL NETWORK BASED PUBLICATION RECOMMENDER FOR ARTICLE SUBMISSION

Seth MICHAIL

INSTITUTE OF NATURAL AND APPLIED SCIENCES

DEPARTMENT OF COMPUTER ENGINEERING

MASTER THESIS

DECEMBER 2020

ANTALYA

REPUBLIC OF TURKEY

AKDENIZ UNIVERSITY

NEURAL NETWORK BASED PUBLICATION RECOMMENDER FOR ARTICLE SUBMISSION

Seth MICHAIL

INSTITUTE OF NATURAL AND APPLIED SCIENCES

DEPARTMENT OF COMPUTER ENGINEERING

MASTER THESIS

DECEMBER 2020

ANTALYA

REPUBLIC OF TURKEY

AKDENIZ UNIVERSITY

INSTITUTE OF NATURAL AND APPLIED SCIENCES

NEURAL NETWORK BASED PUBLICATION RECOMMENDER FOR ARTICLE SUBMISSION

Seth MICHAIL

DEPARTMENT OF COMPUTER ENGINEERING

MASTER THESIS

This thesis was accepted unanimously by the jury on 30/12/2020.

Asst. Prof. Dr. Joseph LEDET (Supervisor)

Prof. Dr. Melih GÜNAY

Asst. Prof. Dr. Asım Sinan YÜKSEL

ÖZET

MAKALE GÖNDERİMİ İÇİN SİNİR AĞ TABANLI YAYIN TAVSİYESİ

Seth MICHAIL

Yüksek Lisans Tezi, Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Asst Prof. Joseph Ledet

Aralik 2020; 46 sayfa

Bu tez ile yapılan çalışma sonucunda yazarlar makalelerini göndermek için en iyi dergiyi bulabilirler. Makalenin içeriğine uygun yayınlanacağı dergiyi tespit etme işi kolay değildir. Makalenin uygun dergide yayınlanmaması, çalışmanın hedef kitleye ulaşamamasına ve dolayısıyla etkisinin zayıf kalmasına neden olmaktadır. Dergilerin içeriğe göre sınıflandırılmasında, dergilerde yayınlanan makalelerin özetleri kullanılabilir. Böylece, dergi, özetler kullanılarak oluşturulan bir bakıma parmak iziyle karakterize edilmesi sağlanır.

Dergi sınıflandırma, bu çalışmada yapay sinir ağları kullanılarak gerçekleştirilmiştir. İlk adımda özetler, doğal dil işleme metotları kullanılarak vektör formatına çevrilir. Daha sonra oluşturulan bu vektör formatı, sinir ağındaki parametleri eğitmekte kullanılır. Bu tez kapsamında yapılan çalışmada, daha önce yapılan fakat detayı paylaşılmayan çalışmaların ötesine gidilerek Web of Science da yer alan tüm yayınlar ve yayıncıların dergileri kullanılmıtşr. Böylece elde edilen sonuçlar hem daha kapsayıcı, hem daha bol seçenekli ve hem de daha kararlı olmuştur.

ANAHTAR KELİMELER: BERT, DistilBERT, Doc2Vec, Masked Language Modelling, Natural Language Processing, Publication, Recommender System, Transformer

JÜRİ: Dr. Öğr. Üyesi Joseph LEDET

Prof. Dr. Melih GÜNAY

Dr. Öğr. Üyesi Asım Sinan YÜKSEL

ABSTRACT

NEURAL NETWORK BASED PUBLICATION RECOMMENDER FOR ARTICLE SUBMISSION

Seth MICHAIL

MSc Thesis in Computer Engineering
Supervisor: Asst. Prof. Dr. Joseph LEDET
December 2020; 46 pages

This thesis is about recommending the best journal to authors for article submission. This task is a challenge because of the need to ensure that the journal is relevant. The significance of the relevancy of a journal is that an article published in a less relevant journal will have less exposure to the intended target audience, and consequently have less of an impact. The conceptual content contained in the abstracts of articles accepted for publication in a journal can be used to characterize that journal, which can be thought of as a "finger print" or "signature" of the journal.

In turn, the content of these abstracts can be imprinted in the weights of neural network models. The abstracts are first converted to vector representations obtained through the methods of natural language processing, in order to be used with neural networks. The main purpose of this work will be exploring neural network architectures for discovering and recommending appropriate journals to those seeking to publish their research.

In this thesis, the current state of the art will be extended and a robust and generic article-to-journal matching tool for all publishers, using Web of Science data, will be proposed.

KEYWORDS: BERT, DistilBERT, Doc2Vec, Masked Language Modelling, Natural Language Processing, Publication, Recommender System, Transformer

COMMITTEE: Asst. Prof. Dr. Joseph LEDET
Prof. Dr. Melih GÜNAY
Asst. Prof. Dr. Asım Sinan YÜKSEL

ACKNOWLEDGEMENTS

I would like to thank my supervisor Asst. Prof. Dr. Joseph LEDET for his guidance
I would also like to thank Prof. Dr. Melih GÜNAY who gave many helpful insights.
I would like to thank all my teachers, because all of their lessons lead up to this thesis.
I am especially grateful to my family for their love and support.
Most importantly, I thank my loving wife, Serap. She is my sunshine, without which, there can be no rainbow.

LIST OF CONTENTS

TEXT OF OATH

I declare that this study "Neural Network Based Publication Recommender for Article Submission", which I present as master thesis, is in accordance with the academic rules and ethical conduct. I also declare that I cited and referenced all material and results that are not original to this work.

30/12/2020

Seth MICHAIL

ABBREVIATIONS

B	: BERT
BERT	: Bidirectional Encoder Representation from Transformers
DB	: DistilBERT
D2V	: Doc to Vec
FPR	: False Positive Rate
LSTM	: Long Short Term Memory
MLM	: Masked Language Model
NLP	: Natural Language Processing
RNN	: Recurrent Neural Network
TF-IDF	: Term Frequency-Inverse Document Frequency
TPR	: True Positive Rate
W2V	: Word to Vec

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

With increasing frequency, performance of academic institutions and researchers are ranked by several organizations and reported routinely. Among the major factors affecting the performance are the quality of academic journal and its impact, which are often measured by the citations (Academic Performance, 2019). In the meantime, with the increase of online publications and ease of publishing through the internet, the number of journals has increased dramatically (Larsen, 2010). With this quantitative increase, the impact of the article drops, unless it is published in a relevant journal and brought to the attention of an audience substantially interested in the topic.

Some problems are well suited for specific solution applications, such as machine learning. One example is choosing an appropriate journal for an author to submit their work to be published. The choice of appropriate publication can be difficult, since there are a large number of journals, and researchers have limited time to read articles needed to gain background for their research. Recommendation systems exist that are not appropriate for this task due to focusing on recommending articles to a researcher for reading, rather than recommending a journal for the submission of an article. Yet finding the best journal to which to submit an article for publication is an even more difficult task due to increasing specialization, resulting in a very large number of journals covering related topics.

Selecting the best journal for article submission is made challenging by the need to ensure that the publication is relevant. The relevancy of the publication is important because an article published in a less relevant journal will have less exposure to the intended target audience. As time progresses, more journals are being started, each with its own specialty or subfield. Since these specialties and subfields are increasingly more closely related, the task of finding an appropriate candidate journal for submission of work becomes increasingly more difficult.

Publications can be characterized by a "signature" that can be a composition of the conceptual content of the abstracts of articles already accepted by these journals. In turn, these abstracts can be characterized by vector representations obtained through the methods of natural language processing. Humans can, with ease, understand the meaning of a correctly written block of text. As of the time of this writing, natural language understan-

ding is still far from feasible. Therefore, an approximation or alternative measures must be used. This is the reason for the vectorization of blocks of texts, such as abstracts.

While using a simple keyword search can be useful, it can also easily fail to properly reflect the relevancy of a journal. This should be immediately clear on the basis that keywords are single words, and keyphrases are short arrangements of words that include one or more keywords. If an article could be written entirely in keywords, a keyword search would be enough to efficiently and effectively find relevant candidate publications for submission. However, natural language is not composed of just keywords, a significant portion of meaning is contextual. It is this context dependent meaning that is a critical part of natural language use that leads to keywords being insufficient to adequately distinguish appropriate journals in which to publish. The increasing specialization, together with increasing relatedness, and the inability of keywords to sufficiently distinguish journals, makes the approach outlined here necessary.

Another problem with using keywords is that not all articles have a list of keywords associated with them. Such a list could be generated, however, different algorithms may generate different keyword lists, given the same article. This would be an additional confounding variable.

This thesis is organized in the following manner. Section 2. gives the background for the task outlined, as well as background for the methods employed. Section 3. gives a detailed description of the methodology of how the task was performed. Section 4. gives results and analysis, to include issues encountered in performing the task. We conclude the paper in Section 5. with a summary of the work done and some final notes.

2. LITERATURE REVIEW

Before discussing the relevant previous works, some background concepts will be introduced. For the purpose of explaining these concepts as used in this research, the following terms are defined:

- Abstract - the condensed overview of a work to be published
- Basis vector - a member of the set of unit vectors that spans the vector space
- Journal - a collection of research published on a periodic basis
- Language model - a probability based mathematical model of language, which may include relations between words
- Natural Language Processing - the computer based methods and techniques for analysing and processing language as used by humans
- Neural network - a biologically inspired mathematical framework, to calculate an approximation to a function, that models some property or event
- Neural network architecture - the collective properties of a neural network
- Publication - a collection of research published on a periodic basis or the process of publishing such research
- Vector space - the set of all vectors that can be represented by a linear combination of basis vectors
- Word embedding - a representation of a word in high dimensional vector space

For further clarification, consider that in the standard Cartesian co-ordinate system, it is traditional to use x and y to represent the dimensions, also called axes. Then we can define a vector e_1 as co-linear with the x axis, having origin at zero, in the direction of positive x, and having unit length (length of 1). Similarly for e_2 and the y axis, then any other vector in this system can be represented as a combination of e_1 and e_2, such that the largest and smallest exponent of any part of any expression is, or is reducible to, 1.

Therefore e_1 and e_2 span this vector space, since if there exists an e_3, it can be rewritten in terms of e_1 and e_2. In general, an embedding is an n-dimensional vector represented in a vector space of dimension greater than n. To illustrate, any vector in the one dimensional vector space spanned by e_1 is an embedding in the vector space spanned by e_1 and e_2. Similarly for any vector in the one dimensional vector space spanned by e_2.

2.1. Natural Language Processing

Natural language processing is the resolution of sequences of symbols into larger structures in order to convey information. Generally, humans are endowed with the potential to process natural language, where this potential is realized through training. The use of natural language is a core component of the type of intelligence that distinguishes humans from other animals. The effectively limitless levels of abstraction, the ability to tell and understand stories, and the capacity to reason about ones own reasoning, are all rooted in, or facilitated by, our use of natural language. Natural language, more specifically writing, lead to the development of advanced mathematics, which in turn, lead to a deeper understanding of nature, and thus the technological advancement we see today.

