Comparison of Machine Learning Approaches for Industry Classification Based on Textual Descriptions of Companies

Andrey Tagarev[1] and **Nikola Tulechki**[1] and **Svetla Boytcheva**[1,2]
[1]Sirma AI trading as Ontotext, Bulgara
[2]Institute of Information and Communication Technologies,
Bulgarian Academy of Sciences
{andrey.tagarev,nikola.tulechki,svetla.boytcheva}@ontotext.com

Abstract

This paper addresses the task of categorizing companies within industry classification schemes. The dataset consists of encyclopedic articles about companies and their economic activities. The target classification schema is build by mapping linked open data in a semi-supervised manner. Target classes are built bottom-up from DBpedia. We apply several state of the art text classification techniques, based both on deep learning and classical vector-space models.

1 Motivation

The era of big data has made the task of integrating several heterogeneous sources quite common and important. This is a challenging task because, typically, data representation of the same object can significantly differ in different sources, both in level of detail and type of available information. In addition, different ontologies (classification schema) are used for the same concepts in different datasets. Ontology mapping is itself a challenge even for human experts. Thus the problem of concept classification is very important in data integration.

This paper presents a comparison of different techniques for solving the task of company industry classification based on textual descriptions of companies in DBpedia[1]. The problem of text-classification is defined as follows: for a collection of company descriptions $D = \{d_1, d_2, ..., d_n\}$, assign one or more industry categories to each company from a discrete set of labels $C = \{c_1, c_2, ..., c_k\}$.

There are more than fifteen different company industry classifications of varying granularity. Some of the most popular are: Industry Classification Benchmark (ICB)[2], Global Industry Classification Standard (GICS)[3], Thomson Reuters Business Classification (TRBC)[4], and International Standard Industrial Classification of All Economic Activities (ISIC)[5].

2 Text Classification Methods

Text classification methods are widely used in several applications like e-mail spam filtering (Youn and McLeod, 2007), news categorization (Lin and Hauptmann, 2002), in marketing for product reviews (Dang et al., 2009), etc.

The classical methods (Aggarwal and Zhai, 2012) in text classification are based on standard techniques. Usually the texts from the training corpus are transformed into vectors, where distinct words represent features. One of the simplest and most efficient methods is the probabilistic classifier Naïve Bayes (NB) that has really good performance even with few examples and sparse features and works with a "naïve" assumption for attributes that are conditionally independent. However, usually not all words in a text are independent. Many techniques have been developed (Al-Aidaroos et al., 2010) to overcome the problems caused by the existence of attribute correlations that can lead to classification bias and negatively affect an NB classifier's performance. McCallum and Nigam (McCallum et al., 1998) demonstrate NB classifier applied on "Industry Sector" data, that contains 6,440 company web pages classified in a hierarchy of 71 industry sectors, with a vo-

[1] https://wiki.dbpedia.org

[2] https://www.ftserussell.com/data/industry-classification-benchmark-icb

[3] https://www.msci.com/gics

[4] https://www.refinitiv.com/en/financial-data/indices/trbc-business-classification

[5] https://unstats.un.org/unsd/publication/seriesM/seriesm_4rev4e.pdf

Proceedings of Recent Advances in Natural Language Processing, pages 1169–1175,
Varna, Bulgaria, Sep 2–4, 2019.
https://doi.org/10.26615/978-954-452-056-4_134

cabulary of size 29,964. The reported results for multinomial NB classifier for 20,000 words reach accuracy up to 0.74, and for 1,000 words multivariate Bernoulli model reaches accuracy up to 0.46. Frank and Bouckaert (Frank and Bouckaert, 2006) propose a solution based on multinomial NB that deals with the problem of unbalanced class sizes in Industry Sector dataset with 105 classes, where the largest category has 102 documents, the smallest has 27. They show how by using a centroid classifier and taking into account the significance of different industry sectors classes this method achieves significant gain in some categories. Maximum Entropy classifier used on Industry sector dataset (Nigam et al., 1999) shows better performance than NB and reaches a higher accuracy of 0.788. Ghani (Ghani, 2000) uses an error-correcting codes method and achieves an accuracy up to 0.886 for Industry Sector dataset with a vocabulary size of 10,000.

