Exploring the Relationship Between Algorithm Performance, Vocabulary, and Run-Time in Text Classification

Wilson Fearn
Brigham Young University
wilson.fearn@gmail.com

Orion Weller
Brigham Young University
oweller@byu.edu

Kevin Seppi
Brigham Young University
kseppi@byu.edu

Abstract

Text classification is a significant branch of natural language processing, and has many applications including document classification and sentiment analysis. Unsurprisingly, those who do text classification are concerned with the run-time of their algorithms, many of which depend on the size of the corpus' vocabulary due to their bag-of-words representation. Although many studies have examined the effect of preprocessing techniques on vocabulary size and accuracy, none have examined how these methods affect a model's run-time. To fill this gap, we provide a comprehensive study that examines how preprocessing techniques affect the vocabulary size, model performance, and model run-time, evaluating ten techniques over four models and two datasets. We show that some individual methods can reduce run-time with no loss of accuracy, while some combinations of methods can trade 2-5% of the accuracy for up to a 65% reduction of run-time. Furthermore, some combinations of preprocessing techniques can even provide a 15% reduction in run-time while simultaneously improving model accuracy.[1]

1 Introduction

With the increasing amount of text data available, text analysis has become a significant part of machine learning (ML). Many problems in text analysis use ML methods to perform their task, ranging from classical problems like text classification and topic modeling, to more complex tasks like question answering. Although neural networks have become increasingly common in the research field, many industry NLP problems can be well served by less complex but more efficient and explainable models, such as Support Vector Machines (SVMs) or K-Nearest Neighbors (K-NN).

[1]Our code and results are publicly available at https://github.com/wfearn/preprocessing-paper

We focus on the text classification problem, where the dominant approach to using these non-neural models is to first calculate the number of unique terms in the dataset (the *vocabulary*, size V) and encode each instance of the dataset into a bag-of-words (BoW) representation (Joachims, 1998; Zhang et al., 2010). This results in a high-dimensional vector of size V that indicates whether each given word of the vocabulary was used in this instance.

However, the vanilla approach to the BoW representation can lead to sub-par performance, as shown by numerous studies that have examined how preprocessing techniques affect the BoW w.r.t. performance and vocabulary size. These studies have examined this representation in fields such as information retrieval (Chaudhari et al., 2015; Patil and Atique, 2013; Beil et al., 2002; Senuma, 2011), text classification (Yang and Pedersen, 1997; Caragea et al., 2012; Uysal and Gunal, 2014; Vijayarani et al., 2015; Kumar and Harish, 2018; HaCohen-Kerner et al., 2020; Symeonidis et al., 2018) and topic modeling (Schofield and Mimno, 2016; Blei et al., 2003). They suggest a myriad of preprocessing techniques that could improve performance, ranging from choosing features that have high mutual information, low frequency, or simply remove punctuation.

Another related problem of the BoW representation is that this sparse high-dimensional vector does not scale well to datasets with large vocabularies. As preprocessing techniques help contribute to a reduced vocabulary, they should also help alleviate this scaling problem, at least according to folklore. However, to the best of our knowledge, no previous study of preprocessing techniques have examined how they contribute to reduced run-time costs, leading to uncertainty about what these techniques do to mitigate the computational complexity in practice.

To remedy this, we analyze how these prepro-

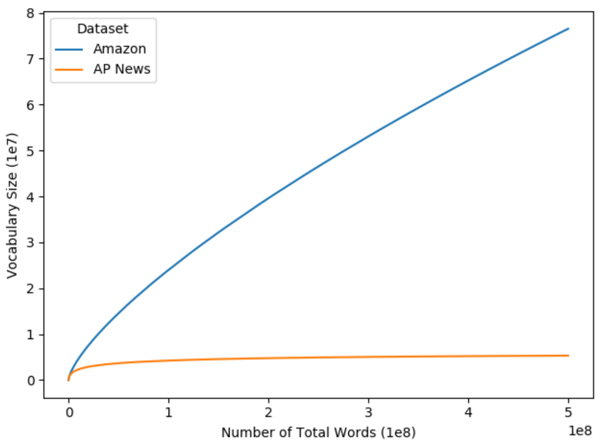

Figure 1: Comparing vocabulary size (in millions) vs the total number of words (in 10s of millions) for the AP News and Amazon corpora. Note that the vocabulary size of AP News w.r.t. the number of documents plateaus much faster than the noisier Amazon corpus.

cessing methods affect not only vocabulary size and performance, but also how they affect training and inference time. To do this, we contribute a comprehensive analysis of 10 different preprocessing methods applied to four machine learning models, evaluated on two datasets with widely varying vocabularies (Figure 1).

Our results show that the individual preprocessing methods provide widely different effects on run-time, with some methods (i.e. rare word filtering and stopword removal) providing significant run-time reductions without losing any performance. We also show that some combinations of preprocessing methods both improve performance and reduce run-time.

2 Experimental Setup

Datasets To see how preprocessing affects run-time, we examine two datasets (in English): the Amazon (He and McAuley, 2016)[2] and AP News corpora (MacIntyre, 1998). These datasets were chosen because of the wide disparity between their vocabularies. The Amazon corpus comes from user product reviews and contains a much higher vocabulary relative to the number of documents, due to its noisy text. The AP News corpus contains professionally-edited news articles and its vocabulary plateaus much faster than the Amazon corpus (Figure 1). We perform sentiment analysis on Amazon and year classification on AP News and report scores with the accuracy metric. We note that we also computed the F1 score alongside accuracy and found our results to be similar; thus we report accuracy since it is easier to understand.

To test the effect of document size on preprocessing, we sampled various-sized[3] datasets from the original corpus and ran our analysis on each, sampling 5 different times with differing random seeds.[4] However, we found that our results were nearly identical across the differing corpus sizes and thus, only report numbers for the 100k size.

Preprocessing Methods We analyze 10 different methods (with their shortened names in parenthesis): lowercasing (lower), rare word filtering (rare), hashing (hash), punctuation removal (nopunct), stopword removal (stop), number removal (nrem), word stemming (stem), lemmatization (lemma), spelling correction (spell), and word segmentation (seg). We choose these methods because of their prevalence in previous work (Symeonidis et al., 2018; Kumar and Harish, 2018; HaCohen-Kerner et al., 2020) and their use in industry (Li et al., 2013; Sanchez-Pi et al., 2014).

Due to the exponential number of possible preprocessing combinations, we run all individual methods but restrict the search space of combinations of these methods. For rare word filtering and word hashing, we first conduct experiments for 9 different levels of filtering individually, using only the best level in future combinations with other methods. Results for all levels of filtering and hashing are in Appendices A and B. We then conduct experiments for all 24 combinations of spelling correction, word segmentation, number removal, and stopword removal, using the best outcome (the pipeline of all four) to combine with other methods. We note that while this is not an exhaustive search of all combinations, our analysis includes the standard preprocessing pipelines as well as many more.

Models We use Scikit-Learn (Pedregosa et al., 2011) for three of the base algorithms, including K-NN (Altman, 1992), Naive Bayes (Rish et al., 2001), and the Support Vector Machine (SVM, (Suykens and Vandewalle, 1999)). We also employ Vowpal Wabbit (Langford et al., 2007; Karampatziakis and Langford, 2010), due to its strong performance and frequent use in industry.

[2]http://jmcauley.ucsd.edu/data/amazon/

[3]5k, 10k, 20k, 30k, 40k, 50k, and 100k instances

[4]Although the Amazon corpus contains many more documents, we keep our sampling consistent with the AP News corpus, as AP News has only 600k instances.