Natural language processing in the context of computer science and computer engineering is the study of how to reproduce, in a computer, the ability to process natural language, which is innate in humans. However, computers require formal languages for their operation, and formal languages are inherently restricted in their expressiveness, and therefore, natural languages must be represented by approximations to allow computers to process them.

2.1.1. Tasks

The development of the methods described were motivated by the need to perform tasks such as named entity recognition, recognizing textual entailment, search engine operations, and coreference resolution. Some tasks, such as reconstructing missing or corrupted tokens, can be used to train a model in such a way that the model can be more easily trained on another task. This is often referred to as transfer learning, and is an approach that has become increasingly popular recently. If, among other factors, the pretraining tasks are chosen carefully enough, a model can achieve state of the art in a wide

range of downstream tasks.

2.1.2. Word Embeddings

Words can be mapped to a vocabulary and used directly, but this would be inefficient due to requiring hundreds of thousands of words. Such an approach would also fail to capture, among other properties, the semantic relatedness of words, information encoded in relative positions of words, or how the presence of some words can affect the meaning of other words. This would lead to poor performance, as is experimentally verified in this work.

An alternative is to convert words to vectors embedded in high dimensional vector space. Often, the higher the dimension, the better the performance, though the scaling is not linear, but also the higher the compute requirements. The basis vectors that span the vector space are not directly related to the words. That is, any given basis vector is not a specific feature, as is the case in some neural network applications. Embedding allows semantic relationships between words to be captured, which is not possible with a simple mapping. The vector representation of words is learned by a neural network, meaning that each embedding is imprinted in the weights.

In 2013, Mikolov et al (Mikolov, 2013a) first discuss continuous bag-of-words and skip-gram models, and later Mikolov et al. (Mikolov, 2014) introduce `word2vec` with great influence on future approaches. They introduce frequent word subsampling and an alternative to hierarchical softmax they call negative sampling. They describe frequent word subsampling as discarding words according to the probability given by equation 2.1

$$P(w_i) = 1 - (t/f(w_i))^{1/2} \tag{2.1}$$

where $f(w_i)$ is the frequency of word w_i in the document and t is a threshold (typically $\sim 10^{-5}$). They define their negative sampling as equation 2.2

$$log\,\sigma({v'_{wO}}^{\top} v_{wI}) + \sum_{i=1}^{k} \mathbb{E}_{w_i} \sim P_n(w) \left[log\,\sigma(-{v'_{w_i}}^{\top} v_{wI})\right] \tag{2.2}$$

which they describe as a simplified version of Noise Contrastive Estimation. Here, σ is the sigmoid function, $P_n(w)$ is the noise distribution, and $-v'_{w_i}$ is the negative sample

vector for word w. They also consider bigrams, weighting according to equation 2.3

$$score(w_i, w_j) = \frac{count(w_i, w_j)}{count(w_i) \times count(w_j)} - \delta \tag{2.3}$$

A later refinement by Horn et al. (Horn,2017) is given by 2.4

$$score(w_i, w_j) = \frac{count(w_i, w_j)}{max\{count(w_i), count(w_j)\}} \tag{2.4}$$

which is 1 if the words only appear as a co-occurrence, and a threshold is usually chosen to be less than one.

In 2014, Le and Mikolov (Mikolov, 2013b) introduced Doc2Vec, based on, and extending, the previous work on `word2vec`. Unlike `word2vec`, it captures the meaning associated with word order, which is lost when using TF-IDF or Continuous-Bag-of-Words models. The principle is to concatenate a paragraph vector with several word vectors, and both forms of vectors are trained using backpropagation with stochastic gradient decent. The purpose is to have vector representations of words and documents that have some semantic similarity. They describe their overall approach, which can be used with common machine learning techniques, during which they introduce two versions of their paragraph vector concept. Paragraph vector distributed memory (PV-DM), which preserves word ordering, and paragraph vector distributed bag-of-words (PV-DBOW), which does not preserve word ordering. They encourage using both in combination.

Lau and Baldwin (Lau, 2016) give a rigorous evaluation of Doc2Vec in 2016. They ask 4 questions, the answers to which they conclude, are that Doc2Vec performs well in different task settings, that PV-DBOW is a better model than PV-DM, that careful hyperparameter optimization can lead to overall performance improvements, and that similarly to `word2vec`, the basic unmodified version of Doc2Vec works well.

2.1.3. Additional Metrics

In addition to embedding words in higher dimensional vector space, a large number of techniques have been developed to extract information for use in various tasks. A few examples are word-word co-occurrence, TF-IDF, and other statistical methods. These approaches are intended for, among other things, document classification

In 2014, Pennington et al. (Pennington, 2014) proposed a weighted least squares regression model defined by equation 2.5

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i{}^T w_j + b_i + \tilde{b}_j - logX_{ij}\right)^2 \tag{2.5}$$

where $f(x_{ij})$ is a weighting function, X_{ij} is the i, j^{th} entry in the word-word co-occurrence matrix, V is the size of the vocabulary, w_i and $\tilde{w}_j$ are context word vectors from W and $\widetilde{W}$ respectively. Yet, the relevance of information represented by relationships between word pairs does not necessarily depend directly or proportionally on the distance separating words.

Zhu and Hu (Zhu, 2017) propose an enhanced version IDF weights, that they call 'context aware', as an efficiency improvement on Doc2Vec. For the weighting they use a global temperature softmax for a normalization function expressed as equation 2.6

$$\psi(\mathbf{w}_j^i) = \frac{|W| e^{\mathbf{w}_j^i/T}}{\sum_{\mathbf{w}_j^i \in W} e^{\mathbf{w}_j^i/T}} \tag{2.6}$$

and they call this model `w-dbow`.

In 2017, Horn et al. (Horn, 2017) develop a method to be used in text classification that is based on the notion of relevancy of words in a body of text for the purpose of representing the content of the text. Their relevancy scores are presented as equation 2.7, with $TPR_c(t_i)$ given in equation 2.8, and $FPR_c(t_i)$ given in equation 2.9

$$r_c\_diff(t_i) = max\{TPR_c(t_i) - FPR_c(t_i), 0\} \tag{2.7}$$

$$TPR_c(t_i) = \frac{|\{k : y_k = c \wedge \mathbf{x}_{ki} > 0\}|}{|\{k : y_k = c\}|} \tag{2.8}$$

$$FPR_c(t_i) = mean(\{TPR_l(t_i) : l \neq c\}) + std(\{TPR_l(t_i) : l \neq c\}) \tag{2.9}$$

Here, $\mathbf{x}_{ki} = tf(t_i) \cdot idf(t)_i)$ and $y_k = c$ means all documents belonging to class c, and $l \neq c$ means all classes other than c. Additionally, the rate quotient is equation 2.10

$$r_c\_quot(t_i) = \frac{min\{max\{z_c(t_i), 1\}, 4\} - 1}{3} z_c(t_i) = \frac{TPR_c(t_i)}{max\{FPR_c(t_i), \epsilon\}} \tag{2.10}$$

and finally, $r_c\_dist(t_i) = 0.5(r_c\_diff(t_i) + r_c\_quot(t_i))$ So the rate distance is the average of the rate difference and rate quotient, where the rate difference is the measure of

words occurring within a class compared to occurring in other classes, and the rate quotient captures words that would otherwise be lost.

Lilleberg, Zhu, and Zhang (Lilleberg, 2015)indicate in 2015 that `word2vec` combined with tf-idf performs better than either alone, supporting the usefulness of tf-idf despite its drawbacks.

The difficulty of text classification can be quantified, according to Collins et al. (Collins, 2018), though only a qualitative assessment is given here.

2.2. Advanced Techniques

Some advanced techniques, based on the previously established concepts, include the use of RNNs and LSTMs to process natural language. These approaches are attempts to resolve long range dependencies and were the basis of the previous state of the art. These approaches have largely been overshadowed by the introduction of the Transformer architecture (Vaswani, 2017), and later, its extension to BERT (Devlin, 2018).

The compute requirements for Transformer scale quadratically with input length, so BERT, which is based on Transformers, is a large model architecture with very high compute requirements. This has motivated various works to reduce the compute cost, with DistilBERT (Sanh, 2019) being one of those works, and will be covered here.

2.2.1. Transformers

In 2017, Vaswani et al. (Vaswani, 2017) introduced the Transformer architecture, which uses attention mechanism alone. Their approach made RNNs and LSTMs nearly obsolete for use on NLP tasks. This was accomplished by encoding the position information in the state, sharing the states across time steps, and including this information when passing the states to the decoder stack. The Transformer architecture now forms the basis for the majority NLP task applications, despite compute requirements scaling quadratically with input sequence length.

The Transformer architecture, as originally proposed, has a stack of encoders and a stack of decoders. Each encoder and decoder is its own neural network with its own sets of weights, and each has its own attention head. Two types of attention are used, self attention and encoder-decoder attention. The self attention determines how other words are related

to a given word. Each attention sublayer of each encoder and decoder use scaled dot product attention, where scaled means divided by the square root of the dimensions of the key vector (for numeric stability). The decoders use an encoder-decoder attention layer that attends to the output of the self-attention layer.

The initial input is the same for Q, K, and V, since there must be a first input, and are in the form of a word embeddings, which can be generated using one of many existing techniques. Thus, the input, I, is dotted with the W^Q, W^K, and W^V to obtain the query, key and value vectors. These vectors can be represented as Q', K', and V', which are then used in the model, where the prime notation indicates being the first instance of these vectors. W^Q, W^K, and W^V are weight matrices, with the rows of W^Q corresponding to word embeddings, the rows of W^K being a sort of memory of embeddings, and the rows of W^V corresponding to the words to be compared against. Each subsequent layer uses the output of the previous layer, the output of the last layer in the encoder is used as the input to the first layer of the decoder.

The W^Q, W^K, and W^V matrices are 512×64 each, and the embeddings are 512, so that the dot product of the embeddings with each of the weight matrices yields a vector of length 64. Since there are 8 attention heads in each multi-head attention block, these vectors are concatenated to give a vector of length 512, which is what is needed for the feed forward network.

For each layer in the encoder-decoder stack, multiple attention layers are used in parallel, their outputs being concatenated before being passed to the next layer. This constitutes multi-head attention, which allows different aspects of the input sequence, that is, different relations between words, to be attended to simultaneously. This means that properties such as word-word co-occurrence can be attended to at the same time as, say, noun-verb relations.

Each word has its own path through the encoder, which is made up of the paths through each self attention layer. These paths are dependent in the self attention component and not dependent, thus allowing parallelism, in the feed forward component, for each layer. Given an input sequence, Transformer outputs a new sequence that is context dependent. That is, each piece of the new sequence is dependent on every piece of the old sequence.

Recurrent neural networks and long-short term memory networks preserve word order

due to their architecture, whereas Transformer attends to all tokens simultaneously in the encoder, and is restricted to previous tokens in left-to-right manner. The positional information is lost in the encoder, meaning that the output sequence order is independent of the input sequence order, and this necessitates a means to preserve word order information.

