university of groningen / faculty of science and engineering

Predicting Editor Activity for Open-Access Mega-Journals Using Fixed-Size Abstract Embeddings

Bachelor's Project Thesis

Joël During, s3468194, j.d.g.during@student.rug.nl,
Supervisors: Dr J.K. Spenader & B. Jansema, MSc, MA.

Abstract: Open-access mega journals (OAMJs) can make thousands of publications per month. Due to their broad scope and large pool of editorial board members, finding a suitable editor for a new manuscript is a complex problem. This work examines editors' responses to invitations to edit new manuscripts for one OAMJ and this dataset's potential to be used in an editorial board member recommender system, which aims to automate the finding of suitable editors. The main challenge in this data is that only a few editors are invited for each manuscript, leading to a highly sparse dataset and limiting the amount of data per editor. Different NLP techniques such as Word2Vec, SciBERT, and Doc2Vec are used to transform the titles and abstracts of new manuscripts into fixed-size embeddings that various machine learning methods can use. We present an experiment to evaluate the performance of these methods in predicting an editor's response to invitations for new manuscripts using either a baseline classifier or support vector classification. We find that the embedding methods presented here provide an improvement over random classification. Word2Vec produces the best results in our experiment, although its performance is similar to that of SciBERT and Doc2Vec. An additional experiment examines whether similar editors can be clustered to reduce the problem of limited data availability per editor in the EA dataset. None of the clustering methods presented here provide an improvement over the method without clustering.

1 Introduction

Since the 2006 launch and consequent success of the PLOS ONE journal, open-access mega-journals (OAMJs) have taken over a large part of the academic publishing landscape (Spezi et al., 2017; Björk, 2015). These OAMJs often operate based on article publishing fees, encouraging high numbers of publications. This business model, combined with the journals' broad scopes and relatively low publishing criteria, has allowed them to publish thousands of papers per month, which earns them their *mega* status. OAMJs maintain a large pool of thousands of editorial board members to support this high throughput of publications. The role of an editor is to find reviewers for a newly submitted manuscript, oversee the peer-reviewing process, and decide whether the manuscript gets published based on the reviewer's comments. Therefore, an editor needs sufficient knowledge of the manuscript's topic. An editor's assistant is tasked with finding and inviting editors with sufficient expertise for new manuscripts. The broad scope and large editor pool of OAMJs make finding editors with sufficient expertise for a newly submitted manuscript a time-consuming, manual process with a low success rate.

The Editorial Board Member Recommender (EBMR) system is being developed to solve this problem by Slimmer AI BV. The EBMR system can recommend the most suitable editors for a new manuscript for large journals, such as OAMJs. It uses a dataset linking scientific authors to their publications in various academic sources to achieve this. The recommended editors can be invited to become the editorial board member for the manuscript in question. At this point, the editor can still decline the invitation if the manuscript is not in their area of expertise, they are too busy, or there is a conflict of interest.

The Editor Activity (EA) dataset used in this paper consists of invitations sent out in the past two years to editors for one open-access journal and those editors' responses. This thesis introduced the EA prediction task, in which the response of an existing editor to an invitation for a previously unseen manuscript is predicted. The EA prediction task is a binary classification per editor, predicting accepted and declined decisions. Predictions of this kind could be used to filter out recommendations of the EBMR system where the editor is likely to decline, which could help decrease the decline rate of invitations. This thesis explores the EA dataset's potential and limitations for the EA prediction task using several machine learning and natural language processing methods.

A binary classifier is trained for each editor on the manuscripts they have previously accepted and declined. A simple baseline classifier is introduced that does not fit any parameters but classifies a new manuscript by looking at the class of the most similar manuscripts in the training data using the cosine similarity measure (Salton & Buckley, 1988). A more complex classifier is the Support Vector Machine (Cortes & Vapnik, 1995) which attempts to linearly separate the training data by mapping it onto a higher-dimensional space. This linear separation can then be used to classify new data points after mapping them onto the same higher-dimensional space. The experiments presented in this paper show that classification using Support Vector Machines produces similar or slightly better results than the simple classification based on cosine similarity for the decision prediction task.

Manuscripts can not directly be used to train a binary classifier since they only contain an abstract and a title in the form of text. This paper explores three different methods to transform various-size abstracts into fixed-size high-dimensional embeddings that can be used to evaluate the semantic similarity between manuscripts. These methods use Word2Vec, SciBERT, or Doc2Vec and obtain fixed-size embeddings directly or by averaging word embeddings. The experiments presented here show that a simple method using fixed-size abstract embeddings from averaged Word2Vec embeddings performs best on the decision prediction task.

The EA dataset's main limitation identified in this thesis is the limited data availability per editor. Several methods are introduced to overcome this limitation by clustering editors based on their accepted manuscripts. Clusters of similar editors will contain more data than the editors individually, hopefully resulting in a better performance on the EA prediction task. Two exact-match methods are introduced that cluster editors that have accepted invitations to the same manuscript in the past. Additionally, a similarity-based clustering method is introduced that creates a fixed-size representation for each editor and clusters editors based on this representation using K-means clustering (Lloyd, 1982). An experiment shows that none of these clustering methods improve the performance on the EA prediction task.

2 Background

The Editor Activity (EA) dataset introduced in this thesis contains 626,333 invitations for 110,473 unique manuscripts submitted to an open-access mega-journal. The number of data points (decisions) per editor varies greatly; figure 2.1 shows that the vast majority of editors have less than 25 data points, while some editors have as many as 250 decisions. Limited data availability per editor is the main challenge of the EA dataset; there are many invitations and manuscripts but not many data points for each editor. The manuscript classification task can be seen as a sparse classification task, where many manuscripts are available, but for each editor, labels are only available for a small subset of the complete set of manuscripts.