Group	Method	Vocab Size ↓	Train Time ↓	Test Time ↓	Accuracy ↑
Individual Methods	stop	99.8 ± 0.2	**69.5** ± 1.4	79.0 ± 3.5	97.4 ± 2.2
	rare	**1.0** ± 0.0	80.6 ± 3.0	**70.3** ± 3.2	**99.3** ± 2.8
	seg	24.6 ± 0.2	93.7 ± 2.4	80.3 ± 2.5	**100.6** ± 1.6
	spell	57.8 ± 0.2	95.1 ± 2.6	89.7 ± 2.6	**99.4** ± 2.3
	hash	10.1 ± 0.0	97.1 ± 4.0	75.7 ± 4.0	**99.2** ± 1.1
	nopunct	61.9 ± 0.2	97.5 ± 2.2	89.5 ± 2.0	**100.7** ± 1.6
	stem	81.7 ± 0.4	97.8 ± 2.0	95.0 ± 2.6	**99.8** ± 1.0
	lower	88.7 ± 0.3	101.7 ± 7.5	100.1 ± 6.6	**99.1** ± 3.0
	nrem	96.2 ± 0.7	101.7 ± 4.0	100.7 ± 5.3	**99.7** ± 1.2
	lemma	98.1 ± 0.5	102.2 ± 5.3	101.5 ± 5.1	**100.3** ± 1.1
Lowest Train/Test Time	spell+seg+nrem+stop+rare	**0.8** ± 0.0	**44.6** ± 1.0	56.8 ± 1.1	95.4 ± 2.0
	stop+rare	0.9 ± 0.0	**46.5** ± 3.5	44.5 ± 2.0	**99.8** ± 0.8
	spell+seg+nrem+stop+hash	7.6 ± 0.0	53.9 ± 2.0	**39.6** ± 1.4	**97.7** ± 2.6
	spell+seg+nrem+stop	14.1 ± 0.0	54.2 ± 1.6	50.9 ± 2.3	**97.6** ± 2.2
	spell+seg+nrem+stop+lemma	11.9 ± 0.0	55.1 ± 0.9	50.1 ± 1.7	97.6 ± 1.3
Highest Accuracy	nopunct+rare	**0.9** ± 0.0	**82.6** ± 1.9	**70.2** ± 1.7	**101.0** ± 1.8
	lower+nopunct+nrem+rare	**0.9** ± 0.0	**87.1** ± 5.5	88.8 ± 5.1	101.1 ± 0.3
	lower+nopunct+rare	**0.9** ± 0.0	86.1 ± 2.7	86.7 ± 2.2	**101.3** ± 0.6
	seg+rare	**0.9** ± 0.0	**86.3** ± 5.8	**73.3** ± 3.9	**101.4** ± 1.5
	spell+seg+rare	**0.9** ± 0.0	89.6 ± 5.6	88.4 ± 5.3	**101.8** ± 0.5

Table 1: Effect of preprocessing techniques on Amazon. Scores are the relative performance of each method over the *no preprocessing* baseline (e.g. stopword removal takes only 69.5% of the baseline's training time). Results are the average (and std) relative performance of the four models, across the five dataset seeds. Bold indicates statistical similarity to the best score, from a two-sample t-test with $\alpha = 0.05$. For brevity, this table only includes individual methods and those with the highest accuracy or lowest train/test time. All results are in Appendix C.

All models use default hyperparameters and our document representations use the BoW representation, consisting of a sparse vector format. These four models provide a wide range of algorithms that might be used, allowing us to show how preprocessing methods generalize across models.

Compute All experiments were performed using 14-core Intel Broadwell processors running at 2.4GHz with 128GB of DDR4 2400 MT/s RAM.

3 Results

We format our results relative to the algorithm with no preprocessing, to easily show how preprocessing changes this baseline performance. We first run each algorithm with no preprocessing, measuring the run-time, vocabulary size, and accuracy. We then report the scores of each preprocessing pipeline relative to the algorithm's baseline (e.g. a model with preprocessing that scores 75% of the no-preprocessing baseline's accuracy has a relative accuracy of 0.75).

As the cross product of the number of methods vs. the number of models is still far too large to include in this paper, we show the average of each model's relative proportion to its respective baseline performance.[5] This aggregation shows us the average relative performance across the four models, helping us generalize our results to be model-independent. For full tables detailing specific model results, see Appendix C. Bold scores in tables indicate statistical similarity to the best score in the column (two-sample t-test, $\alpha = 0.05$).

Individual Techniques We see results for the Amazon corpus in Table 1 and for the AP News corpus in Table 2. On Amazon, each individual preprocessing method performs statistically similar to the baseline's accuracy, while three algorithms (stopword removal, rare word filtering, and word segmentation) also provide a moderate decrease (20-30%) in train and test time. Rare word filtering and stopword removal are effective across both corpora (with rare word filtering being even more effective on AP News, reducing the training time in half), while the other methods do not significantly impact either train-time or accuracy on AP News. We hypothesize that these techniques are more effective on the AP corpus because of its much smaller (and less varied) vocabulary.

[5]We first compute each algorithm's relative score to its baseline (e.g. SVM with rare word filtering vs SVM with no preprocessing) and then take the average of the models for that method (e.g. average the relative performance of rare word filtering on models {K-NN, Naive Bayes, SVM, and Vowpal Wabbit} for the final score for rare word filtering).

Group	Method	Vocab Size ↓	Train Time ↓	Test Time ↓	Accuracy ↑
Individual Methods	rare	**0.1** ± 0.0	**51.4** ± 1.2	**59.1** ± 2.2	**99.8** ± 2.0
	stop	99.5 ± 0.3	82.5 ± 2.9	86.4 ± 2.0	**99.0** ± 1.4
	hash	32.8 ± 0.0	98.5 ± 4.7	84.7 ± 4.0	**99.5** ± 1.7
	spell	65.6 ± 0.2	98.6 ± 4.7	95.2 ± 7.4	**99.7** ± 0.9
	lower	92.3 ± 0.3	98.9 ± 1.9	97.6 ± 3.5	**99.8** ± 1.6
	stem	82.7 ± 0.4	99.1 ± 5.2	95.2 ± 4.0	**100.1** ± 1.4
	seg	47.5 ± 0.2	99.5 ± 2.4	88.5 ± 2.9	**100.3** ± 1.3
	nrem	89.8 ± 0.4	99.8 ± 3.9	98.5 ± 5.0	**99.2** ± 1.1
	nopunct	65.6 ± 0.2	99.9 ± 4.6	92.9 ± 4.9	**99.6** ± 1.4
	lemma	97.4 ± 0.3	100.5 ± 1.2	98.6 ± 1.6	**99.6** ± 1.7
Lowest Train/Test Time	spell+seg+nrem+stop+rare	**0.1** ± 0.0	**29.2** ± 0.5	**49.0** ± 0.8	**99.3** ± 1.7
	spell+nopunct+nrem+stop	39.9 ± 0.1	71.0 ± 1.8	69.4 ± 1.8	**100.1** ± 1.0
	spell+nopunct+nrem+stop+lemma	36.2 ± 0.1	71.3 ± 1.3	68.4 ± 1.4	**99.1** ± 1.6
	spell+seg+nrem+stop	29.3 ± 0.0	72.1 ± 1.8	65.4 ± 3.2	**100.1** ± 1.4
	spell+nopunct+nrem+stop+stem	29.6 ± 0.1	72.4 ± 1.2	68.2 ± 1.8	**98.7** ± 1.8
Highest Accuracy	spell+seg+nrem+stop+stem	**19.3** ± 0.1	74.2 ± 2.3	**66.9** ± 1.5	**99.7** ± 1.6
	spell+nopunct+nrem+stop	39.9 ± 0.1	**71.0** ± 1.8	69.4 ± 1.8	**100.1** ± 1.0
	spell+seg+nrem+stop	29.3 ± 0.0	**72.1** ± 1.8	**65.4** ± 3.2	**100.1** ± 1.4
	spell+seg+nrem+stop+hash	19.8 ± 0.0	**73.1** ± 2.7	**66.6** ± 3.4	**100.2** ± 1.5
	lower+nopunct+nrem+stop+stem	39.4 ± 0.4	75.5 ± 2.4	72.7 ± 3.0	**100.3** ± 1.0

Table 2: Effect of preprocessing techniques on AP News. Scores are the relative performance of each method over the *no preprocessing* baseline (e.g. stopword removal takes only 82.5% of the baseline's training time). Results are the average (and std) relative performance of the four models, across the five dataset seeds. Bold indicates statistical similarity to the best score, from a two-sample t-test with $\alpha = 0.05$. For brevity, this table only includes individual methods and those with the highest accuracy or lowest train/test time. All results are in Appendix C.