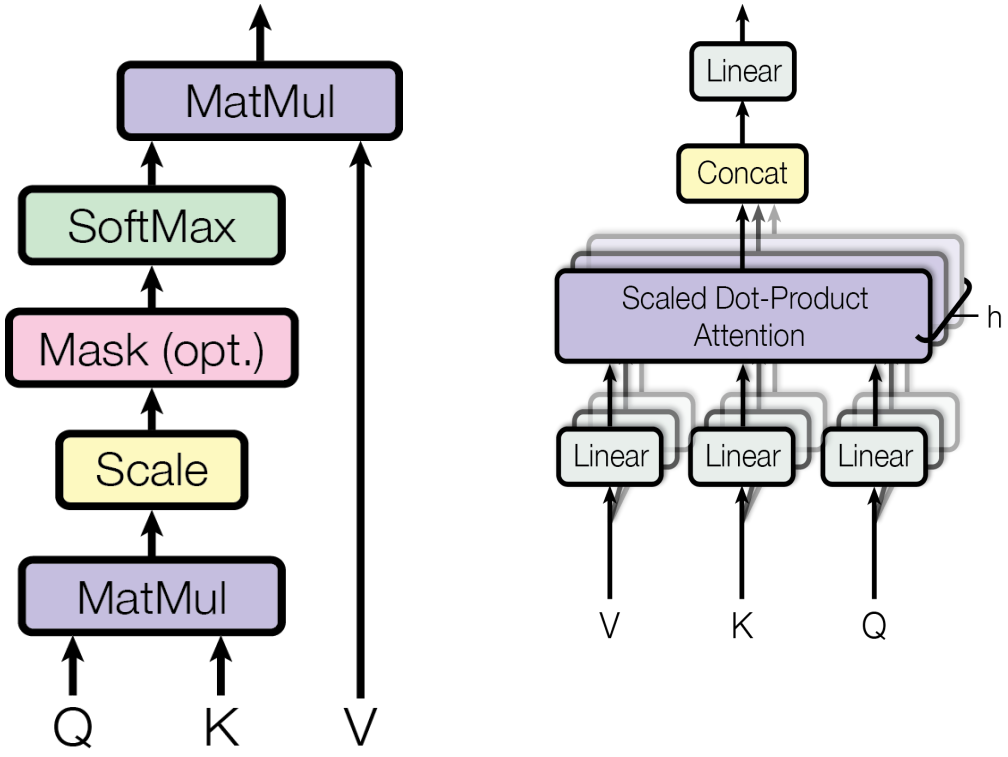
To address the need to preserve positional information, they use positional encodings, which are added to the word embeddings before being processed by the model. The chosen means of generating these positional encodings is given as equations 2.11 and 2.12

$$PE_{(pos,2i)} = sin(pos/\omega) \tag{2.11}$$

$$PE_{(pos,2i+1)} = cos(pos/\omega) \tag{2.12}$$

where $\omega = 1/10000^{2i/d_{model}}$, pos is position, and i is the embedding dimension. Other means of generating positional encodings could be used.

Below are the figures, from Vaswani et al. (Vaswani, 2017), giving a graphical representation of the model architecture of Transformer.



(a) Scaled Dot-Product Attention **(b)** Multi-Head Attention

Figure 2.1. Attention mechanism in Transformer (Vaswani, 2017)

Figure 2.1a illustrates the steps described above, and can be expressed as equation 2.13

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.13}$$

In the original architecture proposed by Vaswani et. al., the Transformer had 6 (N = 6 in Figure 2.2) layers in the encoder, and 6 layers in the decoder. This architecture also used 8 attention heads in each multi-head attention block (h = 8 in Figure 2.1b).

This self-attention is applied for each head in the multi-head attention block of each layer of each encoder and each decoder. With M as the input sequence length, and N is the number of attention layers in each stack, there are two stacks (encoder and decoder), and H attention heads in each multi-head attention block, so 2 $\times$ N $\times$ M $\times$ H applications of this scaled dot-product attention. Each application of scaled dot-product attention results in M $\times$ M operations, which means the complexity scales quadratically with the length of the input sequence. For this reason, applications are often kept to short text environments.

They use skip connections and positional embeddings to preserve word order information. The positional embeddings are element-wise added to the input word embeddings prior to being passed to the first layer of the encoder. The skip connection consists of adding the input word embedding to the output of the attention layer, and the result is normalized to prevent the vectors from becoming too large.

It is important to note that the Transformer model architecture is applicable to more than just natural language processing. It can be applied to anything that can be represented as a sequence.

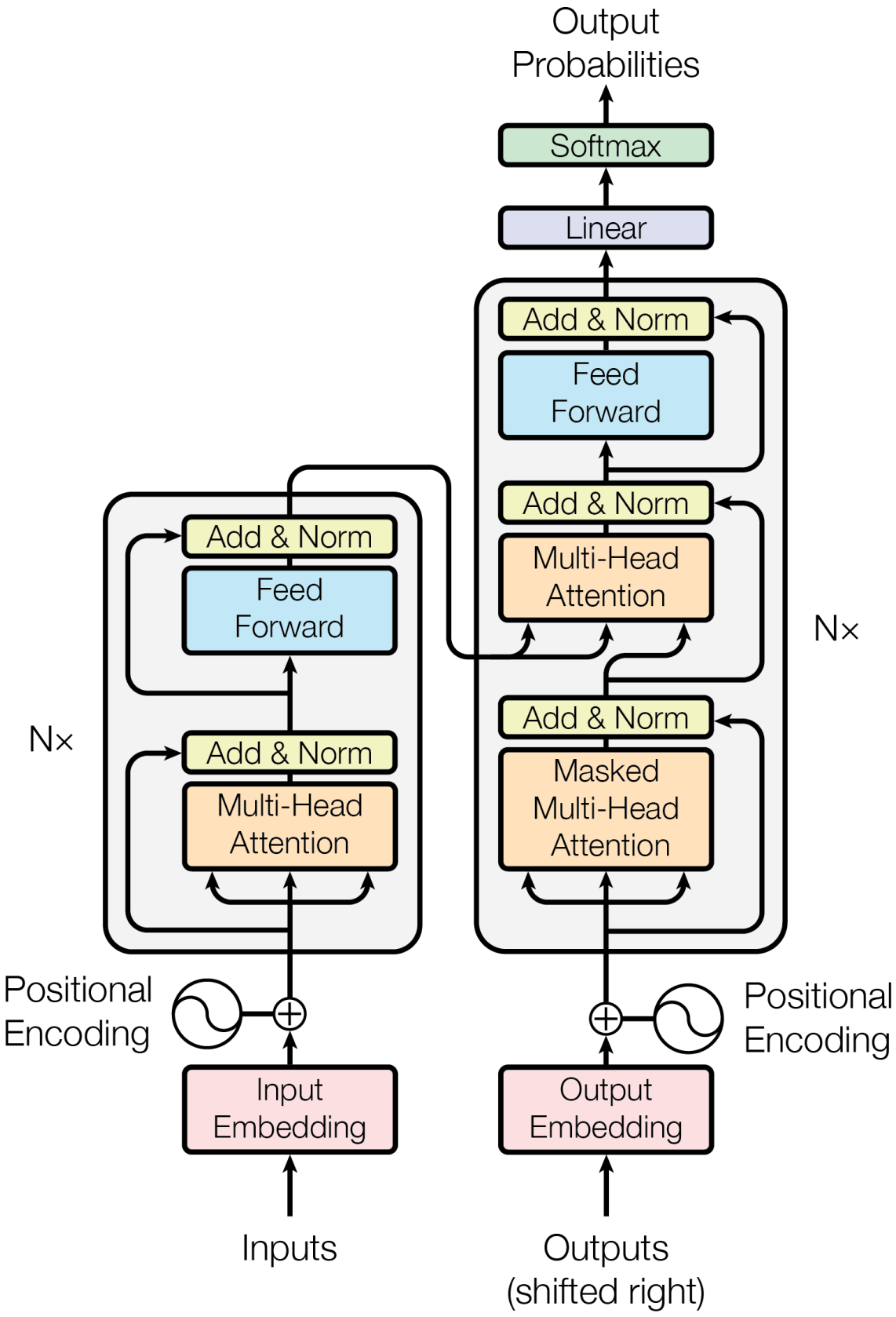


Figure 2.2. Transformer model architecture (Vaswani, 2017)

The steps that follow describe what happens in figure 2.2:

1. Embed: Convert input sequence of words into word embeddings
2. Add: Combine with positional embeddings
3. Project: Inner product with query, key and value weight matrices
4. Attention: Apply scaled dot-product attention, independently for each attention layer in multi-head block
5. Skip: Combine each set of word-positional embeddings with corresponding attention output and normalize
6. Feed: For each attention head, pass through feed forward neural network
7. Skip: For each attention head, combine previous skip connection output with FFNN output and normalize
8. Repeat: Using the result from step 7, repeat steps 3 through 7 for each attention layer of encoder stack
9. Embed: For each previously predicted word, generate embedding
10. Add: For each attention head, combine embedding of previously predicted word and positional embedding
11. Project: Inner product with query, key and value weight matrices
12. Attention: Apply scaled dot-product attention, independently for each self attention layer in multi-head block
13. Skip: Combine each set of word-positional embeddings with corresponding attention output and normalize
14. Add: combine encoder output keys and values with output of previous decoder self attention
15. Attention: Apply scaled dot-product attention, independently for each encoder-decoder attention layer in multi-head block

16. Skip: For each attention head, combine previous skip connection output with encoder-decoder attention output and normalize
17. Feed: For each attention head, pass through feed forward neural network
18. Skip: For each attention head, combine previous skip connection output with FFNN output and normalize
19. Output: Take average of FFNN outputs and apply softmax
20. Repeat: Using the result from step 19, repeat steps 11 through 19
21. Result: The softmax output of the last decoder layer produces the new sequence

2.2.2. Bidirectional Encoder Representations from Transformers

Devlin et al. (Devlin, 2018) developed an approach in 2018 that applies the encoder part of the encoder-decoder stack of a Transformer to generate encoder representations of text. This representation is generated by masking a random portion of the text and setting a model to the task of predicting the masked tokens. Additionally, their approach was applied bidirectionally, meaning that future tokens were considered when predicting the masked or next token. Their model architecture extends the compute demands of the Transformer architecture, thus motivating the search for alternative model architectures with greater efficiency, lower compute requirements, and comparable performance.

The original architecture is commonly referred to as BERT$_{'BASE'}$. In Transformer, there were encoder and decoder stacks with equal number of layers each, whereas BERT replaces the decoder with an encoder of the same size. Hence, Transformer had layers L = 6 + 6, and BERT is just L = 12. Also, Transformer used 512 neurons in each hidden layer and 8 attention heads in the multi-head attention block, whereas BERT uses 768 neurons in each hidden layer and 12 attention heads in its multi-head attention block. Each W^Q, W^K, and W^V is still 512 $\times$ 64, but now, since there are 12 attention heads in each multi-head attention block, this gives the hidden layer size of 768.