It is important to note that every data point represents a manuscript for which the editor was manually selected as an appropriate candidate. Therefore, we would expect decisions to be acceptance of the invitation. However, many were declined. This high decline rate highlights the difficulty of this problem, which is the motivation for this thesis. Another peculiarity of the EA dataset is that the decline rate is higher for editors with more data points, as shown in figure 2.2. The most plausible explanation is that editors who get asked to edit manuscripts more often do not have time for all of them and might, therefore, be more selective in which invitations they accept. The manuscript

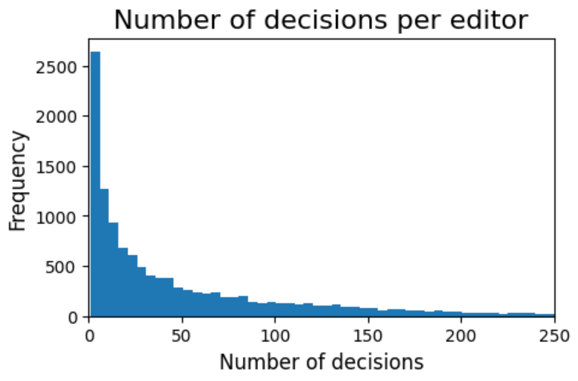


Figure 2.1: Distribution of number of decisions per editor for the EA dataset.

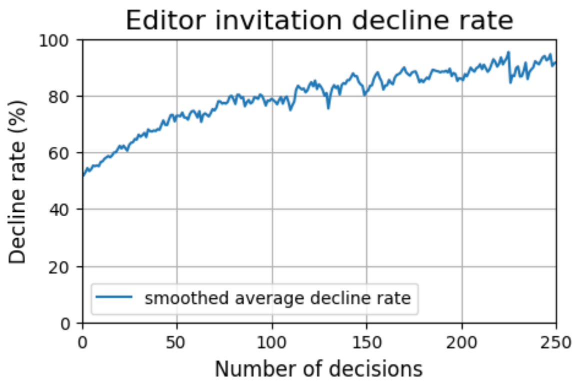


Figure 2.2: Relationship between decline rate and number of decisions per editor for the EA dataset. Decline rate is the average decline rate smoothed over 5 values.

data in the EA dataset can not directly be used to train a binary classifier. We explore three NLP techniques to transform manuscript data from text to a fixed-size numerical representation: Word2Vec, SciBERT, and Doc2Vec.

Word2Vec is a neural architecture that can generate word embeddings based on the usage of the words in a training corpus (Mikolov et al., 2013). Word2Vec obtains embeddings by training a simple neural network with one hidden layer on a corpus. The two architectures available for Word2Vec are *Continuous Bag Of Words (CBOW)*, which predicts a target word using its context as input, and *skip-gram*, which uses a target word to predict its context. The resulting embeddings can be used to evaluate semantic similarity and relationships between words. For example, words that are used in similar contexts in the training corpus are assumed to be semantically similar and will produce similar embeddings.

Similar words in a corpus can be identified by evaluating the cosine similarity of their embeddings (Salton & Buckley, 1988). The cosine similarity of two vectors is the cosine of the angle between the vectors. The resulting similarity value lies in the range $[-1, 1]$, where -1 indicates that the vectors are exactly opposite, 0 indicates that the vectors are orthogonal, and 1 indicates that they are identical. For example, this method has been used to identify relationships between words in a corpus as paths of similar words (Ros, 2020). For the EA prediction task, word embeddings and cosine similarity can not directly be used since manuscripts contain text in the form of titles and abstracts, which contain many words. These various-length texts need to be represented by a fixed-size embedding so they can be used as input to a binary classifier.

Word2Vec word embeddings can be used to represent sentences or longer pieces of text as a fixed-size embedding by averaging the embeddings of all the words in the text. This word averaging model can represent the titles and abstracts of manuscripts in our dataset. In sentences, these fixed-size embeddings have been shown to be surprisingly effective at representing the sentence's content (Adi et al., 2016). For short documents, using this word averaging model can produce similar or better results to more complicated convolutional neural network models (Kalchbrenner et al., 2014) for topic classification, sentiment analysis, and ontology classification tasks (Shen et al., 2018). Other research has found word averaging models to perform well for sentence similarity tasks but to be outperformed by more complex recurrent neural network architectures such as LSTMs (Tai et al., 2015) on sentiment classification tasks (Wieting et al., 2015). Therefore, we will consider not only Word2Vec but also more complex methods such as SciBERT.

The EA prediction task presented in this thesis does not directly correspond to any of the above tasks but involves classification and similarity. Therefore, we expect that word averaging models can perform well relative to more complicated methods for the manuscript classification task.

Other work suggests that word averaging models' performance decreases for longer text pieces (Yuferev & Razin, 2021). In this case, more complex methods might produce better results than Word2Vec. However, we do not expect this limitation to apply to the manuscript classification tasks since the title and abstracts are relatively short (218.2 words on average, SD = 62.6).

Another method to obtain word embeddings is using a more complicated language model such as BERT (Devlin et al., 2018). BERT (Bidirectional Encoder Representations from Transformers) uses bidirectional training of transformers, which is an attention mechanism, allowing the model to access previous states (Vaswani et al., 2017). For the EA prediction task, SciBERT (Beltagy et al., 2019) can be used, which is a BERT-based model pretrained on a corpus of scientific papers. BERT and SciBERT are language models trained for various tasks such as question answering and language inference. However, word embeddings can be extracted from the hidden layers of the architecture, which can then be used in the same way as Word2Vec-generated embeddings, this method of obtaining word embeddings is called *feature extraction* (Peters et al., 2019). The main difference between (Sci)BERT- and Word2Vec-generated word embeddings is that BERT embeddings are context-dependent, whereas Word2Vec generates the same embedding for a word in every context (Miaschi & Dell'Orletta, 2020).