Combination Techniques The combination techniques also show a mild impact on accuracy, with most methods on both corpora performing statistically similar to the baseline. On the Amazon corpus, a handful of methods trade 2-5% of accuracy for up to a 65% reduction in training and testing time ("Lowest Train/Test Time" section in Table 1). Those that do not reduce accuracy (such as stop+rare) can still reduce the training and testing time by up to 55%. We see in the "Highest Accuracy" section that some methods (i.e. spell+seg+rare, etc.) can even improve performance by almost 2% while also reducing run-time by 10-15%. Similarly, when we examine the results on AP News we can find combinations with reduced run-time (up to 70% and 50% reductions in train and test time respectively) with no accuracy loss (but also no gains).

Correlations In order to show the correlation between run-time and the other variables, we show a heatmap of these correlations in Figure 2. Most of these variables are highly correlated with each other, as expected (training time is highly correlated with testing time, etc.). However, although testing time is highly correlated with vocabulary size (0.8 correlation), training time is not highly correlated (0.17), We hypothesize that a low vocabulary directly leads to faster inference, while which words are removed from the vocabulary has a bigger role in how quickly the algorithm converges during training. This hypothesis is also supported by the low correlation between vocabulary size and accuracy, indicating that what is in the vocabulary is more important than its size.

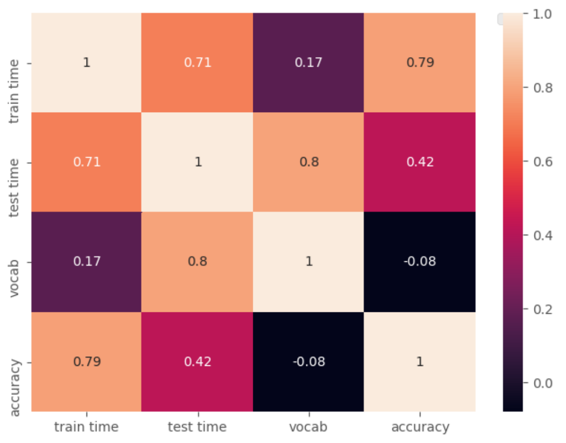

Figure 2: Pearson correlation between the relative performance variables (train time, test time, accuracy, and vocabulary size) from the results of the different preprocessing methods.

4 Related Work

These experiments relate to a large body of work that considers how preprocessing methods affect the downstream accuracy of various algorithms, ranging from topics in information retrieval (Chaudhari et al., 2015; Patil and Atique, 2013; Beil et al., 2002), text classification and regression (Forman, 2003; Yang and Pedersen, 1997; Vijayarani et al., 2015; Kumar and Harish, 2018; HaCohen-Kerner et al., 2020; Symeonidis et al., 2018; Weller et al., 2020), topic modeling (Blei et al., 2003; Lund et al., 2019; Schofield and Mimno, 2016; Schofield et al., 2017a,b), and even more complex tasks like question answering (Jijkoun et al., 2003; Carvalho et al., 2007) and machine translation (Habash, 2007; Habash and Sadat, 2006; Leusch et al., 2005; Weller et al., 2021; Mehta et al., 2020) to name a few. With the rise of noisy social media, text preprocessing has become important for tasks that use data from sources like Twitter and Reddit (Symeonidis et al., 2018; Singh and Kumari, 2016; Bao et al., 2014; Jianqiang, 2015; Weller and Seppi, 2020; Zirikly et al., 2019; Babanejad et al., 2020).

The closest lines of work to ours are those that examine how preprocessing affects text classification accuracy, where recent works like Symeonidis et al. (2018) and HaCohen-Kerner et al. (2020) analyze and cross-compare up to 16 different techniques for four machine learning algorithms. In contrast, our work is the first to examine these preprocessing techniques beyond accuracy, examining them in tandem with how they affect vocabulary size and run-time.

5 Conclusion

In this work we conduct the first study that examines the relationship between vocabulary size, run-time, and accuracy across different models and corpora for text classification. In general, we find that although vocabulary size is highly correlated with testing time, it is not highly correlated with training time or accuracy. In these cases, the specifics of the preprocessing algorithm (the content of what it removes) matter more.

Our experiments show that rare word filtering and stopword removal are superior to many other common preprocessing methods, both in terms of their ability to reduce run-time and their potential to increase accuracy. By using these methods, we show that it is possible to reduce training and testing time by up to 65% with a loss of only 2-5% of accuracy, or in some cases, to provide accuracy and run-time improvements simultaneously. We hope that this study can help both researchers and industry practitioners as they design machine learning pipelines to reach their end-goals.

References

Naomi S Altman. 1992. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185.

Nastaran Babanejad, Ameeta Agrawal, Aijun An, and Manos Papagelis. 2020. A comprehensive analysis of preprocessing for word representation learning in affective tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5799–5810.

Yanwei Bao, Changqin Quan, Lijuan Wang, and Fuji Ren. 2014. The role of pre-processing in twitter sentiment analysis. *Intelligent Computing Methodologies Lecture Notes in Computer Science*, page 615–624.

Florian Beil, Martin Ester, and Xiaowei Xu. 2002. Frequent term-based text clustering. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 436–442. Association for Computing Machinery.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022.

Cornelia Caragea, Adrian Silvescu, and Prasenjit Mitra. 2012. Combining hashing and abstraction in sparse high dimensional feature spaces. In *Association for the Advancement of Artificial Intelligence*.

Gracinda Carvalho, David Martins de Matos, and Vitor Rocio. 2007. Document retrieval for question answering: a quantitative evaluation of text preprocessing. In *Proceedings of the ACM first Ph. D. workshop in CIKM*, pages 125–130.

Anagha Chaudhari, Pravin M Phadatare, Pranil S Kudale, Raviraj B Mohite, Rohan P Petare, Yogesh P Jagdale, and Amitabh Mudiraj. 2015. Preprocessing of high dimensional dataset for developing expert ir system. In *2015 International Conference on Computing Communication Control and Automation*, pages 417–421. IEEE.

George Forman. 2003. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3.

Nizar Habash. 2007. Syntactic preprocessing for statistical machine translation. *MT Summit XI*, pages 215–222.

Nizar Habash and Fatiha Sadat. 2006. Arabic preprocessing schemes for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 49–52.

Yaakov HaCohen-Kerner, Daniel Miller, and Yair Yigal. 2020. The influence of preprocessing on text classification using a bag-of-words representation. *PloS one*, 15(5):e0232525.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, pages 507–517. International World Wide Web Conferences Steering Committee.

Zhao Jianqiang. 2015. Pre-processing boosting twitter sentiment analysis? *2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*.

Valentin Jijkoun, Gilad Mishne, Maarten de Rijke, et al. 2003. Preprocessing documents to answer dutch questions. In *Proceedings of the 15th Belgian-Dutch Conference on Artificial Intelligence (BNAIC'03)*, volume 434. Citeseer.