Support Vector Machines (SVM) (Joachims, 1998) are linear classifiers that, like NB, do not require large training datasets but need more computational time. The most important advantage of SVMs is that they have good performance even for a high dimensional space, such as text classification, where dimensions can be well over 10,000. Rennie and Rifkin (Rennie and Rifkin, 2001) propose application of SVM using one-vs-all and error-correcting output coding for Industry Sector dataset and the results show that this method significantly outperforms NB.

Logistic regression (Genkin et al., 2007) is quite efficient in cases where the dimensions of the feature space surpass the total number of training examples, which is typically the case for text classification datasets.

k-Nearest Neighbor (kNN) Classification method is a proximity-based classifiers that uses distance-based measures for classification task. Usually, such methods use all features in the vector space model which can cause lack of efficiency as not all of them are useful. Several methods for features selection are used, mainly based on the weight of the words in the text. To overcome this problem some modifications of kNN classifier for text documents were proposed, like Weight Adjusted k-Nearest Neighbor Classification (Han et al., 2001). Trstenjak et al.. 2014 present a method for text classification based on kNN and Term frequency inverse document frequency (TF-IDF). Tan (Tan, 2006) demonstrates the performance of kNN and DragPushing strategy based KNN classifier (DPSKNN) methods over subset of 48 sectors of Industry sector dataset (Sector-48 dataset). The Micro-F1 of kNN classifier for Sector-48 dataset is 0.8188 and its Macro-F1 is 0.8235. While DPSKNN shows slightly better performance with for Sector-48 dataset with Micro-F1 0.8544 and Macro-F1 0.8585. In order to emphasize the performance of the methods on common and rare classes, special averages of F1 scores over different classes are used- Micro-F1 - F1 over categories and documents; Macro-F1 - average of within-category F1 values.

Other classification techniques, like Random Forests (Xu et al., 2012), Decision Tree Classifiers (Harrag et al., 2009), Rule–based classifiers, and Conditional Random Fields (CRF) (Lafferty et al., 2001) are also used for this task.

Recent advances in Deep Learning and Transfer Learning introduce more complex methods (Kowsari et al., 2019) with significantly better performance on solving the multi-class multi-label task for industry sectors.

One of the breakthroughs in the area was done by word2vec (Mikolov et al., 2013) that proposes an efficient method for continuous semantics vector representation of words (continuous bag-of-words (CBOW) and skip-grams), by learning from a huge dataset with billions of words. Although its primary purpose is not directly related to text classification, word2vec was used as stepping stone for many other algorithms, because pre-trained word representations are widely used in deep contextual models for word embeddings.

Some more advanced models, like Glove (Pennington et al., 2014) for word representations were proposed, based on so called Global vectors - a new global log-bilinear regression model, where the learning is based on non-zero elements in word-word co-occurrence matrix.

The ELMo (Embeddings from Language Models) (Peters et al., 2018) representation uses deep bidirectional language model (biLM), where each token is assigned a representation which is a function for the whole input sentence.

Skip-thought unsupervised learning model (Kiros et al., 2015) is a generic, distributed sentence encoder that uses robust sentences representation in skip-thought vectors. It uses the idea of continuation of the information in the text

and for an encoded sentence, it tries to reconstruct its surrounding sentences.

Because our dataset is based on DBpedia, the performance of neural networks (NN) algorithms over it for text-based classification task is of primary interest for us.