BERT-generated word embeddings have been used for various NLP tasks involving longer pieces of text, such as citation intent classification (Roman et al., 2021) and automatic text summarization (Wang et al., 2019). These tasks differ considerably from the manuscript classification task but show that BERT embeddings can be used successfully to represent pieces of text as fixed-size embeddings. To evaluate how useful these representations are on the EA prediction task, we will use the SciBERT averaging model in our EA prediction experiment and compare it to the Word2Vec averaging model.

Whereas Word2Vec and (Sci)BERT can be used to obtain a fixed-size manuscript embedding by averaging the word embeddings, the manuscript embeddings can also be obtained directly from Doc2Vec (Le & Mikolov, 2014). Doc2Vec is based on Word2Vec but adds an extra neural network input representing the piece of text or paragraph from which the word comes. The two architectures used for Doc2Vec are *distributed memory*, which uses the paragraph ID and context to predict a target word, and *Distributed Bag Of Words (DBOW)*, which only uses the paragraph ID to predict the words in the paragraph. This paragraph ID would be the manuscript ID in the EA prediction task. After training, fixed-size embeddings can be obtained directly for a new manuscript by calculating the weights to the paragraph ID input.

Doc2Vec embeddings have been used successfully in a text classification task where pieces of text from the internet were classified as containing hate speech or not (Djuric et al., 2015). This task is similar to the EA prediction task because it is a binary classification of pieces of text. The main difference with our task is that the dataset used is much larger. An empirical evaluation of Doc2Vec has found that it can provide better results than word averaging models on a semantic textual similarity task (Lau & Baldwin, 2016). These findings should generalize well to the EA prediction task, where the semantic similarity of abstracts will be important in predicting editor decisions. However, in other tasks, such as sentiment analysis of medical texts (Chen & Sokolova, 2021), Word2Vec was found to produce more reliable results than Doc2Vec. To examine if Doc2Vec performs better than Word2Vec for the EA prediction task, we will use the Doc2Vec model in our experiment and evaluate its performance compared to the Word2Vec averaging model.

3 EA Experiment

In order to evaluate the performance of different embedding models and classifiers on the EA prediction task, an experiment is set up using the EA dataset as training data.

3.1 Data Cleaning

In addition to the manuscripts, invitations, and corresponding editor decisions, the EA dataset contains reasons for decline for a subset of the decline decisions. This label has been recorded from July 2020, and its possible values are *'topic out of expertise'*, *'too busy'*, *'conflict of interest'*, and *'other'*.

Of these reasons, only *'topic out of expertise'* indicates that the editor does not want to edit the manuscript's content. Other reasons are given because of other factors such as the invitation's timing and the paper's authors. Therefore, we filter our dataset to contain only decline decisions for which the reason *'topic out of expertise'* was given. The accepted decisions are filtered to contain only decisions given after July 2020, to ensure both classes of decisions are given in the same time frame. This filtering brings the total number of decisions to 259,660 with a decline rate of 65.2%. To ensure every editor has enough data points to be trained and validated on, editors that do not have at least five accepted and five declined decisions are dropped from the dataset,bringing the total number of editors down to 3,380. For these editors, there are 165,291 decisions with a decline rate of 68.4%. Figure 3.1 shows that the distribution of number of decisions is mostly similar after data cleaning, except there are no editors with less than 10 data points and fewer editors with many data points. Finally, the data is split into 80% training data, 10% validation data, and 10% test data. This split is performed in a stratified way, keeping the ratio of accept to decline decisions as similar as possible for each editor. When there are less than 10 data points for a category for a specific editor, one data point will be added to both the validation and test set and the remainder to the training set. The validation set will be used to tune the model parameters, and the test set will be used to evaluate the performance of the tuned models.

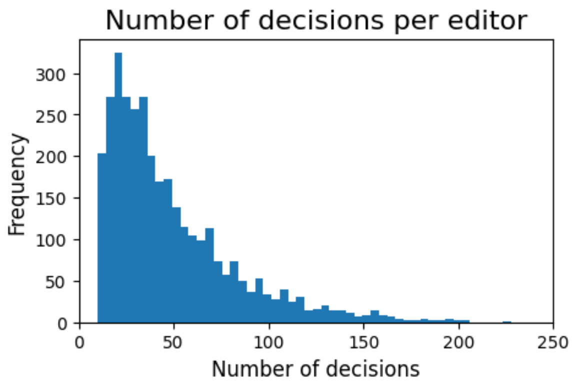


Figure 3.1: Distribution of number of decisions per editor for the EA dataset after data cleaning.

3.2 Models & Training

Each manuscript's title and abstract are extracted as plain text and preprocessed. In this preprocessing step, the title is appended to the abstract and is tokenized, miscellaneous text such as HTML tags is removed, and some frequently occurring stop words such as 'paper' and 'abstract' are removed since they are not discriminative among the abstracts. Then, the processed abstracts are embedded into a fixed-size embedding using one of three models.

The Word2Vec averaging model uses a Word2Vec model trained on a corpus of 250,918 manuscripts that were either submitted to or published by the open-access mega-journal that produced the EA dataset. The Word2Vec model is trained for 10 epochs using the skip-gram method with a window of 10, a minimum word count of 5, and a resulting vector dimensionality of 1000. These parameter values were found by evaluating the performance of the Word2Vec embeddings for the EA prediction task on the validation set. The processed abstracts are transformed by averaging the Word2Vec embeddings of all the abstract's words, resulting in a fixed-size 1000-dimensional embedding for each manuscript. OOV (Out Of Vocabulary) words are ignored in creating fixed-size embeddings.