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

Nikos Karampatziakis and John Langford. 2010. Online Importance Weight Aware Updates. *arXiv e-prints*, page arXiv:1011.1576.

HM Keerthi Kumar and BS Harish. 2018. Classification of short text using various preprocessing techniques: An empirical evaluation. In *Recent Findings in Intelligent Computing Techniques*, pages 19–30. Springer.

John Langford, Lihong Li, and Alex Strehl. 2007. Vowpal wabbit online learning project.

Gregor Leusch, Nicola Ueffing, David Vilar, and Hermann Ney. 2005. Preprocessing and normalization for automatic evaluation of machine translation. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 17–24.

Huiying Li, Qiang Ye, and Rob Law. 2013. Determinants of customer satisfaction in the hotel industry: An application of online review analysis. *Asia Pacific Journal of Tourism Research*, 18(7):784–802.

J. Lund, Piper Armstrong, Wilson Fearn, Stephen Cowley, Emily Hales, and K. Seppi. 2019. Cross-referencing using fine-grained topic modeling. In *NAACL-HLT*.

Robert MacIntyre. 1998. North american news text supplement. *LDC98T30, Linguistic Data Consortium*.

Sneha Mehta, Bahareh Azarnoush, Boris Chen, Avneesh Saluja, Vinith Misra, Ballav Bihani, and Ritwik Kumar. 2020. Simplify-then-translate: Automatic preprocessing for black-box machine translation. *arXiv preprint arXiv:2005.11197*.

Leena H Patil and Mohammed Atique. 2013. A novel approach for feature selection method tf-idf in document clustering. In *2013 3rd IEEE International Advance Computing Conference (IACC)*, pages 858–862. Institute of Electrical and Electronics Engineers.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Irina Rish et al. 2001. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46.

Nayat Sanchez-Pi, Luis Martí, and Ana Cristina Bicharra Garcia. 2014. Text classification techniques in oil industry applications. In *International Joint Conference SOCO'13-CISIS'13-ICEUTE'13*, pages 211–220. Springer.

Alexandra Schofield, Måns Magnusson, and D Mimno. 2017a. Understanding text pre-processing for latent dirichlet allocation. In *Proceedings of the 15th conference of the European Chapter of the Association for Computational Linguistics*, volume 2, pages 432–436.

Alexandra Schofield, Måns Magnusson, and David Mimno. 2017b. Pulling out the stops: Rethinking stopword removal for topic models. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*.

Alexandra Schofield and David Mimno. 2016. Comparing apples to apple: The effects of stemmers on topic models. *Transactions of the Association for Computational Linguistics*, 4:287–300.

Hajime Senuma. 2011. K-means clustering with feature hashing. *Proceedings of the ACL-HLT*, page 122–126.

Tajinder Singh and Madhu Kumari. 2016. Role of text pre-processing in twitter sentiment analysis. *Procedia Computer Science*, 89:549–554.

Johan AK Suykens and Joos Vandewalle. 1999. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300.

Symeon Symeonidis, Dimitrios Effrosynidis, and Avi Arampatzis. 2018. A comparative evaluation of preprocessing techniques and their interactions for twitter sentiment analysis. *Expert Systems with Applications*, 110:298–310.

Alper Kursat Uysal and Serkan Gunal. 2014. The impact of preprocessing on text classification. *Information Processing & Management*, 50(1):104–112.

S Vijayarani, Ms J Ilamathi, and Ms Nithya. 2015. Preprocessing techniques for text mining-an overview. *International Journal of Computer Science & Communication Networks*, 5(1):7–16.

Orion Weller, J. Hildebrandt, I. Reznik, Christopher Challis, E. S. Tass, Q. Snell, and K. Seppi. 2020. You don't have time to read this: An exploration of document reading time prediction. In *ACL*.

Orion Weller and K. Seppi. 2020. The rjokes dataset: a large scale humor collection. In *LREC*.

Orion Weller, Matthias Sperber, Christian Gollan, and Joris Kluivers. 2021. Streaming models for joint speech recognition and translation. *arXiv preprint arXiv:2101.09149*.

Yiming Yang and J. Pedersen. 1997. A comparative study on feature selection in text categorization. In *ICML*.

Yin Zhang, Rong Jin, and Z. Zhou. 2010. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1:43–52.

Ayah Zirikly, Philip Resnik, Ozlem Uzuner, and Kristy Hollingshead. 2019. Clpsych 2019 shared task: Predicting the degree of suicide risk in reddit posts. In *Proceedings of the sixth workshop on computational linguistics and clinical psychology*, pages 24–33.

A Rare Word Filtering

Tables 3 and 4 show the results of rare word filtering on the Amazon and AP News datasets. We filtered at levels corresponding to the geometric progression of values from 1 to half the size of the corpus (we refer to these as levels 1 to 9, with higher numbers being more filtered).

We find that rare word filtering at higher levels provides increased vocabulary and run-time reductions, while also reducing accuracy, in general.

B Word Hashing

Tables 5 and 6 show the effect of different levels of word hashing on model accuracy (where "Size" indicates the number of hash buckets used). We find that word hashing with small numbers of buckets reduces vocabulary and run-time, while also decreasing accuracy in general.

C Full Tables for Method Combinations

In Tables 7 and 8 we show the complete table for preprocessing method combinations on Amazon and AP News respectively.

In Tables 9, 10, 11, and 12, we show the complete results for individuals models (K-NN, Naive Bayes, Vowpal Wabbit, and SVM respectively). All results are similar to the main conclusions found in the body of the paper. However, Naive Bayes in particular shows strong accuracy gains and run-time reductions for preprocessing methods, in comparison to other models.

#	Vocab Size	Train Time	Test Time	Accuracy
9	**0.0** ± 0.0	**40.2** ± 1.4	**51.8** ± 2.0	92.8 ± 0.7
8	0.2 ± 0.0	64.6 ± 2.5	64.0 ± 2.8	96.9 ± 1.1
7	1.0 ± 0.0	80.6 ± 3.0	70.3 ± 3.2	**99.3** ± 2.8
6	4.4 ± 0.0	87.4 ± 2.7	72.4 ± 2.7	**100.3** ± 1.8
5	16.4 ± 0.0	92.4 ± 3.6	76.2 ± 3.8	**99.7** ± 2.4
4	41.4 ± 0.1	94.8 ± 1.9	83.2 ± 2.4	**100.0** ± 2.1
3	63.3 ± 0.1	98.1 ± 3.2	90.8 ± 3.1	**99.7** ± 1.8
1	100 ± 0.2	100.8 ± 2.5	100.1 ± 2.0	**100.0** ± 1.1
2	77.5 ± 0.2	101.4 ± 5.2	97.1 ± 5.9	**100.3** ± 1.9

Table 3: Rare word filtering on the Amazon dataset, across various levels. Scores are the relative performance of each method over the *no preprocessing* baseline. Results are the average (and std) relative performance of the four models, across the five dataset seeds. Bold indicates statistical similarity to the best score, from a two-sample t-test with $\alpha = 0.05$.

#	Vocab Size	Train Time	Test Time	Accuracy
9	**0.0** ± 0.0	**33.9** ± 1.2	**61.1** ± 3.0	**99.3** ± 1.2
8	0.1 ± 0.0	51.4 ± 1.2	**59.1** ± 2.2	**99.8** ± 2.0
7	1.1 ± 0.0	69.6 ± 2.0	68.0 ± 2.0	**100.0** ± 1.5
6	7.2 ± 0.0	83.8 ± 3.7	76.6 ± 5.7	**99.5** ± 0.7
5	31.8 ± 0.0	94.0 ± 4.8	83.0 ± 3.7	**99.4** ± 1.4
4	76.9 ± 0.1	98.2 ± 3.4	96.0 ± 5.5	**99.7** ± 1.3
1	100.0 ± 0.3	99.4 ± 3.3	98.2 ± 2.7	**99.8** ± 1.3
2	99.7 ± 0.2	100.3 ± 4.3	99.2 ± 3.9	**99.6** ± 1.1
3	96.2 ± 0.2	100.6 ± 4.3	98.7 ± 2.4	**100.0** ± 0.9

Table 4: Rare word filtering on the AP News dataset, across various levels. This table is formatted the same as Table 3 (see the caption there for more information).