Since natural language consists of a set of sequences of words that is significantly smaller than the set of all possible permutations of word sequences, it is necessary to train on sentence level sequences in order to extrapolate the underlying structure of natural languages. For this reason, BERT was pretrained on large corpora, BooksCorpus and English Wikipedia, which are composed primarily of long sequences of words in the form of coherent sentences. Though during pretraining, these sequences were not necessarily required to be legitimate sentences, they could be parts of sentences. For example, a sequence could start in the middle of one sentence and end in the middle of another. This pretraining allows BERT to learn general properties of natural languages that can then be used in transfer learning.

The specific embeddings are learned, meaning that the vector representation of words or pieces of words will change through training. This leads to pretraining being beneficial. However, since BERT is pretrained, the weights have been adjusted to accommodate

specific embeddings, and only small changes can be accepted. Therefore, using a separate embedding model is not feasible, since this would require re-initializing the weights and performing the pretraining task again. While this could be done, it would be computationally expensive and would likely not yield significant improvements. Fine-tuning doesn't change the embeddings, only the weights of the model, so a separate model to learn the embeddings is not necessary, and this leads to more consistency over various applications. The embeddings are fixed so that they will have the same vector values for use with different applications.

BERT uses a lookup table for mapping words to embeddings, which includes individual characters as well as pieces of words. To illustrate, the word 'desiderata' is out-of-vocabulary, meaning that it is not in the vocabulary used by BERT. So BERT would tokenize this word as the following list of tokens:

$$[d, \#\#esi, \#\#der, \#\#ata]$$

Note that the double hash-mark is used to indicate that the given token is actually a component of the preceding token. In this way, out-of-vocabulary words can still be represented, and using pieces of words allows for smaller representations. If represented character by character, this would require ten embeddings, but using this word piece approach allows the word to be represented by only four embeddings. Additionally, the word piece approach allows the vocabulary to be much smaller. Having a smaller vocabulary also means maintaining a smaller list of embeddings, since each entry of the vocabulary maps to its own 768 dimensional vector representation.

BERT was pretrained on unlabeled data simultaneously on two tasks. One of these tasks was reconstructing missing or corrupted tokens, the other task was predicting the if the second of two sentences properly follows the first. For the token prediction task, about 15% of the tokens were masked, corrupted through swapping, or flagged as having potentially been swapped. The goal of this task is to recover the correct token. The goal of the next sentence prediction task is to determine if the second sentence directly follows the first, or not. 50% of the second sentences were drawn randomly from the corpus and the other 50% were the correct sentence.

BERT uses special tokens for various purposes during the first steps for use later in

the model. The special tokens used are [PAD], [UNK], [CLS], [MASK], and [SEP].

- The [PAD] token is used to extend input to the required length, whereas if the input is exceeds the maximum length, it is truncated. The maximum length can be specified, but is limited to 512 tokens.
- The [UNK] token is used to represent out-of-vocabulary words that can't be split into smaller tokens, such as out-of-vocabulary Chinese characters.
- The [CLS] token is used to mark the beginning of the input sequence.
- The [MASK] token is used to obscure a randomly select word from the input, and is used in the token prediction pretraining task.
- The [SEP] token is used to indicate the separation between two sentences, and is used in the next sentence prediction pretraining task.

The pretraining tasks are performed at the same time, alternating between next sentence prediction and masked language modeling, and were inspired by long established tasks used in early education. BERT performs well on a multitude of tasks, often outperforming models engineered for specific tasks. The combination of the model architecture and pretraining tasks may help explain how it both performs a broad range of tasks, and how it performs so well at those tasks.

They performed ablation studies to show what the key features of BERT are. In the pretraining tasks ablation, they demonstrate the significance of bi-directionality by training a model without the next sentence prediction task, and another model without the next sentence prediction task, but also restricted to left-to-right conditioning only. On five tasks, the uni-directional models performed less well.

They performed ablation on model size, training multiple size based variations, to show that the best balance of performance and compute expense, with this best choice configuration being BERT$_{BASE}$, which uses twelve attention heads in the multi-head attention block, twelve attention layers, and 768 neurons in each hidden layer. They state that all other hyperparameters were maintained the same for each model. Additionally, they "hint at" scale having significant impact on learning new tasks, such as 'few-shot',

'one-shot', and even 'zero-shot' learning, if using a fine-tuning approach, rather than feature extraction approach

The last ablation study they perform compares the approaches of feature extraction to fine-tuning. They describe some tasks as being difficult to represent using Transformer encoder architecture, as well as the computational expense of some representations, indicating that feature extraction approach can be beneficial. They extract activations from one or more layers without fine-tuning these layers, and use these contextual embeddings as input to a two layer BiLSTM, the output of which is then given to a classifier. They show that their model is competitive with the state-of-the-art, is near equivalent to their fine-tuning based model, and is therefore applicable to both feature extraction and fine-tuning.

2.2.3. DistilBERT

Sanh et. al. (Sanh, 2019) examine reducing the number of parameters used in the BERT model architecture, subject to the constraint that the performance be suitably close to the original. To do this, they use knowledge distillation, a method in which a smaller model is trained to reproduce the behavior of one or more larger models, as well as a triple loss function. The three losses are

- distillation loss - larger model class probability times logarithm of smaller model class probability, summed over classes
- cosine embedding loss - loss based on cosine similarity of two embeddings
- masked language modelling loss - loss based on reconstructing masked or corrupted tokens

They employ a loss distillation function defined as $Loss_{ce} = \sum_i l_i * log(s_i)$, where *l* is the probability for a given class, assessed by the larger model, and s is the probability for a given class, assessed by the smaller model. They then describe their use of a temperature based softmax function, defined as equation 2.14

$$softmax = \frac{exp^{(z_i/T)}}{\sum_j exp^{(z_j/T)}} \tag{2.14}$$

where T is the temperature and z_i is the probability given to class i. Note the similarity to equation 2.6, with its use of weights. In addition to this triple loss, they reduce the number of hidden layers from twelve to six, remove the 'token-type embeddings' and pooler. They indicate, but don't report on, the possibility of reducing the number of heads in the multi-head attention block.

For training their model, they use the pretrained BERT model as the larger model, and their untrained modified BERT architecture as the smaller model. They report achieving '97%' of the accuracy of the larger BERT model, using 40% fewer parameters, and 60% improvement on speed. They mention in passing the smaller size in terms of MBs. The smaller size, together with the equivalent or better performance, has been found in this work to be a significant difference.

2.3. Related Works

As referenced by Yüksel et. al. (Yüksel, 2018), there are 4 existing solutions, two of these solutions are only available for publications owned or operated by a single specific publishing company. The other two, while not limited to journals owned by a single company, are limited in scope. However, they are all proprietary, and thus, not open for comparison or examination of their performance or how they operate. An additional point of interest is that these extant solutions appear to require the proposed article title, together with the abstract and field of study, in order to generate results. The approach outlined in this work does not consider the title, and as such, would not be affected by changes made to the title.

For the work conducted by Yüksel et. al., they describe applying Naïve Bayes and Support Vector Machine classifiers to word embeddings obtained using bag-of-words method, though the approach to the bag-of-words method they use is not made clear. They indicate that they use only the abstracts and that they achieve 70% accuracy. However, their work focuses on predicting the field of research, rather than predicting specific journals. This means that the number of classes needed to be predicted over is much smaller.

Still, their work represents the closest related work that could be found that is also open for inspection, and provides insights into a possible avenue of investigation to address the difficulty of subdividing large datasets to achieve a smaller number of classes to predict over.

3. MATERIAL AND METHOD

A uniform probability distribution was enforced for all datasets to address class imbalance. This was done in two ways. For the dataset used in the Doc2Vec vs. BERT comparison, the samples were duplicated as needed to ensure each class was represented by the same number of samples. For the dataset used in comparing a single monolithic model to a modular form, a threshold number of samples per class were randomly sampled from a larger set, such that each class was represented by the same number of sample, and that each sample was unique.

3.1. Data Collection

The primary source for the datasets for this research came from Web of Science database (WoS, 2020). As a case study, we choose Computer Science as a sub field, not only because of our experience in the field, but also Computer Science being quite interdisciplinary (ClusteringScientist, 2019). A larger dataset was also created, including additional fields, such as physical science, medical science, behavioral science, and engineering.

Figure 3.3 shows the relations between components of the datasets

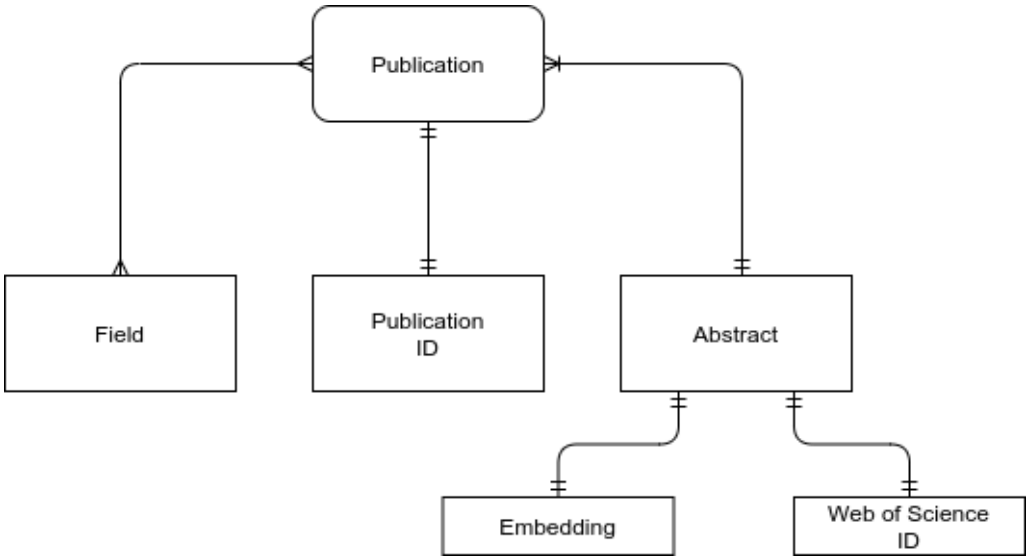


Figure 3.3. Component relations within datasets

3.2. Data Preprocessing

Some issues with the datasets were that publishers were mixed up with case upper and lower. Improper values for various properties, such numerical values for abstract, or mismatched upper and title case for publisher names, or publisher names in the fields property. These issues were addressed primarily through dropping samples that had these types of errors, since the correct values could not be determined with certainty.

3.2.1. Doc2Vec-BERT

The dataset for the Doc2Vec-BERT comparison was maintained in a database containing 444 journals related to computer science, with a total of 23,060 article abstracts. This dataset has significant class imbalance, and to address this issue, a second, reduced copy of the dataset was made for further comparison. For each journal in this reduced copy, if the journal had less than a threshold number of abstracts associated with that journal, it was dropped from the dataset. For the remaining classes, a number was chosen for samples for each class, and if a class had fewer samples, some samples would be duplicated. The resulting dataset contained 29792 samples covering 392 classes.