The SciBERT averaging model uses the pre-trained SciBERT language model. Each processed abstract is split into word pieces using the pre-trained SciBERT tokenizer. The resulting tokenized abstracts are given as input to the SciBERT model. Word piece embeddings are then extracted from the SciBERT architecture by taking the mean of the last hidden state of the network, resulting in 768-dimensional word(piece) vectors. The fixed-size abstract embeddings are then created by averaging the SciBERT embeddings of all word pieces, resulting in a fixed-size 768-dimensional embedding for each manuscript. The embeddings are extracted from the last hidden state instead of, for example, the pooler layer of SciBERT because this provides the best performance on the validation set.

The Doc2Vec model is trained on the same corpus as the Word2Vec model. The Doc2Vec model is trained for 10 epochs using the Distributed Bag Of Words (DBOW) method with a window of 10, a minimum word count of 5, and a resulting vector dimensionality of 300. These parameter values were also found by evaluating the performance of the Doc2Vec embeddings on the validation set. For the processed manuscripts, the embeddings are found by using them as input to the trained Doc2Vec model and calculating the weights to the manuscript ID input. No word averaging is needed for this model, and the resulting manuscript embeddings contain 300 dimensions.

3.3 Classification

A baseline classifier is introduced for the binary classification of manuscripts per editor. This simple classifier does not fit any parameters on the training data. Instead, it predicts the editor's decision for a new manuscript by evaluating the similarity of its embedding with the embeddings of all data points for this editor in the training set. The cosine similarity measure is used to evaluate the similarity of two vectors. The cosine similarity between two vectors $\vec{a}$ and $\vec{b}$ is denoted as $d_{cos}(\vec{a}, \vec{b})$. For any given editor, the training data contains a set of accepted and declined manuscripts. The embeddings of a set of n accepted manuscripts are denoted as $\mathbb{A} = \{\vec{a_1}, \cdots, \vec{a_n}\}$, and that of a set of m declined manuscripts as $\mathbb{D} = \{\vec{d_1}, \cdots, \vec{d_m}\}$. The prediction for a new manuscript with embeddings $\vec{x}$ can be made using one of three methods.

The *max prediction method* computes the max similarity between the target manuscript and those in the accepted and declined sets:

$$a_{\max} = \max_{\vec{a} \in \mathbb{A}} d_{cos}(\vec{x}, \vec{a}), \tag{3.1}$$

$$d_{\max} = \max_{\vec{d} \in \mathbb{D}} d_{cos}(\vec{x}, \vec{d}). \tag{3.2}$$

The prediction is then made using the following function:

$$\text{prediction}_{\max} = \begin{cases} \textit{accepted} & \text{if } a_{\max} \geq d_{\max} \\ \textit{declined} & \text{otherwise} \end{cases} \tag{3.3}$$

The *mean prediction method* computes the mean similarity between the target manuscript and those in the accepted and declined sets:

$$a_{\text{mean}} = \frac{\sum_{\vec{a} \in \mathbb{A}} d_{cos}(\vec{x}, \vec{a})}{n} \tag{3.4}$$

$$d_{\text{mean}} = \frac{\sum_{\vec{d} \in \mathbb{D}} d_{cos}(\vec{x}, \vec{d})}{m} \tag{3.5}$$

The prediction is then made using these values in the same way as for the max prediction method.

The *max-k prediction method* combines the previous two methods. It computes $\mathbb{A}_k$ and $\mathbb{D}_k$: the k manuscript embeddings in the accepted and declined set with the highest similarity value to the target manuscript. Then, it uses the mean prediction method on $\mathbb{A}_k$ and $\mathbb{D}_k$ to make a prediction for the target manuscript.

In evaluating these prediction methods on the validation set for the EA prediction task, the max-k method with $k = 3$ provides the best performance. This is the prediction method used in the rest of our experiments.

Support Vector Classification can also be used for binary classification of manuscripts in the EA prediction task. This method trains a support vector machine on the training data for each editor. The support vector machines fit a linear separation between the two classes of data points, potentially by mapping them onto a higher-dimensional space. This separation can be achieved using a *linear kernel*, *polynomial kernel*, or *Gaussian kernel*. For the EA prediction task, support vector machines with the polynomial kernel are used with degree 3 and a regularization parameter of 10. These parameter values were found by evaluating the performance of support vector classification on the validation set.

4 EA Experiment Results

We evaluated the performance of the different embedding models and classifiers on the EA prediction task with the experiment as explained above. The training set is used to train the classifiers, and the test set is used to evaluate their performance. As a measure of accuracy over all classifiers (one for each editor), the weighted averaged f1-score is used. The results for the *Word2Vec averaging model*, *SciBERT averaging model*, and *Doc2Vec* model for both

Embedding Model	**Baseline Classification**	**Support Vector Classification**	**Random (Stratified)**
			0.65
Word2Vec averaging	0.80	**0.82**	
SciBERT averaging	0.79	0.81	
Doc2Vec	0.79	0.79	

Table 4.1: Weighted averaged f1-scores for different embedding models and classifiers. The random classifier is a stratified model which makes guesses based on the distribution of classes in the training data.

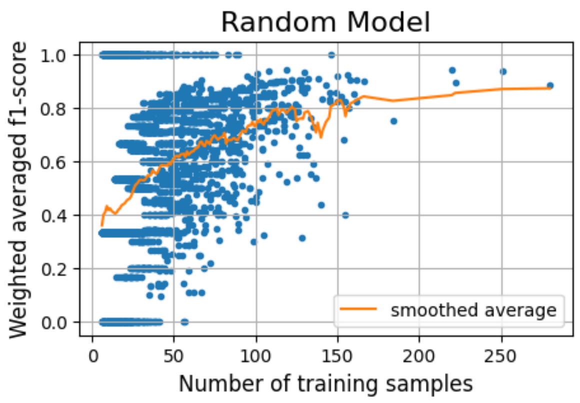


Figure 4.1: Correlation between weighted averaged f1-score per editor on the testing set and the number of training samples for the stratified random model. Each dot represents an editor and the smoothed average f1-score is shown.