Size	Vocab Size	Train Time	Test Time	Accuracy
500	**0.1** ± 0.0	**92.7** ± 3.7	**76.9** ± 3.1	91.7 ± 1.7
60000	15.1 ± 0.0	**96.1** ± 1.5	76.7 ± 2.4	**99.3** ± 1.0
10000	2.5 ± 0.0	**96.2** ± 1.6	**73.2** ± 2.2	**97.9** ± 1.3
1000	0.3 ± 0.0	**96.2** ± 4.1	**76.5** ± 3.5	93.5 ± 1.6
40000	10.1 ± 0.0	**97.1** ± 4.0	**75.7** ± 4.0	**99.2** ± 1.1
4000	1.0 ± 0.0	97.3 ± 1.1	**75.0** ± 1.0	96.7 ± 1.6
20000	5.0 ± 0.0	**97.6** ± 3.5	**74.6** ± 3.1	**98.7** ± 1.2
8000	2.0 ± 0.0	**98.2** ± 6.8	**74.5** ± 4.8	**97.9** ± 1.1
2000	0.5 ± 0.0	99.2 ± 3.7	77.2 ± 2.6	95.1 ± 1.6
6000	1.5 ± 0.0	100.6 ± 6.0	**76.7** ± 4.9	97.1 ± 1.5

Table 5: Word Hashing on the Amazon dataset, across various levels. Scores are the relative performance of each method over the *no preprocessing* baseline. Results are the average (and std) relative performance of the four models, across the five dataset seeds. Bold indicates statistical similarity to the best score, from a two-sample t-test with $\alpha = 0.05$.

Size	Vocab Size	Train Time	Test Time	Accuracy
500	**0.4** ± 0.0	**93.5** ± 2.5	81.4 ± 1.8	**99.5** ± 1.3
1000	0.9 ± 0.0	**94.7** ± 2.7	**79.9** ± 2.6	**99.8** ± 1.0
4000	3.5 ± 0.0	**94.8** ± 2.8	**77.5** ± 2.0	**99.4** ± 1.2
10000	8.7 ± 0.0	**95.4** ± 3.1	**78.9** ± 3.9	**98.9** ± 1.8
6000	5.2 ± 0.0	**95.6** ± 2.9	**78.5** ± 1.8	**99.0** ± 0.9
2000	1.7 ± 0.0	**96.2** ± 4.8	**79.9** ± 4.0	**98.9** ± 1.2
20000	17.3 ± 0.0	**96.6** ± 3.4	81.2 ± 2.4	**99.1** ± 1.4
8000	7.0 ± 0.0	**96.8** ± 5.0	**80.3** ± 4.7	**99.0** ± 1.8
40000	32.8 ± 0.0	**98.5** ± 4.7	84.7 ± 4.0	**99.5** ± 1.7
60000	44.5 ± 0.1	**99.0** ± 5.3	88.5 ± 6.3	**99.1** ± 1.3

Table 6: Word Hashing on the AP News dataset, across various levels. This table is formatted the same as Table 5 (see the caption there for more information).

Method	Vocab Size	Train Time	Test Time	Accuracy
spell+seg+nrem+stop+rare	**0.8** ± 0.0	**44.6** ± 1.0	56.8 ± 1.1	95.4 ± 2.0
stop+rare	0.9 ± 0.0	**46.5** ± 3.5	44.5 ± 2.0	99.8 ± 0.8
spell+seg+nrem+stop+hash	7.6 ± 0.0	53.9 ± 2.0	**39.6** ± 1.4	97.7 ± 2.6
spell+seg+nrem+stop	14.1 ± 0.0	54.2 ± 1.6	50.9 ± 2.3	97.6 ± 2.2
spell+seg+nrem+stop+lemma	11.9 ± 0.0	55.1 ± 0.9	50.1 ± 1.7	97.6 ± 1.3
seg+nrem+stop+lemma	18.5 ± 0.1	55.1 ± 2.3	54.3 ± 4.3	96.3 ± 3.7
spell+nopunct+nrem+stop+lemma	31.9 ± 0.2	55.3 ± 1.5	56.7 ± 1.8	96.9 ± 1.2
spell+nopunct+nrem+stop	34.1 ± 0.2	55.8 ± 1.6	56.4 ± 2.1	97.8 ± 1.6
seg+nrem+stop	21.0 ± 0.1	55.9 ± 1.3	52.5 ± 1.6	97.5 ± 1.4
lower+nopunct+nrem+stop+lemma	47.0 ± 0.3	56.0 ± 2.2	61.2 ± 1.8	96.0 ± 1.7
lower+nopunct+nrem+stop	48.4 ± 0.3	56.5 ± 1.3	59.4 ± 1.8	96.0 ± 2.3
seg+nrem+stop+stem	14.0 ± 0.1	58.5 ± 1.5	56.8 ± 1.8	97.9 ± 0.7
spell+seg+nrem+stop+stem	8.5 ± 0.0	58.7 ± 1.6	54.8 ± 2.3	97.8 ± 1.4
lower+nopunct+nrem+stop+stem	39.3 ± 0.2	59.0 ± 1.6	63.2 ± 2.9	96.2 ± 2.4
spell+nopunct+nrem+stop+stem	27.7 ± 0.1	59.1 ± 2.6	61.4 ± 4.0	97.5 ± 1.3
stop	99.8 ± 0.2	69.5 ± 1.4	79.0 ± 3.5	97.4 ± 2.2
lower+rare	1.0 ± 0.0	80.5 ± 2.0	69.6 ± 1.7	**99.8** ± 3.1
rare	1.0 ± 0.0	80.6 ± 3.0	70.3 ± 3.2	**99.3** ± 2.8
spell+rare	0.9 ± 0.0	80.7 ± 2.2	70.9 ± 2.6	**99.7** ± 1.8
stem+rare	0.9 ± 0.0	81.4 ± 2.2	69.9 ± 1.7	99.4 ± 1.6
nrem+rare	1.0 ± 0.0	82.0 ± 5.6	70.1 ± 3.2	99.6 ± 1.2
nopunct+rare	0.9 ± 0.0	82.6 ± 1.9	70.2 ± 1.7	**101.0** ± 1.8
lemma+rare	1.0 ± 0.0	82.7 ± 6.5	69.6 ± 1.8	**100.1** ± 1.5
lower+nopunct+rare	0.9 ± 0.0	86.1 ± 2.7	86.7 ± 2.2	**101.3** ± 0.6
seg+rare	0.9 ± 0.0	86.3 ± 5.8	73.3 ± 3.9	**101.4** ± 1.5
lower+nopunct+nrem+rare	0.9 ± 0.0	87.1 ± 5.5	88.8 ± 5.1	101.1 ± 0.3
spell+seg+rare	0.9 ± 0.0	89.6 ± 5.6	88.4 ± 5.3	**101.8** ± 0.5
seg	24.6 ± 0.2	93.7 ± 2.4	80.3 ± 2.5	**100.6** ± 1.6
spell	57.8 ± 0.2	95.1 ± 2.6	89.7 ± 2.6	**99.4** ± 2.3
hash	10.1 ± 0.0	97.1 ± 4.0	75.7 ± 4.0	99.2 ± 1.1
nopunct	61.9 ± 0.2	97.5 ± 2.2	89.5 ± 2.0	**100.7** ± 1.6
stem	81.7 ± 0.4	97.8 ± 2.0	95.0 ± 2.6	99.8 ± 1.0
lower	88.7 ± 0.3	101.7 ± 7.5	100.1 ± 6.6	**99.1** ± 3.0
nrem	96.2 ± 0.7	101.7 ± 4.0	100.7 ± 5.3	99.7 ± 1.2
lemma	98.1 ± 0.5	102.2 ± 5.3	101.5 ± 5.1	100.3 ± 1.1

Table 7: Full results of preprocessing methods on Amazon. Scores are the relative performance of each method over the *no preprocessing* baseline. Results are the average (and std) relative performance of the four models, across the five dataset seeds. Bold indicates statistical similarity to the best score, from a two-sample t-test with $\alpha = 0.05$.