The relative class balance was considered by comparing each journal on the basis of the number of article abstracts they contained. It was found that some journals contained only one article's abstract, while many others contained 76 article abstracts, with a range of number of article abstracts in between. This indicates that some journals have significantly fewer abstracts, meaning they are underrepresented, and thus will lead to increased misclassification. The class interference was moderate in significance, due to initially having 444 classes, and this directly leads to increased difficulty in classification, but noise was not a consideration as the data were already examined for missing or misplaced data. Overall, the difficulty of this classification task was ameliorated partly by the dataset size not being very large; the total number of words being estimated less than 25 million.

Class imbalance was addressed in two ways. First, by discarding all samples from journals having less than a threshold number of articles published. After minor preprocessing, this threshold was set to 16 articles. This resulted in the datasets having 392 journal classes, which leads to the problem of class interference. The ANN was first trained with

approximately 24.000 ($\approx 80\%$) abstracts for training dataset, with the remainder approximately evenly split for validation and test datasets. The relative sizes of the datasets were due to the need to balance having a large enough test and validation datasets for testing and validation, and large enough training dataset so that the model could learn each class. The second way to address class imbalance was by duplicating samples, such that each journal was represented by the same number of samples. After leveling the probability distribution, the main dataset was split into test, validation, and training datasets.

3.2.2. Monolithic-Modular

For monolithic and modular models, samples having abstracts containing less than 20 words were dropped. The reasoning for this is that the abstract is intended to be an overview of the concepts presented in a given article, and having less than 20 words should not be enough to accomplish this goal. After this process was applied, a minimum threshold of 220 samples per class was implemented, such that all samples for any given class were unique. The resulting dataset contained 296340 covering 1347 classes.

For the modular form, six major fields were created, into which the existing fields associated with each sample were split and accumulated. These major fields are mathematics, physical science, medical science, behavioral science, engineering, and a catch-all field for those that didn't seem appropriate for the first five, referred to as other. This leads naturally to thinking of the existing fields, as listed in the 'Fields' property for each sample, as being sub-fields of the major fields. The specific assignment of these sub-fields to their corresponding major fields may be a source of bias, and thus possibly affecting performance of the models.

Since the intention of using subsets, with potential overlap of conceptual content, is to reduce the number of journals a model would need to distinguish in order to make a prediction, finding an appropriate means of performing the subdivision task is of central importance.

For each modular dataset, a journal was included if it had the corresponding field associated with it. This means that if a journals 'Fields' property contained, say, biomedical engineering, it would be included in both the engineering dataset and the medical science dataset. These datasets were formed from the same dataset that served as the base for the

monolithic datasets

The datasets and models for this comparison were larger than for the previous comparison, and this lead to some difficulty in this classification task, due to limited availability of compute resources.

3.3. Libraries Used

The main libraries used were TensorFlow, Numpy, Pandas, Gensim, Transformers, and MatPlotLib, where only MatPlotLib was non-essential because it was used for visualization of model performance during training. Numpy and Pandas, along with Python built-ins, provided data workflow functionality needed for TensorFlow, Gensim, and Transformers. Gensim and Transformers were the libraries providing the base model architecture, with Gensim providing the Doc2Vec architecture and Transformers providing the BERT and DistilBERT architectures. TensorFlow provided the necessary functionality to adapt the Doc2Vec, BERT and DistilBERT architectures to the specific task addressed by this thesis.

3.4. Data Workflow

For the Doc2Vec models, once the embeddings were generated, they were stored in a Numpy array, where the row position in the array corresponded to the position of the sample in the originating dataset. This array was then stored in a Pandas pickle file (because, unlike the csv format, the pickle format preserves the datatype of the array, saving the step of re-conversion). This approach was taken due to the two lengths of embeddings used, and more importantly, because this was the first approach taken and insurmountable difficulties were encountered in adapting this method to the new approach of storing the embeddings directly in the dataset.

For the BERT and DistilBERT models, for each sample, the 'input_word_ids' were first stripped of punctuation, and then split into a list of string tokens. These string tokens were the string representation of integers, because that's how Pandas stored them when saving to file. The list of tokens were then converted to integers and stored in a Numpy array, because TensorFlow expects the input to be in this form. Using TensorFlows Dataset library, the form factor of the datasets, as used by BERT, were created using all the samp-

les of the source dataset. The source datasets and TensorFlow Dataset datasets constituted the dataflow pipeline

3.5. Models

Several models were compared to find the best performance. Each approach has its advantages and disadvantages. These will be outlined in each of the following sections, and will generally be concerned with compute and storage requirements, as well as specifics of each approach.

3.5.1. Doc2Vec vs BERT

A Doc2Vec model was implemented using Gensim[1] The input into Gensim was abstracts, since Gensim includes its own methods for splitting and cleaning the text, the only manual preprocessing of the data that was necessary was reading in and storing to memory the abstract text, and to remove a few special characters. The abstracts were tokenized and a vector representation generated by Doc2Vec, and then placed in a matrix. Initially, `word2vec` was investigated, but ultimately abandoned due to training and testing accuracy below 1%.

Gensim outputs a vocabulary of word embeddings. Using this vocabulary each abstract was again processed using functions included with Gensim to generate abstract vectors as the sum of the word embeddings of the abstract. This is shown in the pseudocode below.

[1] https://radimrehurek.com/gensim/index.html

Algorithm 1 Continuous space vector representation of words

```
procedure EMBEDDING(a, b)
    Load Model
    Load Dataset
    Create WordsList
    Create empty feature vector, X
    for i < size of dataset do
        Create list of words, g, tokenized from abstract
        Append g to WordsList
        for j < size of g do
            if g[j] in vocabulary of docVectors then
                Create word vector same size as X from g[j]
                for k < size of X; k++ do
                    Append each component of word vector to X
    return X;
```

A Gensim model with output length 1200 was considered but not implemented in order to maintain greater comparability with the BERT model. The Gensim models created were based on the need compare with BERT model architecture, which uses input feature vectors limited in length to 256 features. An additional architecture was used for comparison with the internal representation used in BERT, and this additional architecture differed only in the number of feature vector components, with 768. All other parameters were the same, and are given as follows.

Distributed Bag-of-Words versions of Doc2Vec was used with $dm = 0$, $vector\_size = 768$ or $vector\_size = 256$, $window = 4$, $\alpha = 0.06$, $min\_alpha = 2.5 \times 10^{-7}$, $min\_count = 3$, $max\_vocab\_size = None$, $sample = 10^{-4}$, $workers = 8$, $epochs = 50$, $hs = 0$, $negative = 7$, $ns\_exponent = -0.7$, $dbow\_words = 0$. Respectively, these parameters are as follows [7]:

- whether or not to use distributed memory
- number of vector components used to represent each abstract
- maximum distance between the current and predicted word within a sentence

- the initial learning rate of the model
- the minimum learning rate
- minimum number of occurrences of a word needed for the word to be used in constructing the abstract vector
- whether or not to limit the size of the vocabulary (None means no limit)
- threshold frequency for downsampling high frequency words
- number of worker threads used for training the model
- number of times to pass over the training dataset
- whether or not to use hierarchical softmax for the output
  - if 0, default softmax is used
  - must be 0 for negative sampling
- number of "noise" words used in negative sampling
- shape of the negative sampling probability distribution
  - if negative, samples low-frequency words more than high-frequency words
  - if zero, sample high and low frequency words equally
  - if positive, samples high-frequency words more than low-frequency words
- whether skip-gram is used
  - if 0, skip-gram is not used

After training, the Doc2Vec model was used to generate the feature vectors used as input to an ANN. The variations of the model architectures are according to feature vector dimension, number of hidden layers, and initial learning rate.

The vector representations of the abstracts were then used to train the ANN to classify the abstracts. The model architectures explored had either zero or 2 hidden layers. For the zero hidden layer models, 10% dropout was applied to the output layer. For the ANNs

that use 2 layers, the first layer had 300 neurons, and the second containing 100 neurons. In the usual way, the output had a number of neurons corresponding to the number of journals, in this case, 392. Deeper architectures were investigated, and the performance was found to degrade with depth. The dropout, which was set to 20% for the first layer, and 10% for the second layer during training, improved the learning by extending the time over which learning occurred. The performance of each model was measured using cross entropy softmax loss function, as provided by TensorFlow. During training, the loss function for each model was optimized using Adam Optimizer. None of the architectures investigated used any additional techniques, such as convolution, recurrence, or gating.

The same exploration of Doc2Vec embeddings was applied to BERT token IDs, with the zero hidden layer variation returning only 'nan' for training loss, for both initial learning rates. For this reason, this configuration is not reported. The results for the 2 hidden layer of this architecture are reported, thought they failed to perform. The failure to perform is not surprising, due to the fact that the BERT token IDs are not embeddings, but indices in a lookup table for embeddings. In other words, the token IDs, by themselves, are not meaningful and encompass no relation between words.

The predictions were obtained by TensorFlow after training of ANN, where the predictions were in the form of the predicted journal ID for a given abstract being tested. This predicted journal ID is used in a lookup table to find the name of the publisher. The number of correct predictions was counted and divided by the total number of predictions to get the overall prediction accuracy. For class based accuracy, an array with length being the number of journals was used, with each journal ID mapped to each array index. When a correct prediction was made, the value at the corresponding array index was incremented. The 'best-of-n' accuracy was determined by taking IDs corresponding to the n highest softmax outputs for the sample, and if the correct journal was among these n, the prediction was considered successful.

The confidence for each prediction was also recorded, and was determined as the difference between the first and second highest softmax values for each sample. This confidence was assessed after determining if the prediction was correct on the first pick, and was recorded separately for when the prediction was correct versus incorrect.

3.5.2. Monolithic vs Modular

Since classifiers, in general, perform better when the number of classes is not too large, a comparison approach was developed consisting of multiple models, each predicting over fewer classes. To achieve this, the main dataset was divided according to major fields. The fields were mathematics, physical science, behavioral science, medical science, engineering, and everything else.

Each model used 10% dropout applied to the output, with no hidden layers, and an initial learning rate of 3 $\times 10^{-5}$ for DistilBERT and 5 $\times 10^{-5}$ for BERT. Early stopping was implemented with patience 4 and threshold of 0.001 applied to validation loss, though the conditions necessary for early stopping were never reached for any of the models.

The models were constructed using pretrained models from the Hugging Face Transformers [2] module. This allowed for comparison of the architecture, without needing to be concerned about comparability between model architectures.

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

4. RESULTS AND DISCUSSION

It was observed that by increasing the number of recommendations, the accuracy can be improved, and that the outcome of the model reduces the number of possible journals to be considered, based on the relevance of such journals to the work of the researcher.

4.1. Doc2vec vs BERT

The bodies of the existing articles are not included due to the possibility of influencing the development of a "signature", especially since the result of this work is intended to allow only the submission of abstracts. The justification for this restriction is the need of significant computation required by the natural language processing methods, as well as the abridging nature of abstracts leading to possible reduction of confounding.

Many architectures and configurations were explored, and the best performance was obtained using two hidden layers, with 300 neurons and 100 neurons in the first and second hidden layers, respectively, along with other factors listed previously. The other architectures and configurations are not reported because they performed poorly by comparison.

For testing, there can be several explanations for incorrect predictions. One can be that a test sample is dissimilar enough to not be correctly predicted, with no additional meaning. Another possibility is that the test sample was published in a journal that was not the best fit, and that if such samples had been published in a better fitting journal, that the sample would have been more likely to have been predicted correctly. This can be difficult to determine, especially given the overlap in conceptual content of the publications represented.