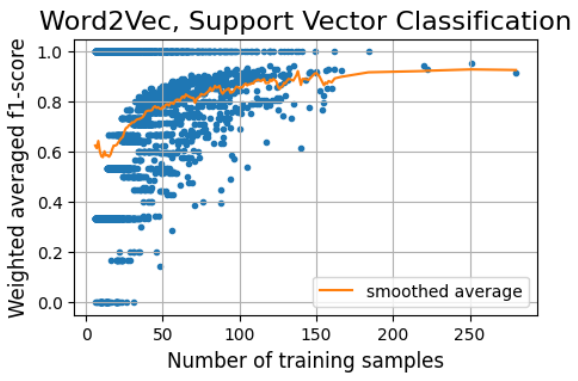


Figure 4.2: Correlation between weighted averaged f1-score per editor on the testing set and the number of training samples for the Word2Vec averaging model with support vector classification.

the *baseline classifier* and *support vector classifier* can be seen in table 4.1.

Figure 4.1 shows the weighted averaged f1-score for each editor individually for the model using random (stratified) classification. This stratified model makes guesses based on the distribution of accept and decline decisions in the training set for each editor. Interestingly, the average score of editors increases with the number of data points. Since this model does not train on the manuscripts, this indicates that the distribution of accept and decline decisions differs for editors with more data points. Figure 2.2 shows this distribution change for the EA dataset; the decline rate increases for editors with more decisions, which explains why the random model performs better for those editors.

As seen in table 4.1, all of the embedding models and classifiers give similar performance. The combination that produces the best performance is the Word2Vec averaging model with support vector classification. Figure 4.2 shows the weighted average f1 scores for this combination for each editor individually. This model offers a substantial improvement over the random prediction model, showing that it is learning from the training data. For editors with more than 60 training samples, this model gets close to a weighted averaged f1-score of 90%, but the performance is much lower for editors with very few training samples. The performance per editor of the other embedding models and classifiers is very similar and can be found in appendix A.

5 Clustering Experiment

An additional experiment was set up to evaluate whether clustering editors based on their accepted manuscripts increases performance on the EA prediction task. Clustering similar editors could help overcome the limitations of sparse data and limited data availability of the EA dataset. For this experiment, the same data cleaning steps are used as for the EA experiment, as described in section 3.1. The embedding model and classifier that produced the best results in the EA experiment will be used, which is the Word2Vec averaging model with support vector classification. Editors will only be clustered based on the content of their accepted manuscripts because we aim to cluster editors that would accept similar manuscripts. This technique differs from most clustering techniques because it is not unsupervised; the data labels are used to determine what data to use for clustering. We introduce three supervised clustering methods for use on the EA dataset.

Exact-match clustering exploits the fact that for most manuscripts in the EA dataset, multiple editors have been invited to edit them. Therefore, it is common for a manuscript to have been accepted by multiple editors. In exact-match clustering, editors that have accepted the same manuscript are clustered together under the assumption that they are likely to accept the same manuscripts in the future. After clustering, one support vector classifier is trained on each cluster's accepted and declined manuscripts. For a new manuscript, an editor's response will be predicted using the classifier of the cluster with which the editor is associated.

Exact-match clustering has two main drawbacks. Firstly, an editor only needs to have an exact match with one editor in the cluster to be assigned to it. Because of this, large clusters might form where any editor only has indirect links (through another editor) with most of the other editors in the cluster. These indirect links are a weaker indicator that two editors might accept the same manuscripts in the future and, therefore, might decrease the performance. Secondly, in exact-match clustering, editors with more accepted manuscripts are likelier to have an exact match to another editor. Because of this, editors with few training samples have a lower chance of clustering than editors with many training samples. This is problematic because the EA prediction system already performs well for editors with many training samples but not for editors with few data samples.

Local exact-match clustering aims to combat the problem of indirect linking with other editors by only clustering an editor with editors to whom they have a direct exact-match link. As a result, each editor will have its own 'local' cluster. One support vector classifier is then trained per editor on all the accepted and declined manuscripts in its local cluster, and predictions are made using the editor's own classifier. Local exact-match clustering solves the problem of indirect linking but still exhibits the problem that editors with more accepted manuscripts are more likely to cluster with other editors.

Similarity-based clustering clusters editors based on a fixed-size representation per editor. This editor representation is computed by averaging the fixed-size embeddings (obtained from the Word2Vec averaging model) of all the editor's accepted manuscripts in the training set. This way, editors that have accepted many similar manuscripts should obtain similar editor representations. The editors are then clustered using K-means clustering (Lloyd, 1982) using a K value of 100, 500, or 1000. One support vector classifier is trained per cluster on the accepted and declined manuscripts of the editors in the cluster. For new manuscripts, a prediction for an editor is made using the classifier of the cluster associated with the editor. This method of clustering clusters all editors, independently of how much data is available per editor. Therefore, it does not exhibit the behavior of (local) exact-match clustering that editors with more data are more likely to be clustered.

6 Clustering Results

We evaluated the performance of the different clustering methods on the EA prediction task with the experiment explained above. The training set is used for clustering and to train the classifiers. The test set is used to evaluate their performance. The results, in terms of weighted averaged f1-scores, can be seen in table 6.1.

Figure 6.1 shows the weighted averaged f1-score per editor together with the number of decisions of the editor. We can see that the average score is lower than that of the method without clustering shown in figure 4.2. The lower weighted aver-

Clustering Method	Performance
None (Baseline)	**0.82**
Exact-Match	0.72
Local Exact-Match	0.78
Similarity-Based, $k = 100$	0.75
Similarity-Based, $k = 500$	0.77
Similarity-Based, $k = 1000$	0.79

Table 6.1: Weighted averaged f1-scores for different clustering methods on the EA prediction task.