Method	Vocab Size	Train Time	Test Time	Accuracy
spell+seg+nrem+stop+rare	**0.1** ± 0.0	**29.2** ± 0.5	**49.0** ± 0.8	**99.3** ± 1.7
rare	**0.1** ± 0.0	51.4 ± 1.2	59.1 ± 2.2	**99.8** ± 2.0
spell+nopunct+nrem+stop	39.9 ± 0.1	71.0 ± 1.8	69.4 ± 1.8	**100.1** ± 1.0
spell+nopunct+nrem+stop+lemma	36.2 ± 0.1	71.3 ± 1.3	68.4 ± 1.4	**99.1** ± 1.6
spell+seg+nrem+stop	29.3 ± 0.0	72.1 ± 1.8	65.4 ± 3.2	**100.1** ± 1.4
spell+nopunct+nrem+stop+stem	29.6 ± 0.1	72.4 ± 1.2	68.2 ± 1.8	**98.7** ± 1.8
spell+seg+nrem+stop+hash	19.8 ± 0.0	73.1 ± 2.7	66.6 ± 3.4	**100.2** ± 1.5
spell+seg+nrem+stop+lemma	25.4 ± 0.1	73.6 ± 1.9	67.6 ± 1.7	**99.4** ± 1.5
lower+nopunct+nrem+stop+lemma	49.1 ± 0.1	73.7 ± 1.3	72.8 ± 2.0	98.4 ± 1.4
seg+nrem+stop	40.7 ± 0.2	74.1 ± 3.4	73.1 ± 3.3	**99.5** ± 1.3
spell+seg+nrem+stop+stem	19.3 ± 0.1	74.2 ± 2.3	66.9 ± 1.5	**99.7** ± 1.6
lower+nopunct+nrem+stop	51.4 ± 0.2	74.2 ± 3.5	74.0 ± 4.4	**99.6** ± 1.3
seg+nrem+stop+lemma	36.4 ± 0.1	75.2 ± 2.1	68.1 ± 3.5	**99.1** ± 1.2
lower+nopunct+nrem+stop+stem	39.4 ± 0.4	75.5 ± 2.4	72.7 ± 3.0	**100.3** ± 1.0
seg+nrem+stop+stem	29.3 ± 0.1	76.9 ± 2.1	63.9 ± 1.5	**99.4** ± 1.7
stop	99.5 ± 0.3	82.5 ± 2.9	86.4 ± 2.0	**99.0** ± 1.4
hash	32.8 ± 0.0	98.5 ± 4.7	84.7 ± 4.0	**99.5** ± 1.7
spell	65.6 ± 0.2	98.6 ± 4.7	95.2 ± 7.4	**99.7** ± 0.9
lower	92.3 ± 0.3	98.9 ± 1.9	97.6 ± 3.5	**99.8** ± 1.6
stem	82.7 ± 0.4	99.1 ± 5.2	95.2 ± 4.0	**100.1** ± 1.4
seg	47.5 ± 0.2	99.5 ± 2.4	88.5 ± 2.9	**100.3** ± 1.3
nrem	89.8 ± 0.4	99.8 ± 3.9	98.5 ± 5.0	**99.2** ± 1.1
nopunct	65.6 ± 0.2	99.9 ± 4.6	92.9 ± 4.9	**99.6** ± 1.4
lemma	97.4 ± 0.3	100.5 ± 1.2	98.6 ± 1.6	**99.6** ± 1.7

Table 8: Full results of preprocessing methods on AP News. Scores are the relative performance of each method over the *no preprocessing* baseline. Results shown are the average (and std) relative performance of the four models, across the five dataset seeds. Bold indicates statistical similarity to the best score, from a two-sample t-test with $\alpha = 0.05$.

Method	Vocab Size	Train Time	Test Time	Accuracy
spell+seg+nrem+stop+rare	**0.8** ± 0.0	**47.0** ± 0.8	**46.7** ± 1.0	92.3 ± 3.3
seg+nrem+stop+lemma	18.5 ± 0.1	56.7 ± 0.9	52.2 ± 1.6	**89.3** ± 12.1
lower+nopunct+nrem+stop+lemma	47.0 ± 0.3	56.8 ± 1.2	50.9 ± 1.0	90.9 ± 4.2
spell+seg+nrem+stop	14.1 ± 0.0	57.0 ± 0.9	**48.2** ± 1.5	92.2 ± 6.8
spell+seg+nrem+stop+lemma	11.9 ± 0.0	57.2 ± 1.1	53.8 ± 0.9	92.8 ± 2.7
spell+seg+nrem+stop+hash	7.6 ± 0.0	57.2 ± 1.4	48.3 ± 0.8	92.1 ± 6.3
spell+nopunct+nrem+stop+lemma	31.9 ± 0.2	57.5 ± 1.4	52.9 ± 1.1	92.0 ± 2.2
seg+nrem+stop	21.0 ± 0.1	57.8 ± 0.7	**47.7** ± 1.2	94.7 ± 2.5
spell+nopunct+nrem+stop	34.1 ± 0.2	58.2 ± 1.6	48.4 ± 0.8	95.5 ± 3.3
lower+nopunct+nrem+stop	48.4 ± 0.3	58.8 ± 1.8	**47.6** ± 1.3	89.2 ± 6.0
lower+nopunct+nrem+stop+stem	39.3 ± 0.2	59.6 ± 1.2	61.7 ± 4.2	88.0 ± 5.8
spell+nopunct+nrem+stop+stem	27.7 ± 0.1	60.5 ± 2.2	62.8 ± 4.3	95.1 ± 2.5
spell+seg+nrem+stop+stem	8.5 ± 0.0	61.4 ± 2.0	62.4 ± 4.7	94.2 ± 2.8
seg+nrem+stop+stem	14.0 ± 0.1	61.8 ± 2.0	65.2 ± 0.8	94.2 ± 0.8
stop	99.8 ± 0.2	69.3 ± 1.8	52.8 ± 3.6	**93.4** ± 6.6
rare	1.0 ± 0.0	80.5 ± 1.5	96.4 ± 0.5	**95.5** ± 9.3
lower+rare	1.0 ± 0.0	82.7 ± 2.2	100.6 ± 2.4	**93.6** ± 8.8
spell+rare	0.9 ± 0.0	82.9 ± 1.4	104.2 ± 6.1	94.0 ± 4.2
lemma+rare	1.0 ± 0.0	83.2 ± 3.2	99.3 ± 1.8	**97.4** ± 3.5
stem+rare	0.9 ± 0.0	84.4 ± 1.9	101.1 ± 2.2	95.8 ± 3.1
nopunct+rare	0.9 ± 0.0	85.3 ± 2.7	101.0 ± 2.6	**99.4** ± 4.4
nrem+rare	1.0 ± 0.0	91.1 ± 16.3	102.7 ± 8.7	95.6 ± 2.7
spell	57.8 ± 0.2	95.9 ± 2.7	103.8 ± 3.3	**96.4** ± 5.7
seg+rare	0.9 ± 0.0	96.7 ± 14.5	112.5 ± 9.7	**100.6** ± 3.5
seg	24.6 ± 0.2	98.0 ± 3.5	105.0 ± 2.7	**99.3** ± 3.0
stem	81.7 ± 0.4	99.0 ± 1.6	102.4 ± 1.9	**98.5** ± 2.0
lower	88.7 ± 0.3	99.4 ± 2.9	102.9 ± 2.8	**96.0** ± 6.5
hash	10.1 ± 0.0	99.7 ± 1.3	100.7 ± 1.9	**99.4** ± 1.2
lemma	98.1 ± 0.5	101.2 ± 2.5	101.9 ± 2.9	**100.7** ± 1.3
nopunct	61.9 ± 0.2	101.6 ± 3.5	103.5 ± 3.0	**100.1** ± 3.9
nrem	96.2 ± 0.7	102.1 ± 1.5	101.7 ± 2.4	**99.7** ± 1.8