For additional comparison, the BERT tokens IDs were given as input to a simple neural network, which is how Doc2Vec embeddings are often used. The BERT tokens IDs were used as inputs to an ANN to confirm that they don't work, which is expected because they are keys used for finding the corresponding embeddings in a lookup table.

For all two hidden layer implementations, the first layer had 300 neurons and used hyperbolic tangent activation with 5×10^{-4} L2 kernel regularization and 20 percent dropout, The second layer had 100 neurons and used swish activation, again with 5×10^{-4} L2

kernel regularization and 10 percent dropout. For the output layer, there were 392 neurons and used softmax with no L2 kernel regularization or dropout. The loss function used was categorical cross-entropy loss, with accuracy metric.

None of the Doc2Vec models achieved a confidence ratio of 1, nor did BERT tokens alone. Only the BERT pretrained model achieved a confidence ratio over 1. For all variants having no hidden layers, 10% dropout was applied to the output layer. In all cases, the probability of randomly selecting 10 journals, without replacement, and the correct journal being among those ten is 2.58%. A variant of the model using only BERT tokens, and having no hidden layers, is not reported due to the training and validation losses being consistently 'nan'.

The Doc2Vec embedding lengths were 256 and 768, since this instance of BERT takes an input of length 256 and produces and output of 768. Since Doc2Vec doesn't use a pretrained model, the embedding length of 768 is examined for comparison with BERT. The initial learning rates were 1.5×10^{-3} and 1.5×10^{-5}, though it is more typical to use 5×10^{-5} or 3×10^{-5} for BERT.

In Table 4.1 the results are shown for two Doc2Vec embedding lengths, two initial learning rates, and two topologies, and for BERT tokens and pretrained model. 'Pass' refers to the model being trained with a number samples equal to the number of samples in the given training dataset. For each model, if the ground truth label was among the ten highest softmax outputs, the prediction was considered correct. This is referred to as 'Best of ten' or 'Top Ten' accuracy, which is reported in Table 4.1. In this table, DistilBERT with zero hidden layers performed best, at 71.2 %.

Table 4.1. Doc2Vec vs. DistilBERT Accuracy Comparison

	Hidden	**LR_0**	**Pass 1**	**Pass 2**	**Pass 3**	**Pass 80**
Doc2Vec	2	1.5E-3	7.1%	12.8%	17.4%	43.3%
Doc2Vec	2	1.5E-5	3.2%	4.0%	3.4%	7.1%
Doc2Vec	0	1.5E-3	4.8%	3.9%	4.7%	5.1%
Doc2Vec	0	1.5E-5	3.1%	3.7%	3.9%	10.6%
Doc2Vec_{768}	2	1.5E-3	4.9%	6.9%	8.7%	50.8%
Doc2Vec_{768}	2	1.5E-5	3.5%	2.8%	2.4%	5.9%
Doc2Vec_{768}	0	1.5E-3	3.6%	3.0%	3.7%	3.3%
Doc2Vec_{768}	0	1.5E-5	3.2%	4.0%	5.5%	9.4%
BERT_{tokens}	2	1.5E-3	4.0%	3.4%	3.7%	4.3%
BERT_{tokens}	2	1.5E-5	2.3%	2.6%	2.3%	5.6%
BERT_{distil}	2	1.5E-5	32.8%	44.0%	53.1%	NA
BERT_{distil}	0	1.5E-5	59.8%	68.1%	**71.2%**	NA

If the ground truth corresponds to the highest softmax output of a model, this is referred to as 'first pick'. Similarly, if the ground truth corresponds to the second highest softmax output of a model, this is referred to as 'second pick', and so on. Figure 4.4 illustrates accuracy for Doc2Vec vs BERT, sorted according to 'pick', and this is referred to as 'pick order accuracy'

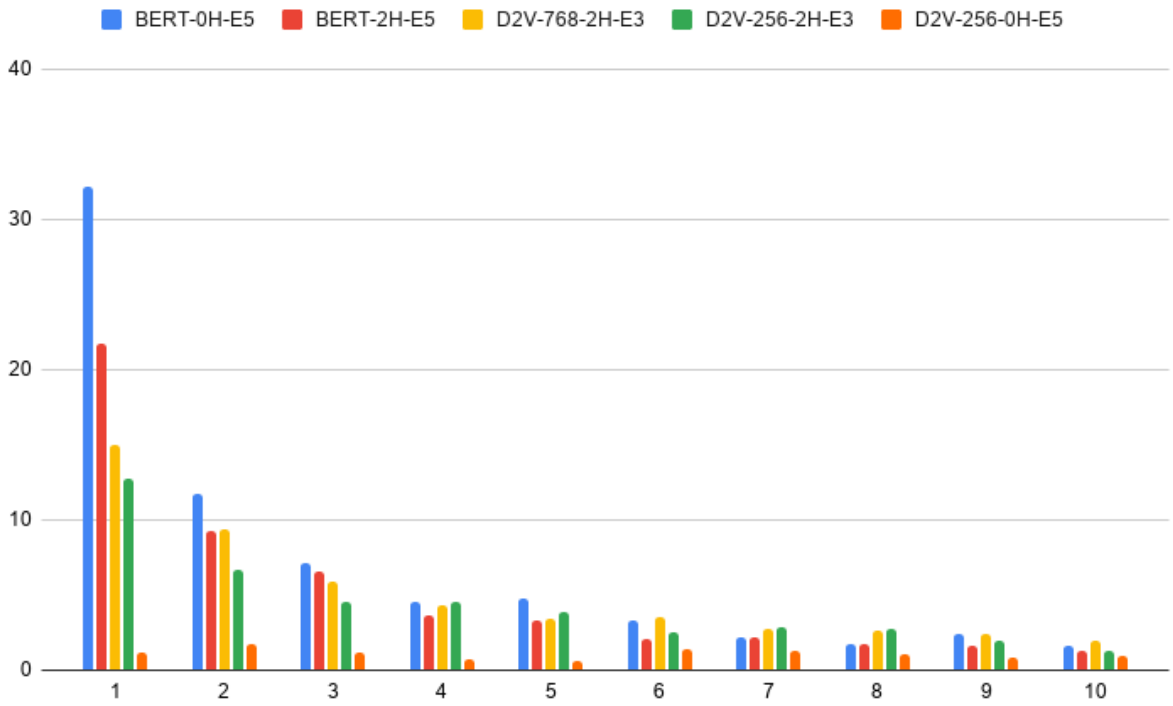


Figure 4.4. Pick order accuracy for Doc2Vec vs BERT

As a sanity check, a randomly selected sample from the test dataset was given to the best performing of each model. The outputs are shown in the Table 4.2, along with the corresponding softmax probability, in descending order (for brevity, only the top 5 are shown). This table illustrates some key differences between Doc2Vec and DistilBERT models. As can be seen, the Doc2Vec models assign a much lower probability when correct, and the distribution is closer to uniform. Additionally, the lower weighted predictions for the Doc2Vec models appear to be less related to the ground truth label than the corresponding lower weighted predictions given by the DistilBERT model.

The apparent closer relatedness of the DistilBERT outputs should be expected since DistilBERT attends to multiple relations between words and considers word position and distance. This means that the concepts contained in abstracts are better represented in the DistilBERT architecture than in the Doc2Vec architecture.

Table 4.2 shows example outputs from several models. The ground truth abstract is *"Managing multimedia data requires more than collecting the data into storage archives and delivering it via networks to homes or offices. We survey technologies and applications for video-content analysis and retrieval. We also give specific examples."* from the paper "Applications of video-content analysis and retrieval".

Table 4.2. Example Output for Doc2Vec vs DistilBERT

(Ground Truth = IEEE MULTIMEDIA)	
Doc2Vec 256 dimensional	
Probability	**Journal**
26.6 %	IEEE MULTIMEDIA
22.7 %	IEEE COMPUTER GRAPHICS AND APPLICATIONS
16.6 %	COMPUTING IN SCIENCE AND ENGINEERING
6.6 %	IEEE PERVASIVE COMPUTING
3.8 %	IEEE SOFTWARE
Doc2Vec 768 dimensional	
Probability	**Journal**
11.6%	IEEE MULTIMEDIA
6.3 %	COMPUTING IN SCIENCE AND ENGINEERING
5.5 %	IT PROFESSIONAL
4.1 %	SIGMOD RECORD
3.2 %	IEEE COMPUTER GRAPHICS AND APPLICATIONS
DistilBERT	
Probability	**Journal**
93.6%	IEEE MULTIMEDIA
1.0 %	IEEE COMPUTER GRAPHICS AND APPLICATIONS
0.8 %	ACM TRANSACTIONS ON MULTIMEDIA COMPUTING COMMUNICATION
0.4 %	MULTIMEDIA SYSTEMS
0.4 %	MULTIMEDIA TOOLS AND APPLICATIONS

4.2. Monolithic vs Modular

It is conjectured here that selecting the top 500 most populous journals would lead to higher accuracy for the monolithic form, at the expense of excluding too many journals to be useful. This conjecture is based on having a larger number of samples per class to train on.

The results suggest that further dividing the fields, such that all sub level models handle at most, say, 200 journals, could lead to maximized accuracy. In practice, an abstract could be submitted to multiple sub models, and the resulting predictions could be combined, and further metrics applied.

For fields of engineering and medical science, the trend of accuracy depending on number of classes appears to deviate slightly. This may be due to human introduced bias, in particular, in selecting which minor fields belong to which major fields. In turn, this suggests alternative means of determining which minor fields belong to which major fields would benefit from either expert input, an algorithmic approach, or both. Additionally, there is the possibility that the assignment of minor fields to publications may have been sub-optimal.

For Table 4.3, all models were trained for 15 epochs and 2 passes over the training dataset. Furthermore, each training dataset consisted of the same number of samples per journal, but different number of journals. The table is sorted according to the number of journals representing each category. Model name refers to the category of journals the model was trained on, with 'monolithic' being all categories combined. **Accuracy**$_1$ refers to the highest softmax output corresponding to ground truth, **Accuracy**$_{10}$ refers to one among ten highest softmax outputs corresponding to ground truth, as described previously, and $\mathbf{P}_{r10}$ refers to the probably of the ground truth corresponding to one of ten randomly selected journals. Confidence is defined here as difference of sums of the first and second highest softmax outputs. Confidence ratio is the ratio of the confidence when correct to the confidence when incorrect, which is expected to be higher if the model learns well.