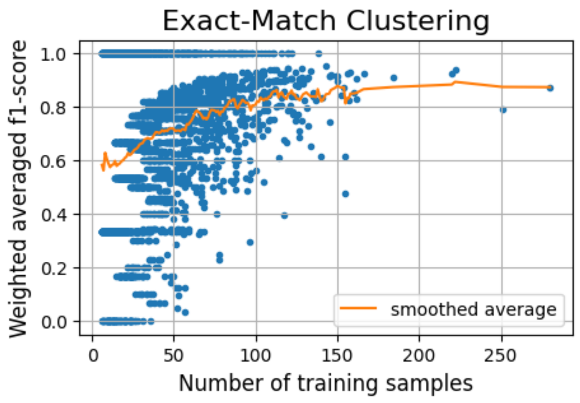


Figure 6.1: Correlation between weighted averaged f1-score per editor on the testing set and the number of training samples for the exact-match clustering method.

age f1-score over all editors of 0.72 reflects this. The other clustering methods also perform worse than the method without clustering; the performance per editor for these methods can be found in appendix B.

7 Discussion

We found that the EA prediction system using the Word2Vec averaging model produces similar or better results to more complicated methods, which is in line with the findings of (Shen et al., 2018) and partially in line with the findings of (Wieting et al., 2015). In the EA prediction task, SciBERT-generated word embeddings performed well when used in an averaging model, which aligns with the findings of (Roman et al., 2021) and (Wang et al., 2019). The Doc2Vec model was found to be useful in binary classification, which is in line with (Djuric et al., 2015). However, the Doc2Vec model did not perform better for the EA prediction task than a word averaging model. This result is in line with the results of (Chen & Sokolova, 2021) but in conflict with the findings of (Lau & Baldwin, 2016). The most likely explanation for this is that the EA prediction task is too different from the duplicate detection and semantic similarity tasks used by Lau & Baldwin, which is why the results do not generalize to our task.

For the EA prediction task, we found the SciBERT averaging model to perform similarly to the Word2Vec averaging model. Since the main differentiating factor of SciBERT word(piece) embeddings from Word2Vec word embeddings is their context sensitivity, our results indicate that context-sensitive information of embeddings is not more predictive for the EA prediction task than static embeddings. One explanation for this is that abstracts and titles of manuscripts are very information-rich pieces of text with many specific content words. Therefore, the semantic content of the words might be a lot more indicative of the content of the abstract than the context in which they are used, especially since in a word averaging model, the context words are also used to create the fixed-size embedding of the abstract. An important consideration is that SciBERT is a much more complicated model than Word2Vec and is therefore more computationally expensive.

The correlation between performance and the number of training samples per editor for the word averaging model shown in figure 4.2 is also present in the results of the random model shown in figure 4.1. Therefore, it is unclear whether the word averaging models benefit from more training data or if they only perform better for editors with more training samples because the decline rate is higher (as shown in figure 2.2). Nevertheless, the word averaging model offers a substantial improvement over random classification. However, the performance is still relatively low for editors with very few training samples (less than 25) in comparison to editors with more training samples. This limitation can be problematic because many editors have very few training samples, as shown in figure 3.1. The average accuracy over the whole testing set of 82% is only achieved on average for editors with more than 60 training samples.

For the EA prediction task, we found that none of the clustering methods presented here improved the performance of the EA prediction system. The exact-match clustering method performed the worst of all the clustering methods, most likely because it clusters editors through indirect links and because this method is more likely to cluster editors with many data points. The local exact-match clustering method solved the problem of indirect linking, and we observed better performance on the EA prediction task. However, this method is still more likely to cluster editors with more data points, which is a possible explanation for why it performs worse than the EA prediction system without clustering. The similarity-based clustering method solved this limitation of the exact-match clustering methods, but still performed worse on the EA prediction task than the method without clustering. A possible explanation for this is that the editors in the EA dataset are too different from one another, which is why clustering only decreases the performance. Another explanation is that the editor representations used for similarity-based clustering are inaccurate because too much information gets lost in averaging the fixed-size manuscript representations, which are in turn averages of word embeddings. In future work, an experiment could be set up to test this theory by classifying groups of texts using a single averaged fixed-size representation.

While the EA prediction task is interesting in and of itself, it is not guaranteed to be exactly representative of the usage of the EA prediction system in combination with the EBMR system. One reason for this is that the current EA data comes from invitations for which the editors were manually selected without the help of the EBMR system. Therefore, the EA dataset contains a higher decline rate than expected from invitations sent out by the EBMR system. The performance measures presented here might not be a good indication of real-world performance and should only be used to compare different models. Once invitation data becomes available from the EBMR system, it is important to reevaluate the performance of the word averaging model on the EA prediction task before incorporating it into the EBMR system.

In future work, the integration of the EA prediction system and EBMR system should be implemented. The most obvious way to integrate these two systems is by using the EA prediction system as a filtering layer on top of the EBMR system. This filtering layer could prevent invitations from being sent out that are likely to be declined. In this approach, there is a possibility that invitations are filtered out that would have been accepted (false negatives). If this is undesirable, high precision on decline decisions indicates better performance. However, if the goal is to decrease the decline rate as much as possible without regard for false negatives, a high recall of decline decisions indicates a better performance. To ensure that the desired precision, recall, or overall accuracy is achieved, the decision can be made to only use the EA prediction system for editors with sufficient samples. For example, if an overall accuracy of at least 70% is required, we could decide, based on figure 4.2, to only use the EA prediction system for editors with more than 25 samples.

8 Conclusion

This thesis presents an explorative overview of different embedding models and classifiers and their performance on the EA prediction task. The main challenge of the EA prediction task is the sparsity of the dataset, which results in a very limited amount of data per editor. We have demonstrated that an EA prediction system with fixed-size embeddings and a binary classifier can offer a substantial improvement over a random model for the EA prediction task. However, the EA prediction task's challenges cause this system's main limitations: its performance is worse for editors with very few data samples. This limitation might not be a deal-breaker when the system is incorporated with the EBMR system, depending on the design decisions made in the process.