One of the latest algorithms XLNet (Yang et al., 2019) demonstrates the best performance for DBpedia with error 0.62. Where "classification error" is defined as 1.0 minus classification accuracy. XLNet incorporates ideas from Transformer-XL (Dai et al., 2019). The main advantage of Transformer-XL(Dai et al., 2019) is that it allows the capture of longer-term dependencies and resolves context fragmentation problem.

Other methods with comparable results are Universal Language Model Fine-tuning (ULMFiT), (Howard and Ruder, 2018) with error 0.8 for DBpedia dataset.

Although deep learning algorithms show significant improvement in accuracy they require huge amount of labeled training examples and are computationally expensive in comparison to classical algorithms which do not need such large training datasets.

Some semi-supervised methods (Johnson and Zhang, 2016), (Sachan et al., 2019) and unsupervised methods (Xie et al., 2019) have been proposed for use in combination with supervised models to improve their performance. For example, combination $BERT_Large$+UDA of Unsupervised Data Augmentation (UDA) (Xie et al., 2019) and BERT (Devlin et al., 2018) demonstrate error 1.09 for text classification for DBpedia. The state-of-the-art (SOTA) $BERT_Large$'s error for the DBpedia is 0.64.

In this study we will compare the performance of some deep learning and transfer learning algorithms over DBpedia companies descriptions. We will present experiments with ULMfit and Glove to a baselines of one-hot unigram and one-hot bigram models. These methods were chosen because they are not so computationally expensive in comparison with the others and will serve as a baseline to exploit the potential of the deep learning approach for text-based classification of industry sectors.

3 Dataset

The dataset[6] we used for the experiments is encyclopedic data, consisting of approximately 300,000 textual descriptions of organizations and a classification based on DBPedia classes, which itself is based on Wikipedia. The descriptions are simply the English language abstracts[7] of the the Wikipedia articles about the organizations, and are thus relatively homogeneous in size and style. The *Industries* classification is based on the nature of organization's activity and is generated from the *industry*[8] property of DBPedia. The over 17,500[9] distinct "industries" are normalized *via* a custom mapping we have developed in an iterative manner guided by the taxonomy's commercial applicability. The end result is a multi–level hierarchy but the experiments described in this paper concern only the 32 top-level classes. For example, the organization "Bulgaria Air"[10] is, according to DBPedia, an *Airline*[11], which according to our mapping is a sub-industry of *Air Transport*[12], itself a sub-industry of *Transport*[13], the top-level industry in our mapping. Note that, some organizations, such for example, the "East Japan Railway Company"[14] are directly classified the top-level industry. As is visible in Table 1, the largest classes have well over ten thousand positive examples while many of the smallest have much fewer than a thousand.

4 Experiments and Results

To compare the performance of the methods discussed, we trained a series of algorithms on the same split of our data to get comparable results. We randomly split the training data into three roughly equal parts that were used to carry out three-fold cross validation. In each fold, 60% of the data was used as training data, 7% as valida-

[6] https://gitlab.ontotext.com/trainings/global_datathon/blob/master/data/dt18-ontotext-simple.csv.zip

[7] In Wikipedia the abstracts are the short descriptions between the title and the table of contents of the article

[8] http://dbpedia.org/ontology/industry

[9] DBpedia is rather noisy, over 12,000 of these values are *hapaxes*

[10] http://dbpedia.org/resource/Bulgaria_Air

[11] http://dbpedia.org/resource/Airline

[12] http://dbpedia.org/resource/Air_Transport

[13] http://dbpedia.org/resource/Transport

[14] http://dbpedia.org/resource/East_Japan_Railway_Company

tion data to determine when to stop training and 33% was used as the test data shown to the algorithm only after training is complete. The results reported here are the cumulative performance of the algorithms trained on each fold so each entry in the dataset has been seen as a testing example by one fold one time.

4.1 Linear Baseline

Serving as a baseline, we have a straightforward linear model- a perceptron with no hidden layers. The company descriptions are first processed through a standard NLP pipeline for stopword removal and stemming and then each unigram is converted into a one-hot vector representation[15]. The input for the perceptron is the sum of the unigram vectors.

