textPrep: A Text Preprocessing Toolkit for Topic Modeling on Social Media Data

Rob Churchill and Lisa Singh
Georgetown University, U.S.A.

Keywords: Text Preprocessing, Topic Modeling, Data Science, Social Media, textPrep.

Abstract: With the rapid growth of social media in recent years, there has been considerable effort toward understanding the topics of online discussions. Unfortunately, state of the art topic models tend to perform poorly on this new form of data, due to their noisy and unstructured nature. There has been a lot of research focused on improving topic modeling algorithms, but very little focused on improving the quality of the data that goes into the algorithms. In this paper, we formalize the notion of preprocessing configurations and propose a standardized, modular toolkit and pipeline for performing preprocessing on social media texts for use in topic [redacted]

Figure 1 shows examples of topics identified from tweets by state of the art topic models during the 2016 US Presidential election. When the entire tweet is used as input into a topic modeling algorithm (the first three word clouds in Figure 1), we see that the topics contain stopwords, hashtags, user handles, plural words, and even misspellings. The last word cloud (bottom right) uses preprocessed tweets and does not contain the same amount of noise. We can determine that it is about Trump refusing to release his tax returns. While a great deal of effort has been spent creating topic models with social media data in mind, little attention has been paid to the impact of preprocessing decisions made prior to generating topic models.

Researchers have found that many traditional state

[redacted] a document collection $\mathcal{D}$, for each document D_i in $\mathcal{D}$, we tokenize D_i on whitespace to get a series of n tokens $D_i = \{d_1, d_2, \ldots, d_n\}$. Tokens may be terms, punctuation, numbers, web addresses, emojis, etc. We ask two questions. First, which tokens should be removed prior to topic model creation? Second, how can we determine if we have done a good job preprocessing? To help systematically conduct preprocessing and assess the effectiveness of different preprocessing decisions, we present textPrep, a toolkit for preprocessing text data. Second, to demonstrate its value and the importance of preprocessing, we identify preprocessing rules and arrange these rules into preprocessing configurations that generate different data sets for use by topic modeling algorithms.

We find that preprocessing has significant effects

Churchill, R. and Singh, L.
textPrep: A Text Preprocessing Toolkit for Topic Modeling on Social Media Data.
DOI: 10.5220/0010559000600070
In *Proceedings of the 10th International Conference on Data Science, Technology and Applications (DATA 2021)*, pages 60-70
ISBN: 978-989-758-521-0

on topic model performance, but that models and data sets are not equally affected by the same amounts and types of preprocessing. Some models and data sets are more positively affected than others, and in some cases, preprocessing can hurt model performance. In general, for our case studies, doing more thorough preprocessing helps model performance far more than it hurts. Finally, we find that while certain preprocessing methods can appear to produce similar quality data sets, the quality of topics that are generated on these data sets can diverge quickly for less apt configurations. Our hope is that by building an easy to use toolkit and demonstrating the impact of certain preprocessing rules and configurations on the quality of topics generated by state of the art topic modeling algorithms on noisy social media data sets, more data scientists and researchers will add preprocessing analysis to their topic modeling pipeline, thereby en- [redacted] formalize a preprocessing taxonomy that combines useful preprocessing rules and configurations, 3) we propose a simple preprocessing methodology that applies configurations of rules to document tokens to generate better quality data sets that can be used by topic modeling algorithms, 4) we conduct extensive empirical case studies of preprocessing configurations on three large social media data sets, and evaluate the data quality and topic quality of each configuration using three different topic models, and 5) we summarize our findings through a set of best practices that will help those less familiar with topic modeling determine which approaches to use with which algorithms.

2 RELATED LITERATURE

Preprocessing. In the early 2000s, there were a handful of papers related to data preprocessing from the database community that focused on enabling users to better understand the quality of their data set (Vassiliadis et al., 2000; Raman and Hellerstein, 2001), and describing data quality issues focused on storage and pruning (Rahm and Do, 2000; Knoblock et al., 2003). More recently, researchers have shown the impact of preprocessing on text classification (Srividhya and Anitha, 2010; **?**). Allahyari et al. mention text preprocessing in their survey of text mining, but do not evaluate any methods (Allahyari et al., 2017). Our work considers a much larger set of preprocessing approaches and focuses on an unsupervised topic modeling task as opposed to a supervised text classification task. Denny and Spirling analyze the ef- [redacted]

Topic Models. There are many types of topic models ranging from generative to graph-based, unsupervised, semi-supervised, and supervised. In this paper, we focus on the most widely used type, the unsupervised generative topic model.