As seen in table 4.1, a Word2Vec averaging model with support vector classification offers the best performance. The SciBERT averaging model and Doc2Vec model perform slightly worse, but the results of all three methods are very similar. We can not conclude that the Word2Vec averaging model performs statistically significantly better than the SciBERT averaging model or the Doc2Vec model. However, the Word2Vec averaging model is the simplest of these three models. Therefore, by the principle of Occam's razor, we conclude that the Word2Vec averaging model is the most suitable model for the EA prediction task. Using the Word2Vec averaging model with support vector classification, table 6.1 shows that all of the clustering methods presented in this thesis decrease the system's performance. From this, we conclude that clustering editors based on their accepted manuscripts does not improve performance on the EA prediction task.

This thesis has found the Word2Vec averaging model with support vector classification without clustering to offer the best performance on the EA prediction task. Future work can incorporate this model into the EBMR system and evaluate how this impacts the decline rate of invitations sent out to editors.

References

Adi, Y., Kermany, E., Belinkov, Y., Lavi, O., & Goldberg, Y. (2016). Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *arXiv preprint arXiv:1608.04207*.

Beltagy, I., Lo, K., & Cohan, A. (2019). Scibert: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*.

Björk, B.-C. (2015). Have the "mega-journals" reached the limits to growth? *PeerJ*, *3*, e981.

Chen, Q., & Sokolova, M. (2021). Specialists, scientists, and sentiments: Word2vec and doc2vec in analysis of scientific and medical texts. *SN Computer Science*, *2*(5), 1–11.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273–297.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Djuric, N., Zhou, J., Morris, R., Grbovic, M., Radosavljevic, V., & Bhamidipati, N. (2015). Hate speech detection with comment embeddings. In *Proceedings of the 24th international conference on world wide web* (pp. 29–30).

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Lau, J. H., & Baldwin, T. (2016). An empirical evaluation of doc2vec with practical insights into document embedding generation. *arXiv preprint arXiv:1607.05368*.

Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. In *International conference on machine learning* (pp. 1188–1196).

Lloyd, S. (1982). Least squares quantization in pcm. *IEEE transactions on information theory*, *28*(2), 129–137.

Miaschi, A., & Dell'Orletta, F. (2020). Contextual and non-contextual word embeddings: an in-depth linguistic investigation. In *Proceedings of the 5th workshop on representation learning for nlp* (pp. 110–119).

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Peters, M. E., Ruder, S., & Smith, N. A. (2019). To tune or not to tune? adapting pretrained representations to diverse tasks. *arXiv preprint arXiv:1903.05987*.

Roman, M., Shahid, A., Khan, S., Koubaa, A., & Yu, L. (2021). Citation intent classification using word embedding. *Ieee Access*, *9*, 9982–9995.

Ros, K. (2020). Iteratively linking words using word2vec and cosine similarity..

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, *24*(5), 513–523.

Shen, D., Wang, G., Wang, W., Min, M. R., Su, Q., Zhang, Y., ... Carin, L. (2018). Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. *arXiv preprint arXiv:1805.09843*.

Spezi, V., Wakeling, S., Pinfield, S., Creaser, C., Fry, J., & Willett, P. (2017). Open-access mega-journals: The future of scholarly communication or academic dumping ground? a review. *Journal of documentation*.

Tai, K. S., Socher, R., & Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

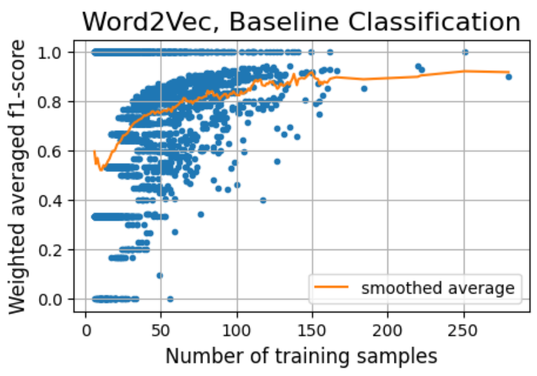
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

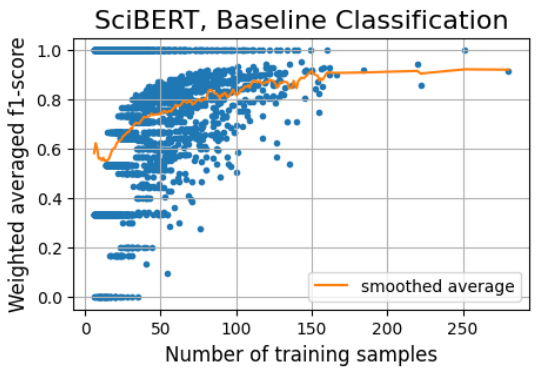
Wang, Q., Liu, P., Zhu, Z., Yin, H., Zhang, Q., & Zhang, L. (2019). A text abstraction summary model based on bert word embedding and reinforcement learning. *Applied Sciences*, *9*(21), 4701.

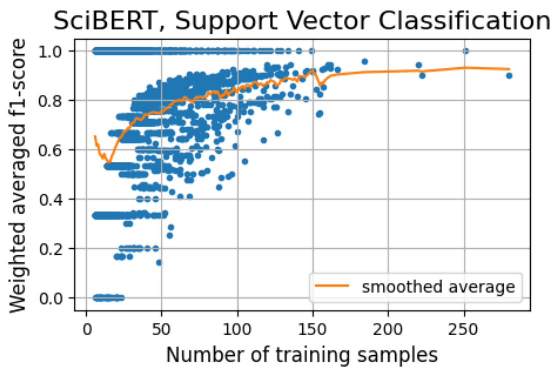
Wieting, J., Bansal, M., Gimpel, K., & Livescu, K. (2015). Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*.