4.2 Customized Linear Model

The second approach is a customized linear model. It utilizes the same NLP preprocessing of the text and feeds into the same linear perceptron but the features include unigrams and bigrams. Each feature is still represented as a one hot vector as in the baseline model.

4.3 GloVe

The third approach serves as a baseline attempt for incorporating context vectors. It uses the same preprocessing steps as the linear baseline approach (i.e. NLP pipeline for stopword removal and stemming, resulting text is processed into unigrams) but instead of one–hot vectors, GloVe vector embeddings are used. Specifically the 300-dimensional GloVe vectors trained on the large Common Crawl corpus of 840 billion tokens with a vocabulary of 2.2 million words. While there is no additional training of the GloVe embeddings for our specific data, the corpus used to extract context is many orders of magnitude greater than our text data.

The resulting vectors are once again fed into a linear perceptron but because the 300-dimensional resulting vector is much smaller than the one-hot representation in the first two experiments, this approach had much lower training times than the other examined alternatives. Training times were generally under a minute instead of 15-60 minutes.

[15] binary vectors that are all zero values except for the index corresponding to the word or ngram

4.4 ULMfit

Finally, we tested one of the state of the art algorithms for classification with context vectors-ULMfit(Howard and Ruder, 2018) by fast.ai[16]. This is the most sophisticated of the four approaches and the one that varies the most from the initial three.

The first major distinguishing aspect of this approach is the text preprocessing. Rather than the traditional NLP stemming pipeline feeding into one-hot vectors, we use all available company descriptions in order to train a fully custom Language Model based on AWD-LSTM which produces our context vectors directly. For the purposes of these experiments, we used the default settings for the network but there is significant opportunity for in-depth exploration of the language model's performance with various configurations.

The classification training step is implemented as an additional layer added onto an already trained language model. The effect allows relative quick initial training of the context vectors followed by some fine-tuning of the context vectors along the specific classification layer training.

5 Discussion

To compare the performance of the four experiments, let's first look at the class-by-class breakdown of their performance as shown in Table 1. There we can see the F1-score achieved by each algorithm on each of the 32 industries. The second column of the table shows the number of companies of that class and we can see there is a very big discrepancy- from over 76 thousand for the largest class to barely 300 for the smallest class. We can similarly observe that the algorithms achieve good results on the larger classes but their performance degrades and becomes increasingly erratic on the smaller classes.

Looking through the data we can make several other observations. Each algorithm has at least a few classes where they get very poor results and several classes where they achieve the highest results. The difference in results is generally inversely proportional to the size of the class although there are some notable exceptions e.g. the bigram linear model significantly underperforms on the 3rd and 4th largest classes. Overall it is not possible to identify a clearly superior algorithm