Table 4.3. Performance Comparison of BERT Models

Model name	**Journals**	**Conf. ratio**	**Accuracy**$_1$	**Accuracy**$_{10}$	$\mathbf{P}_{r10}$
B_other	34	5.57	76.4%	**99.8%**	34.2%
DB_other	34	5.57	75.2%	**99.0%**	34.2%
B_behavioral	74	4.39	67.0%	**97.9%**	14.8%
DB_behavioral	74	4.35	68.7%	**98.2%**	14.8%
B_mathematics	91	2.09	47.6%	**92.6%**	11.6%
DB_mathematics	91	1.97	48.8%	**94.0%**	11.6%
B_engineering	422	1.69	37.2%	**80.6%**	2.4%
DB_engineering	422	1.75	40.6%	**83.0%**	2.4%
B_medical	519	2.71	45.0%	**83.0%**	1.9%
DB_medical	519	2.87	48.9%	**86.2%**	1.9%
B_physical	670	1.27	29.9%	**74.1%**	1.5%
DB_physical	670	1.52	34.4%	**77.6%**	1.5%
B_monolithic	1347	1.38	30.7%	**73.2%**	0.7%
DB_monolithic	1347	1.83	35.8%	**77.3%**	0.7%

Table 4.4. Example Output for Several BERT Models

BERT Monolithic Ground Truth = JOURNAL OF PROTEOME RESEARCH	
Probability	**Journal**
42.1 %	JOURNAL OF PROTEOMICS
8.7 %	PROTEOMICS
6.1 %	JOURNAL OF PROTEOME RESEARCH
3.4 %	MOLECULAR AND CELLULAR PROTEOMICS
2.2 %	PLANT JOURNAL
DistilBERT Medical Ground Truth = SURGICAL ENDOSCOPY AND OTHER INTERVENTIONAL TECHNIQUES	
Probability	**Journal**
72.8 %	SURGICAL ENDOSCOPY AND OTHER INTERVENTIONAL TECHNIQUES
10.4 %	JOURNAL OF GASTROINTESTINAL SURGERY
9.1 %	WORLD JOURNAL OF SURGERY
1.4 %	ANNALS OF SURGICAL ONCOLOGY
1.3 %	OBESITY SURGERY
DistilBERT Engineering Ground Truth = FOOD MICROBIOLOGY	
Probability	**Journal**
62.6 %	INTERNATIONAL JOURNAL OF FOOD MICROBIOLOGY
24.0 %	FOOD MICROBIOLOGY
4.2 %	FOOD CONTROL
2.5 %	FOOD RESEARCH INTERNATIONAL
1.1 %	JOURNAL OF FOOD PROTECTION

Table 4.4 presents a few examples outputs from the monolithic and modular forms. These examples suggest high relatedness between the correct prediction and the additional predictions. The correct predictions often appear to have high probability assigned, suggesting the models are confident in their prediction. DOIs for the published articles are 10.1021/pr5012262, 10.1007/s00464-014-3643-2 and 10.1016/j.fm.2014.01.004.

Figure 4.5 depicts the accuracy with respect to pick order for DistilBERT and BERT architectures, respectively, where the pick order accuracy is as described previously. The greatest difference in accuracy between models appears to be for the first pick, with the highest performing models tending to have the fewest journals over which to make predictions.

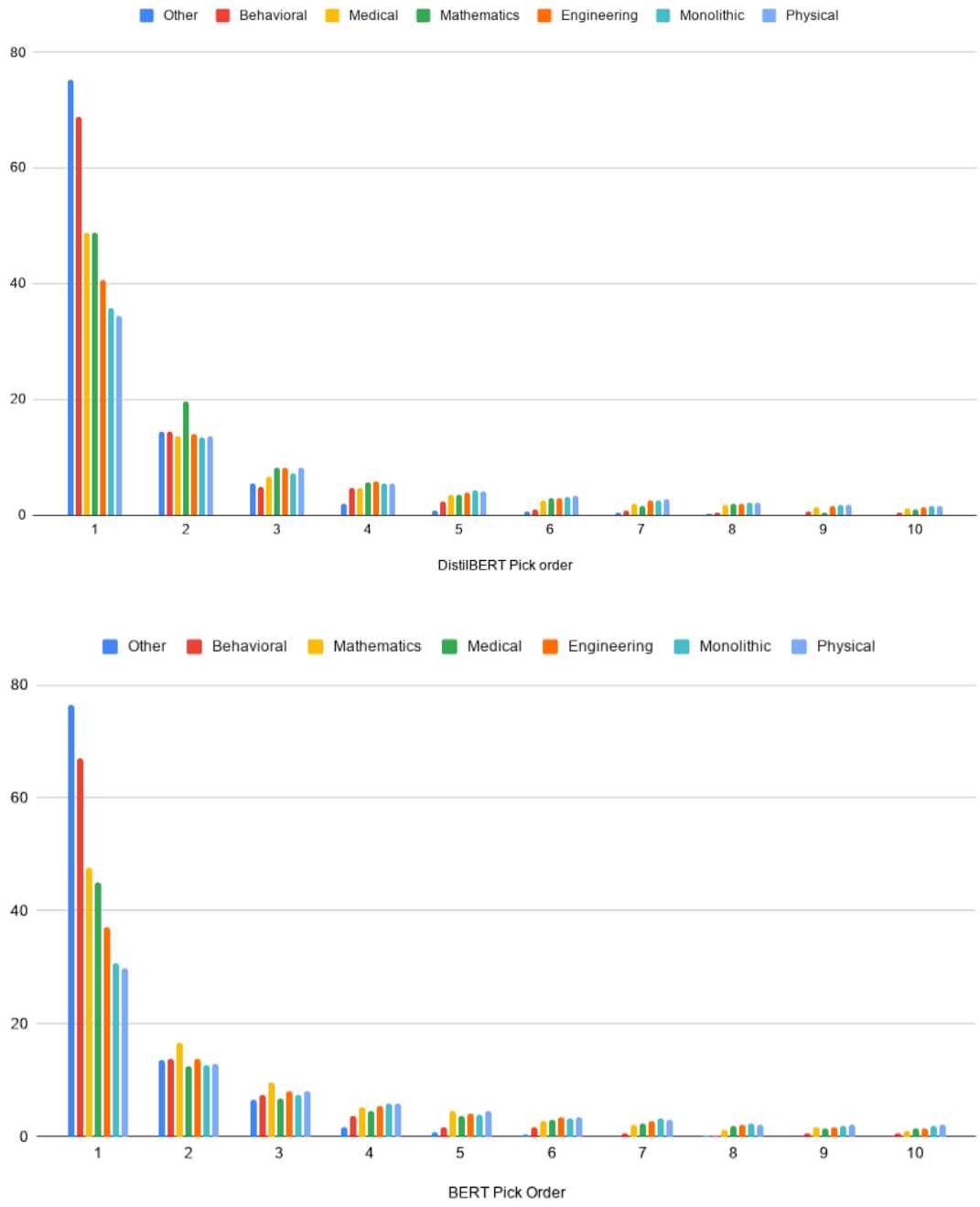


Figure 4.5. Pick order accuracy for DistilBERT

Figure 4.6 depicts the best of ten accuracy compared to the number of journals for BERT and DistilBERT. For each variant, the BERT and DistilBERT models use the same datasets. As can be inferred, the handcrafted assignment of subfields to major fields, and by extension, journals to categories, can have a more significant effect on performance than choice of model architecture.

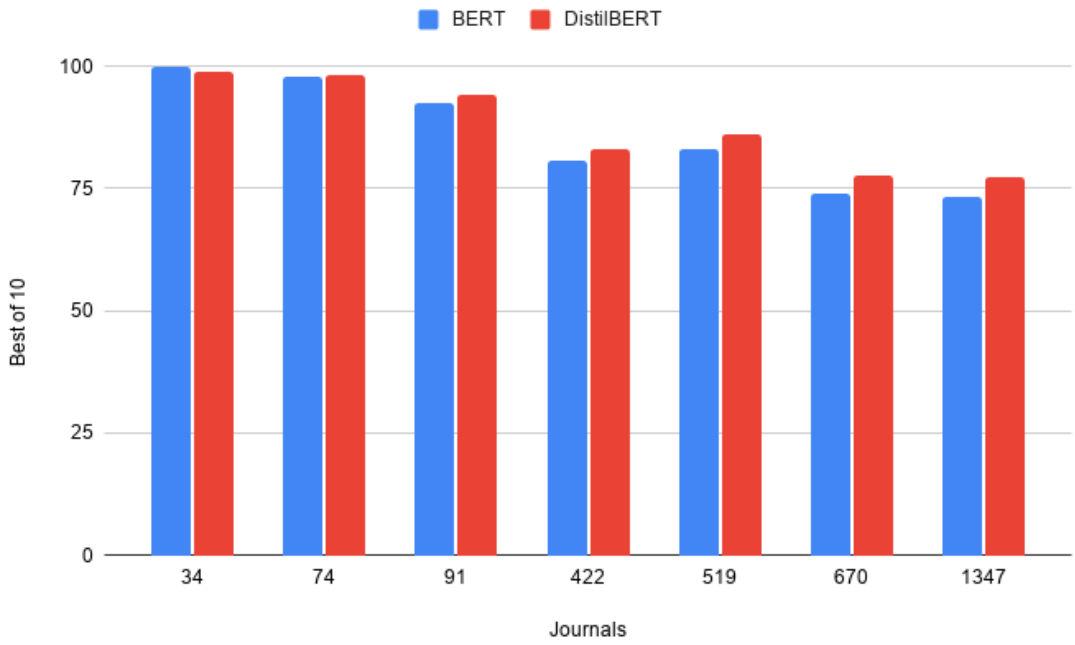


Figure 4.6. Accuracy by number of journals for BERT and DistilBERT

The following Figure. 4.7 is a visual aide illustrating the per sample confidence for correct, and per sample confidence for incorrect predictions made by the monolithic model.

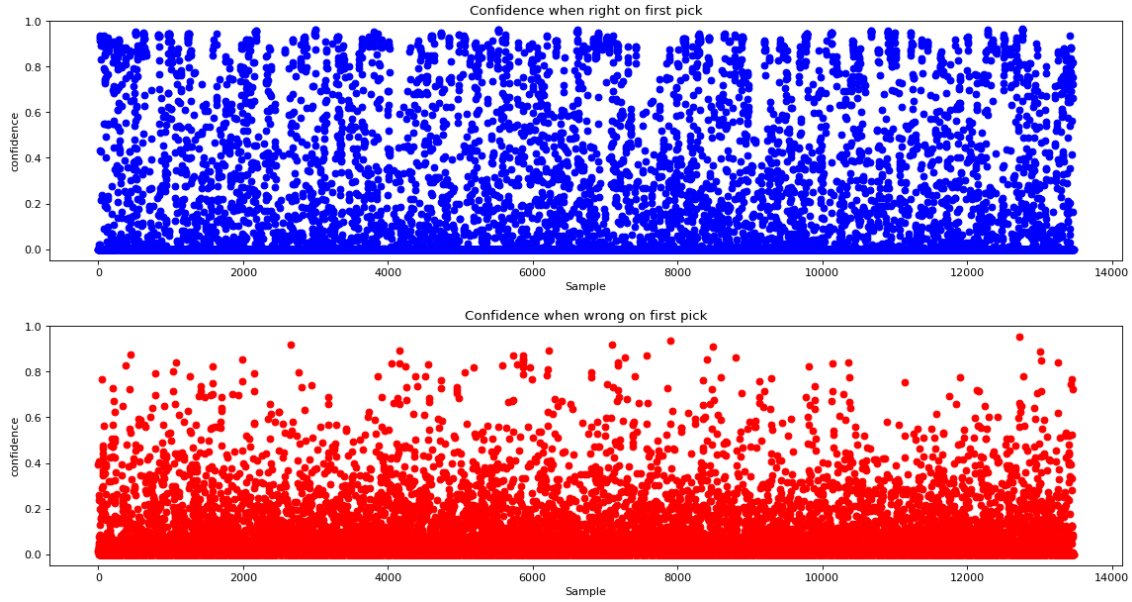


Figure 4.7. Confidence when correct and incorrect (DistilBERT monolithic)

Figure. 4.8 is a visual aide illustrating the per sample confidence for correct, and per sample confidence for incorrect predictions made by the medical science sub-model.