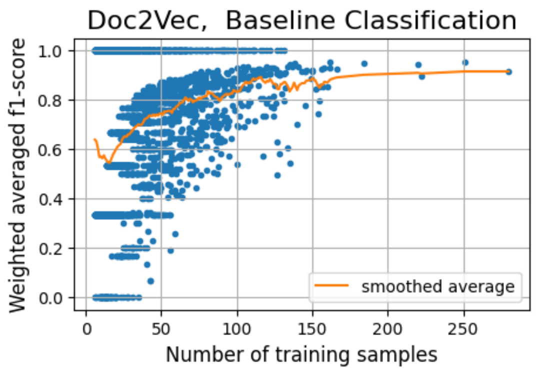
Yuferev, V. I., & Razin, N. A. (2021). Word-embedding based text vectorization using clustering. *Modeling and Analysis of Information Systems*.

A EA Experiment Results









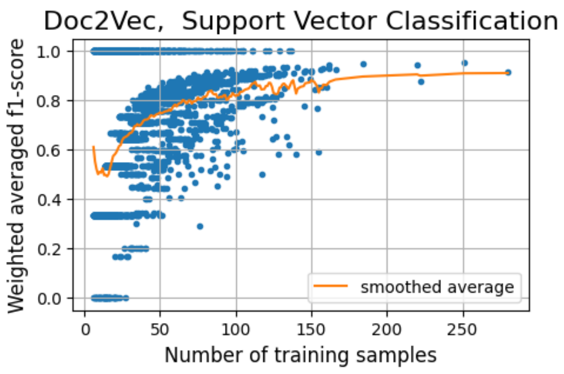


Figure A.1: Plots showing the correlation between weighted averaged f1-score per editor on the testing set and the number of training samples for the different embedding models and classifiers.

B Clustering Experiment Results

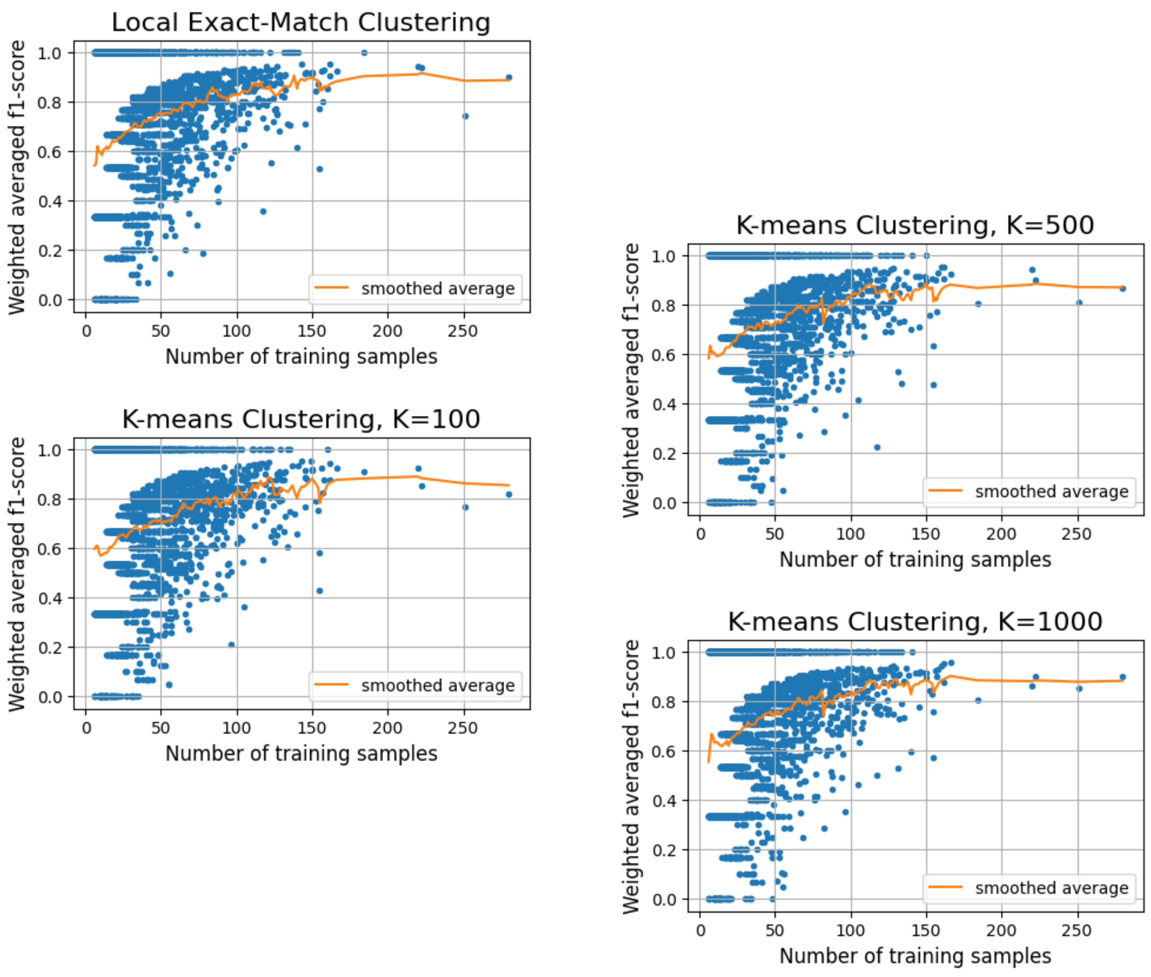


Figure B.1: Plots showing the correlation between weighted averaged f1-score per editor on the testing set and the number of training samples for the different clustering methods.