[16] https://github.com/jannenev/ulmfit-language-model

Industry	Size	hot-unigram	hot-bigram	GloVe	ULMfit
Entertainment_and_publishing	76309	0.98	0.98	0.96	0.98
Education	55221	0.98	0.98	0.97	0.99
Travel_and_sport	44768	0.99	0.68	0.90	0.98
Public_sector	26391	0.97	0.63	0.98	0.97
Information_technology	10255	0.82	0.97	0.67	0.80
Transport	10007	0.86	0.99	0.79	0.93
Manufacturing	7757	0.93	0.93	0.72	0.66
Financial_services	6086	0.79	0.87	0.61	0.86
Retail	4464	0.72	0.84	0.66	0.67
Food_and_Beverage	3748	0.64	0.83	0.59	0.83
Nonprofit_organization	3655	0.67	0.69	0.68	0.83
Personal_and_household_goods	3206	0.70	0.76	0.96	0.60
Automotive	2564	0.77	0.78	0.46	0.76
Telecommunications	2500	0.89	0.67	0.73	0.74
Aerospace_and_defense	2425	0.69	0.62	0.76	0.63
Engineering	1758	0.30	0.80	0.84	0.27
Utility	1599	0.46	0.52	0.86	0.68
Commercial_and_professional_services	1268	0.61	0.49	0.53	0.07
Fossil_fuel	1213	0.74	0.54	0.74	0.75
Cultural_heritage	1139	0.69	0.76	0.48	0.90
Pharmaceuticals_and_life_sciences	1062	0.81	0.71	0.72	0.76
Real_estate	920	0.52	0.46	0.41	0.64
Healthcare	915	0.48	0.75	0.57	0.51
Marketing	902	0.44	0.53	0.54	0.36
Conglomerate_(company)	780	0.73	0.74	0.81	0.06
Construction_and_materials	764	0.42	0.72	0.50	0.28
Mining	665	0.72	0.91	0.66	0.69
Justice_and_law	577	0.54	0.90	0.72	0.91
Chemical_industry	526	0.56	0.47	0.48	0.15
Agriculture	359	0.32	0.53	0.39	0.19
Forest_and_paper	328	0.88	0.36	0.39	0.22
Metal	302	0.45	0.37	0.88	0.25

Table 1: Class-by-class comparison of F1 scores between the four algorithms

	hot-unigram			hot-bigram			GloVe			ULMfit		
	Prec	*Rec*	*F1*	*Prec*	*Rec*	*F1*	*Prec*	*Rec*	*F1*	*Prec*	*Rec*	*F1*
Micro	0.943	0.901	0.922	0.944	0.909	0.926	0.913	0.899	0.906	0.956	0.888	0.921
Macro	0.788	0.625	0.689	0.786	0.658	0.712	0.700	0.674	0.686	0.861	0.572	0.641

Table 2: Comparison of overall performance between the four algorithms

by looking at the individual classes although it is worth noting that the bigram linear model performance degraded on some of the largest classes.

If we turn our attention to Table 2, we can examine a micro and macro view of the algorithm performance. The micro-average is obtained by summing up each individual decision of the algorithm while the macro-average is obtained by averaging the scores for each class. The first observation here is that the macro-average recall of the ULMfit is particularly low which makes its macro-average F1 score similarly lower than the others. However, because of the huge class size imbalance demonstrated in Table 1, the macro-average is a poor metric for our particular problem. The conclusion we can draw from this is that ULMfit has achieved some abysmal recall on the smallest classes; a likely cause for this is the lack of stemming in the language model used.

Looking to the micro-averages, the algorithms have achieved much closer performance. The GloVe linear approach is the only one falling significantly behind in F1-score while the hot-bigram model narrowly achieves the best F1-score. To

that end there is a clear trade-off, however, with the ULMfit approach having the higher precision while the ngram linear models achieve better recall. As already mentioned, this better recall is likely caused by the stemming in the NLP pipeline which the ULMfit context vectors cannot overcome with the limited size of the corpus.

6 Conclusion and Further Work

The analysis of the results shows that all of the tested approaches produce relatively close results with none emerging as clearly superior to all the others. Overall we observed that ULMfit achieves higher precision while one-hot vector linear models achieve better recall. The linear GloVe vector algorithm achieved inferior results to the other three options overall.

The analysis also shows that the different models have surprisingly varying behavior on the smaller classes indicating that there is still room for improvement in the scores achieved for those smaller classes. ULMfit, while giving comparable results to the linear approaches, presents the best opportunity for that improvement.

There are a few viable directions for exploring further improvement in the performance of the algorithm. One approach would be to work towards improving the reliability of the language model by testing the effect of various parameters or beginning with already trained embedding vectors that are only fine-tuned on our corpus. An alternative direction of experimentation would be to look at the data itself- we know it isn't a true gold standard and expect a relatively large rate of error so it is possible that many of the algorithm "mistakes" are actually errors in the underlying data rather than of the algorithms themselves.