Table 9: Effect of preprocessing techniques on Amazon with the K-NN model. Scores are the relative performance of each method over the *no preprocessing* baseline. Results shown are the average (and std) relative performance across the five dataset seeds. Bold indicates statistical similarity to the best score, from a two-sample t-test with $\alpha = 0.05$.

Method	Vocab Size	Train Time	Test Time	Accuracy
stop+rare	**0.9** ± 0.0	**44.7** ± 5.4	**5.8** ± 0.2	**104.5** ± 0.8
spell+seg+nrem+stop+hash	7.6 ± 0.0	**49.7** ± 1.4	12.3 ± 0.0	**103.5** ± 0.8
spell+seg+nrem+stop	14.1 ± 0.0	**50.2** ± 1.0	18.7 ± 0.6	**104.4** ± 0.2
spell+seg+nrem+stop+lemma	11.9 ± 0.0	**50.4** ± 0.4	16.6 ± 0.3	**104.0** ± 0.4
seg+nrem+stop+lemma	18.5 ± 0.1	52.0 ± 2.3	22.7 ± 0.6	**103.7** ± 0.5
spell+nopunct+nrem+stop+lemma	31.9 ± 0.2	52.1 ± 0.7	34.8 ± 0.3	103.1 ± 0.9
spell+nopunct+nrem+stop	34.1 ± 0.2	52.8 ± 0.8	36.9 ± 0.4	102.3 ± 1.0
lower+nopunct+nrem+stop+lemma	47.0 ± 0.3	53.1 ± 1.4	49.1 ± 0.8	101.7 ± 1.4
seg+nrem+stop	21.0 ± 0.1	53.1 ± 1.6	25.3 ± 1.0	**103.6** ± 1.7
spell+seg+nrem+stop+stem	8.5 ± 0.0	53.3 ± 2.3	13.3 ± 0.3	102.7 ± 0.9
spell+nopunct+nrem+stop+stem	27.7 ± 0.1	54.2 ± 0.5	31.1 ± 0.4	101.1 ± 1.4
seg+nrem+stop+stem	14.0 ± 0.1	54.3 ± 0.8	18.7 ± 0.3	102.9 ± 0.7
lower+nopunct+nrem+stop	48.4 ± 0.3	54.5 ± 0.7	50.2 ± 0.6	101.5 ± 1.5
lower+nopunct+nrem+stop+stem	39.3 ± 0.2	55.6 ± 1.2	41.9 ± 0.4	101.5 ± 1.3
rare	1.0 ± 0.0	69.0 ± 0.8	6.5 ± 0.1	100.6 ± 1.2
lower+rare	1.0 ± 0.0	69.9 ± 1.5	6.4 ± 0.0	**102.3** ± 1.9
stop	99.8 ± 0.2	69.9 ± 1.2	99.2 ± 1.2	101.1 ± 0.7
nrem+rare	1.0 ± 0.0	70.3 ± 1.2	6.5 ± 0.1	101.6 ± 0.9
spell+rare	**0.9** ± 0.0	70.5 ± 0.9	6.4 ± 0.1	101.5 ± 0.9
stem+rare	**0.9** ± 0.0	70.7 ± 1.7	6.4 ± 0.2	100.4 ± 1.7
lemma+rare	1.0 ± 0.0	71.2 ± 0.8	6.5 ± 0.2	100.7 ± 0.7
nopunct+rare	**0.9** ± 0.0	73.2 ± 1.6	6.5 ± 0.2	101.2 ± 0.7
seg+rare	**0.9** ± 0.0	73.3 ± 2.2	6.5 ± 0.1	101.5 ± 1.2
seg	24.6 ± 0.2	87.2 ± 1.6	29.1 ± 0.3	101.0 ± 0.7
hash	10.1 ± 0.0	90.1 ± 1.1	15.4 ± 0.2	99.3 ± 1.5
spell	57.8 ± 0.2	90.4 ± 1.7	59.9 ± 0.5	99.6 ± 1.3
nopunct	61.9 ± 0.2	94.8 ± 2.4	64.1 ± 0.8	100.6 ± 0.8
stem	81.7 ± 0.4	96.0 ± 1.0	82.7 ± 1.0	99.8 ± 1.2
lower	88.7 ± 0.3	97.7 ± 2.3	89.9 ± 1.5	100.4 ± 0.9
lemma	98.1 ± 0.5	99.5 ± 1.1	98.4 ± 0.9	100.8 ± 1.2
nrem	96.2 ± 0.7	99.8 ± 1.5	97.0 ± 1.6	99.6 ± 1.2

Table 10: Effect of preprocessing techniques on Amazon with the Naive Bayes model. Scores are the relative performance of each method over the *no preprocessing* baseline. Results shown are the average (and std) relative performance across the five dataset seeds. Bold indicates statistical similarity to the best score, from a two-sample t-test with $\alpha = 0.05$.