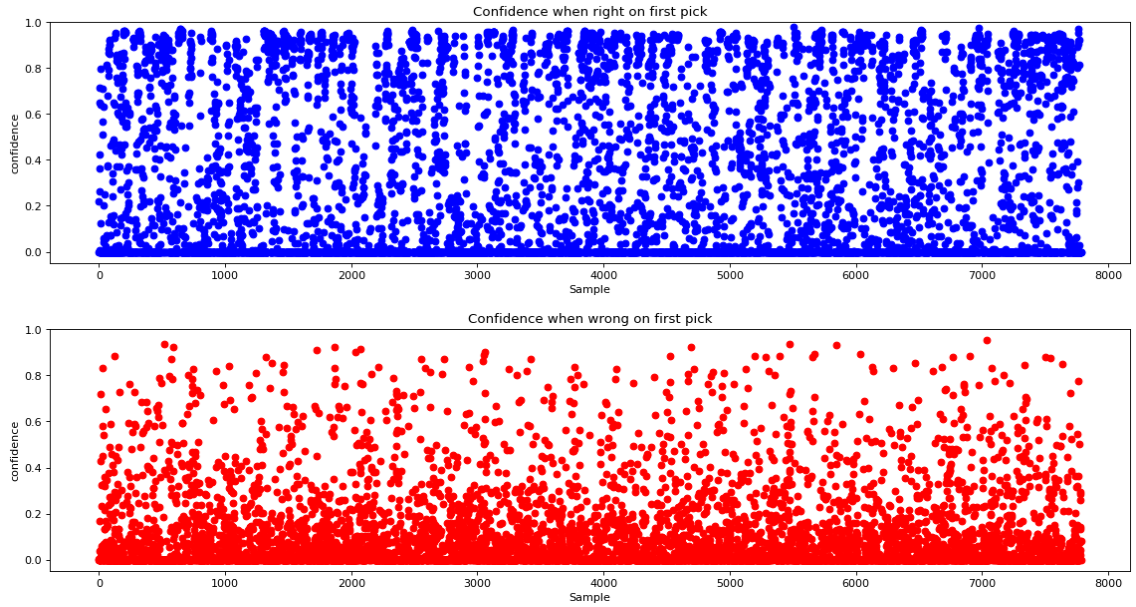


Figure 4.8. Confidence when correct and incorrect (DistilBERT medical)

4.3. Additional Considerations

The Gensim implementation of Doc2Vec does not generate the same embeddings given the same input, which is caused by the random state involved in generating the embeddings. After training the Gensim model, reloading the model to make inference on a sample test, then reloading again, produced a different embedding. The proposed work-around of setting the random number generator seed failed to produce the desired result. This makes the Gensim implementation not viable for some use cases, necessitating an alternative solution in those situations.

Additional reasons for not continuing with Gensim are that BERT models use positional encoding, therefore keeping the important position information. BERT model architecture is a closer approximation to how humans use natural language, due to the bi-directional application of the attention mechanism, as well as the nature of the pretraining tasks, but requires a large amount of compute resources and is slow, due in part to the computation scaling quadratically with input sequence length.

In some cases, the model would not learn, and the weights needed to be reinitialized. Sometimes this would need to be done several times before learning would start. While it is known that neural networks are often affected by the initialization of weights, re-initialization usually resolves the issue quickly. That multiple reinitializations were sometimes needed, for configurations that produced reasonable results when employed previously, suggests that the BERT model architecture can be very sensitive to weights initialization. None of the Doc2Vec models experienced this issue during this work.

If the number of journals in a category is too small, the approach outlined here would be less beneficial. While it would be less useful in reducing the number of candidate journals to consider for submission of work, it may still be useful for sorting such a list according to relevance.

4.4. Unresolved Approaches

An attempt was made to create a neural network model that would determine which sub-model to pass an abstract to. This approach gave poor performance, and so the details are not included here. Due to the nature of such a "dispatcher" model, very high accuracy

would be needed, since the results of the dispatcher model would determine which model the abstract would be given to. If given to the wrong model, the overall results would be expected to be very low in accuracy because the wrong model may not have seen samples from the corresponding journal. The model would still be forced to make a decision though, since all outputs of all models correspond to an existing journal, meaning that there is no "previously unseen" class. If such an approach could be made successful, it would simplify the process for the prospective author, and would further support the overall approach of subdividing the endeavor of classifying journals based on relevancy of content.

This dispatcher model failed to perform sufficiently, achieving approximately 40% accuracy on the first pick, but an accuracy over 80% was needed, for the reasons outlined above. The only differences between this model and the other models described in this work were that the classes were major fields, of which there were only six, and that binary cross-entropy loss with binary accuracy metric was used, instead of categorical cross-entropy loss.

The intention behind using multilabel instead of converting to multiclass was that multilabel should allow 'partial credit', in contrast to the 'all-or-nothing' of one-hot encoded multiclass representation for the loss. This approach would be expected to convey greater benefit when the number of classes is large. That is, if there are, say 1000 classes, and a sample may correspond to any combination of these classes, converting from multilabel to multiclass would be infeasible. These results taken together, suggest some other approach to dividing the datasets may be necessary

5. CONCLUSION

The main benefits of the proposed solution is recommendation of journals an author may not have been previously aware of, recommendation of journals that are more relevant to the topic of an article being submitted, and reducing the number of journals that would need to be considered, thus reducing the effort of finding an appropriate journal in which to submit for publication. Additionally this approach, combined with other approaches, may lead to more efficient distribution of conducted research or discovery of new connections between research fields.

The Gensim model benefits from generating larger feature vectors, but is more sensitive to class interference given a large number of classes for a prediction to span. For cases where high accuracy and reproducibility of embeddings are not important, or significant compute resources are not available, Doc2Vec may still be useful.

Finally, the writing styles of humans can vary in subtle ways that can be difficult for humans, especially those without specialized training, to detect. Machine learning techniques can detect and amplify subtle patterns, and thus distinguish publications small variations in subject matter and writing style of authors, among other factors.

Some key points to consider are:

- Reducing the number of classes a classifier has to classify over leads to improved results, motivating the approach explored here.
- Human introduced bias may affect results, indicating that investigating other techniques for subdividing datasets, such as clustering, is warranted.
- The trade off between compute requirements and accuracy favors higher compute requirements in this particular application.
- Existing solutions are proprietary, hence are not open to examination or comparison on the basis of relevance, scope, or accuracy of results.

6. REFERENCES

Le, Q. and Mikolov, T. 2014. "Distributed Representations of Sentences and Documents." *International conference on machine learning*. 2014.

Mikolov, T. and Sutskever, I. and Chen, K. and Corrado, G. and Dean, J. 2013. "Distributed Representations of Words and Phrases and their Compositionality." *Advances in neural information processing systems 26* (2013): 3111-3119.

Mikolov, T. and Chen, K. and Corrado, G. and Dean, J. 2013. "Efficient Estimation of Word Representations in Vector Space." *International Conference on Learning Representations*

Pennington, J. and Socher, R. and Manning, C.D. 2014. "Glove: Global vectors for word representation." *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*.

Lau, J.H. and Baldwin, T. 2016. "An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation." *Proceedings of the 1st Workshop on Representation Learning for NLP*.

Lilleberg, J. and Zhu, Y. and Zhang, Y. 2015. "Support vector machines and word2vec for text classification with semantic features." *14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*. IEEE.

Gensim https://radimrehurek.com/gensim/models/doc2vec.html#gensim.models.doc2vec.Doc2Vec [Last access date: 2019-06-02].

Collins, E. and Rozanov, N. and Zhang, B. 2018. "Evolutionary Data Measures: Understanding the Difficulty of Text Classification Tasks." *Proceedings of the 22nd Conference on Computational Natural Language Learning*.

Zhu, Z. and Hu, J. 2017. Context Aware Document Embedding. *Arxiv* http://arxiv.org/abs/1707.01521

Horn, F. and Arras, L. and Montavon, G. and Müller, K. and Samek, W. 2017. Exploring text datasets by visualizing relevant words. *Arxiv* http://arxiv.org/abs/1707.05261

Sanh, V. and Debut, L. and Chaumond, J. and Wolf, T. 2019. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter." *EMC^2: 5th Edition Co-located with NeurIPS'19*

Devlin, J. and Chang, M.W. and Lee, K. and Toutanova, K. 2018. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." *Proceedings of NAACL-HLT 2019*, pages 4171–4186

Vaswani, A. and Shazeer, N. and Parmar, N. and Uszkoreit, J. and Jones, L. and Gomez, A.N. and Kaiser, L. and Polosukhin, I. 2017. "Attention Is All You Need." *Advances in neural information processing systems*.

Larsen, P.O. and Von Ins, M. 2010. "The rate of growth in scientific publication and the decline in coverage provided by Science Citation Index." *Scientometrics* 84.3 (2010): 575-603.

Alkan, T.Y. and Gunay, M. and Ozbek, F. and San, B.T. and Kitapci, O. 2019. "Assessment of Academic Performance at Akdeniz University." *DOI* https://doi.org/10.1007/978-3-030-36178-5_87

Alkan, T.Y. and Gunay, M. and Ozbek, F. 2019. "Clustering of Scientist using Research Areas at Akdeniz University." *2019 4th International Conference on Computer Science and Engineering (UBMK). IEEE*

Yüksel, A.S. and Karabiyik, M.A. Vertical Search Engine for Academic Publications. *International Conference on Cyber Security and Computer Science*

ÖZGEÇMİŞ

SETH MICHAIL

nochwysid@gmail.com

ÖĞRENİM BİLGİLERİ

Yüksek Lisans 2018-2021	Akdeniz Üniversitesi Fen Fakültesi, Bilgisayar Mühendisliği, Antalya
Lisans 2008-2014	Arizona State University College of Liberal Arts and Sciences, Department of Physics, Tempe Arizona, USA

MESLEKİ VE İDARİ GÖREVLER

Uzman 2020-Devam Ediyor	Uğur Optik Makina Antalya

ESERLER

Uluslararası bilimsel toplantılarda sunulan ve bildiri kitaplarında basılan bildiriler

1- Michail, S., Hazır, U., Alkan, T.Y., Ledet, J. & Günay, M.. (2020). A Recommendation System for Article Submission to Researchers. In International Conference on Artificial Intelligence and Applied Mathematics in Engineering (ICAIAME 2020).