Acknowledgements

The research presented in this paper is partially funded by project The Intelligent Matching and Linking of Company Data project (CIMA), grant BG16RFOP002-1.005-0168-C01 by the European Unions European Regional Development Fund through Operational Programme Innovations and Competitiveness 2014-2020 call Intelligent Specialization. We are grateful to anonymous reviewers for useful comments and suggestions.

References

Charu C Aggarwal and ChengXiang Zhai. 2012. A survey of text classification algorithms. In *Mining text data*, Springer, pages 163–222.

Khadija Mohammad Al-Aidaroos, Azuraliza Abu Bakar, and Zalinda Othman. 2010. Naive bayes variants in classification learning. In *2010 International Conference on Information Retrieval & Knowledge Management (CAMP)*. IEEE, pages 276–281.

Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860* .

Yan Dang, Yulei Zhang, and Hsinchun Chen. 2009. A lexicon-enhanced method for sentiment classification: An experiment on online product reviews. *IEEE Intelligent Systems* 25(4):46–53.

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

Eibe Frank and Remco R Bouckaert. 2006. Naive bayes for text classification with unbalanced classes. In *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, pages 503–510.

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

Rayid Ghani. 2000. Using error-correcting codes for text classification. In *ICML*. pages 303–310.

Eui-Hong Sam Han, George Karypis, and Vipin Kumar. 2001. Text categorization using weight adjusted k-nearest neighbor classification. In *Pacific-asia conference on knowledge discovery and data mining*. Springer, pages 53–65.

Fouzi Harrag, Eyas El-Qawasmeh, and Pit Pichappan. 2009. Improving arabic text categorization using decision trees. In *2009 First International Conference on Networked Digital Technologies*. IEEE, pages 110–115.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146* .

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

Rie Johnson and Tong Zhang. 2016. Supervised and semi-supervised text categorization using lstm for region embeddings. *arXiv preprint arXiv:1602.02373* .

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*. pages 3294–3302.

Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. 2019. Text classification algorithms: A survey. *Information* 10(4):150.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of the Eighteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., pages 282–289.

Wei-Hao Lin and Alexander Hauptmann. 2002. News video classification using svm-based multimodal classifiers and combination strategies. In *Proceedings of the tenth ACM international conference on Multimedia*. ACM, pages 323–326.

Andrew McCallum, Kamal Nigam, et al. 1998. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*. Citeseer, volume 752, pages 41–48.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Kamal Nigam, John Lafferty, and Andrew McCallum. 1999. Using maximum entropy for text classification. In *IJCAI-99 workshop on machine learning for information filtering*. volume 1 (1), pages 61–67.

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

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* .

Jason DM Rennie and Ryan Rifkin. 2001. Improving multiclass text classification with the support vector machine. In *Series/Report no. AIM-2001-026CBCL-210*. http://hdl.handle.net/1721.1/7241.

Devendra Singh Sachan, Manzil Zaheer, and Ruslan Salakhutdinov. 2019. Revisiting lstm networks for semi-supervised text classification via mixed objective function. In *AAAI 2019*.

Songbo Tan. 2006. An effective refinement strategy for knn text classifier. *Expert Systems with Applications* 30(2):290–298.

Bruno Trstenjak, Sasa Mikac, and Dzenana Donko. 2014. Knn with tf-idf based framework for text categorization. *Procedia Engineering* 69:1356–1364.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. 2019. Unsupervised data augmentation. *arXiv preprint arXiv:1904.12848* .

Baoxun Xu, Xiufeng Guo, Yunming Ye, and Jiefeng Cheng. 2012. An improved random forest classifier for text categorization. *JCP* 7(12):2913–2920.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237* .

Seongwook Youn and Dennis McLeod. 2007. A comparative study for email classification. In *Advances and innovations in systems, computing sciences and software engineering*, Springer, pages 387–391.