Method	Vocab Size	Train Time	Test Time	Accuracy
spell+seg+nrem+stop+rare	**0.8** ± 0.0	**42.1** ± 1.5	**46.6** ± 1.4	97.0 ± 1.3
stop+rare	0.9 ± 0.0	47.7 ± 4.0	**51.4** ± 4.1	97.5 ± 0.9
lower+nopunct+nrem+stop	48.4 ± 0.3	53.3 ± 1.6	56.1 ± 1.2	97.5 ± 0.8
spell+seg+nrem+stop	14.1 ± 0.0	54.3 ± 3.2	57.2 ± 3.3	97.9 ± 0.9
spell+nopunct+nrem+stop+lemma	31.9 ± 0.2	54.4 ± 2.3	57.9 ± 2.7	97.3 ± 1.2
seg+nrem+stop+lemma	18.5 ± 0.1	54.5 ± 4.0	59.9 ± 9.5	97.3 ± 1.2
lower+nopunct+nrem+stop+lemma	47.0 ± 0.3	54.9 ± 5.7	58.0 ± 4.8	96.7 ± 0.7
spell+nopunct+nrem+stop	34.1 ± 0.2	54.9 ± 3.1	57.8 ± 2.8	97.9 ± 0.8
spell+seg+nrem+stop+hash	7.6 ± 0.0	55.0 ± 3.0	58.3 ± 3.4	97.5 ± 0.7
seg+nrem+stop	21.0 ± 0.1	55.4 ± 2.3	58.5 ± 2.5	97.2 ± 0.8
seg+nrem+stop+stem	14.0 ± 0.1	55.5 ± 1.8	57.8 ± 2.0	98.1 ± 0.6
spell+seg+nrem+stop+stem	8.5 ± 0.0	57.2 ± 0.8	59.6 ± 0.9	98.1 ± 0.9
lower+nopunct+nrem+stop+stem	39.3 ± 0.2	57.2 ± 3.1	59.9 ± 3.2	98.4 ± 1.4
spell+nopunct+nrem+stop+stem	27.7 ± 0.1	58.9 ± 5.9	61.9 ± 4.6	98.1 ± 0.5
stop	99.8 ± 0.2	66.9 ± 1.5	69.2 ± 2.4	97.9 ± 0.7
nrem+rare	1.0 ± 0.0	78.5 ± 3.5	79.1 ± 3.4	**100.6** ± 0.6
spell+rare	0.9 ± 0.0	79.1 ± 3.6	79.7 ± 2.8	**101.7** ± 1.1
lemma+rare	1.0 ± 0.0	79.8 ± 4.4	80.0 ± 3.9	**101.5** ± 1.0
stem+rare	0.9 ± 0.0	79.9 ± 3.3	80.2 ± 3.0	**100.8** ± 0.8
lower+rare	1.0 ± 0.0	80.2 ± 2.8	80.5 ± 2.3	**101.7** ± 0.7
nopunct+rare	0.9 ± 0.0	80.5 ± 2.2	80.7 ± 2.1	**101.8** ± 1.2
lower+nopunct+rare	0.9 ± 0.0	80.9 ± 3.6	81.7 ± 3.5	**101.4** ± 0.7
seg+rare	0.9 ± 0.0	82.9 ± 4.8	83.4 ± 5.1	**101.8** ± 0.5
rare	1.0 ± 0.0	84.1 ± 9.0	84.9 ± 8.3	**100.6** ± 0.3
lower+nopunct+nrem+rare	0.9 ± 0.0	84.4 ± 9.5	85.7 ± 9.1	**101.3** ± 0.2
spell+seg+rare	0.9 ± 0.0	86.1 ± 9.4	86.9 ± 8.9	**102.1** ± 0.5
seg	24.6 ± 0.2	94.5 ± 1.9	94.6 ± 2.6	**102.2** ± 1.4
nopunct	61.9 ± 0.2	95.3 ± 2.6	95.8 ± 2.0	**101.9** ± 0.5
spell	57.8 ± 0.2	98.6 ± 5.0	98.1 ± 6.0	**101.5** ± 1.0
stem	81.7 ± 0.4	98.9 ± 4.6	97.7 ± 4.3	**100.9** ± 0.5
hash	10.1 ± 0.0	102.8 ± 12.2	102.1 ± 12.3	**100.5** ± 0.8
nrem	96.2 ± 0.7	104.3 ± 11.9	103.3 ± 11.4	100.0 ± 0.8
lower	88.7 ± 0.3	107.9 ± 17.2	107.7 ± 15.5	**100.9** ± 1.7
lemma	98.1 ± 0.5	109.4 ± 16.3	107.8 ± 14.9	100.0 ± 1.0

Table 11: Effect of preprocessing techniques on Amazon with the Vowpal Wabbit model. Results shown are the average (and std) relative performance, across the five dataset seeds. Bold indicates statistical similarity to the best score, from a two-sample t-test with $\alpha = 0.05$.

Method	Vocab Size	Train Time	Test Time	Accuracy
spell+seg+nrem+stop+rare	**0.8** ± 0.0	**44.7** ± 0.6	**77.1** ± 0.9	96.7 ± 1.4
stop+rare	0.9 ± 0.0	47.1 ± 1.1	**76.2** ± 1.6	97.4 ± 0.8
spell+seg+nrem+stop	14.1 ± 0.0	55.4 ± 1.5	**79.3** ± 3.8	95.7 ± 0.7
seg+nrem+stop+lemma	18.5 ± 0.1	57.1 ± 1.8	**82.2** ± 5.6	94.9 ± 1.0
spell+nopunct+nrem+stop	34.1 ± 0.2	57.2 ± 0.8	82.7 ± 4.4	95.7 ± 1.1
seg+nrem+stop	21.0 ± 0.1	57.4 ± 0.5	**78.5** ± 1.7	94.5 ± 0.4
spell+nopunct+nrem+stop+lemma	31.9 ± 0.2	57.4 ± 1.5	81.4 ± 3.2	95.1 ± 0.6
spell+seg+nrem+stop+lemma	11.9 ± 0.0	57.6 ± 1.3	**80.1** ± 3.9	96.1 ± 0.9
lower+nopunct+nrem+stop+lemma	47.0 ± 0.3	59.3 ± 0.6	86.7 ± 0.6	94.8 ± 0.5
lower+nopunct+nrem+stop	48.4 ± 0.3	59.5 ± 1.3	83.7 ± 4.1	95.7 ± 0.9
seg+nrem+stop+stem	14.0 ± 0.1	62.6 ± 1.3	85.7 ± 4.2	96.6 ± 0.6
spell+seg+nrem+stop+stem	8.5 ± 0.0	62.7 ± 1.2	84.1 ± 3.2	96.2 ± 1.1
spell+nopunct+nrem+stop+stem	27.7 ± 0.1	62.8 ± 1.6	89.8 ± 6.8	95.9 ± 0.7
lower+nopunct+nrem+stop+stem	39.3 ± 0.2	63.5 ± 0.9	89.5 ± 4.0	96.8 ± 1.3
stop	99.8 ± 0.2	71.9 ± 1.3	94.6 ± 6.9	97.3 ± 0.8
nrem+rare	1.0 ± 0.0	88.3 ± 1.5	92.1 ± 0.7	100.4 ± 0.5
rare	1.0 ± 0.0	88.7 ± 0.9	93.6 ± 4.0	100.5 ± 0.5
lower+rare	1.0 ± 0.0	89.1 ± 1.4	90.7 ± 1.8	**101.5** ± 0.8
lower+nopunct+nrem+rare	0.9 ± 0.0	89.9 ± 1.5	92.0 ± 1.2	**100.9** ± 0.3
spell+rare	0.9 ± 0.0	90.3 ± 2.8	93.3 ± 1.5	**101.5** ± 1.2
stem+rare	0.9 ± 0.0	90.5 ± 2.0	91.9 ± 1.6	**100.7** ± 0.7
nopunct+rare	0.9 ± 0.0	91.3 ± 1.1	92.6 ± 1.9	**101.6** ± 0.8
lower+nopunct+rare	0.9 ± 0.0	91.3 ± 1.9	91.8 ± 0.8	**101.2** ± 0.6
seg+rare	0.9 ± 0.0	92.4 ± 1.9	90.9 ± 0.7	**101.6** ± 0.6
spell+seg+rare	0.9 ± 0.0	93.2 ± 1.8	89.9 ± 1.6	**101.5** ± 0.6
seg	24.6 ± 0.2	95.0 ± 2.6	92.7 ± 4.5	**100.0** ± 1.5
spell	57.8 ± 0.2	95.6 ± 1.2	97.0 ± 0.8	99.9 ± 1.2
hash	10.1 ± 0.0	95.6 ± 1.4	84.7 ± 1.5	97.7 ± 0.8
lemma+rare	1.0 ± 0.0	96.9 ± 17.6	92.5 ± 1.4	**101.0** ± 0.7
stem	81.7 ± 0.4	97.3 ± 0.8	97.4 ± 3.2	99.9 ± 0.2
nopunct	61.9 ± 0.2	98.5 ± 0.4	94.8 ± 2.1	100.2 ± 1.0
lemma	98.1 ± 0.5	98.7 ± 1.5	97.9 ± 1.8	99.8 ± 0.8
nrem	96.2 ± 0.7	100.8 ± 1.2	100.9 ± 5.9	99.5 ± 1.0

Table 12: Effect of preprocessing techniques on Amazon with the SVM model. Scores are the relative performance of each method over the *no preprocessing* baseline. Results shown are the average (and std) relative performance across the five dataset seeds. Bold indicates statistical similarity to the best score, from a two-sample t-test with $\alpha = 0.05$.