The most prevalent topic model in the unsupervised generative class of models is Latent Dirichlet Allocation (Blei et al., 2003). LDA has inspired the vast majority of generative models since its inception. It uses a bag-of-words model, with the goal of finding the parameters of the topic/term distribution that maximizes the likelihood of documents in the data set over k topics. LDA has inspired the vast majority of other generative models, including HDP (Teh et al., 2006), DTM (Blei and Lafferty, 2006), CTM (Lafferty and Blei, 2006), Twitter-LDA (Zhao et al., 2011), Authorless Topic Models (Thompson and Mimno, 2018), and Topics over Time (Wang and McCallum, 2006).

Another model, Dirichlet Multinomial Model (DMM) (Nigam et al., 2000), also known as the mixture of unigrams model, was conceived before LDA and differs in one main aspect. Whereas LDA works under the assumption that every document is generated from a distribution of topics, DMM is simpler; it assumes that each document is generated from a single topic. While LDA's ability to generate documents from a mixture of topics is superior for most traditional types of documents such as books, research papers, and newspaper articles, the simplicity of DMM's generation makes it well-suited for use in social media posts, which are much smaller and therefore more likely to truly be generated from a single topic. DMM was improved, optimized, and brought back to life by Yin and Wang in 2014 (Yin and Wang, 2014).

[redacted]

ding space are incremented as well. This creates a "rising tide lifts all boats" effect, raising the likelihood of words similar to the sampled word, and the coherence of topics.

In our study, we use LDA (Blei et al., 2003), DMM (Yin and Wang, 2014), and GPUDMM (Li et al., 2016). These three were selected because they represent the different generative approaches well. LDA is the traditional, ubiquitous topic model, DMM represents a social media-tailored approach, and GPUDMM represents the newer word-embedding aided methods.

3 textPrep: PYTHON PREPROCESSING TOOLKIT

To encourage more consistent preprocessing for topic models, we have created a Python preprocessing toolkit for Topic Modeling (textPrep).[1] The toolkit includes each preprocessing rule we use to create our configurations, as well as a streamlined pipeline for creating other configurations. The toolkit takes advantage of other standard text processing libraries, including NLTK (Bird et al., 2009), and Gensim (Řehůřek and Sojka, 2010). The rules can be easily added to configurations, or used standalone on a single document or a whole data set. Furthermore, because rule modules are designed to be used on a single document, they are ready out of the box for use in preprocessing text using PySpark pipelines.

The preprocessing pipeline is designed in a modu-

[redacted]

a given data set or experiment. It is important to balance data quality metrics. High average token frequency only matters if there is still a considerably high vocabulary size – we do not want a data set with a small number of very frequent unique tokens, nor do we want a data set with a million very infrequent unique tokens. Attaining better data quality can save time and resources before ever running topic models.

[1] textPrep can be found at https://github.com/GU-DataLab/topic-modeling-textPrep

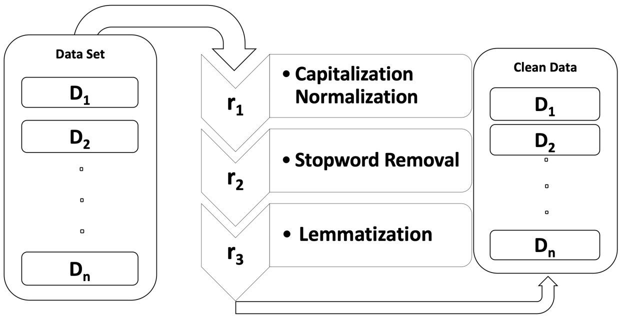

Figure 2: The Preprocessing Pipeline.

4 PREPROCESSING CLASSES AND RULES

A *preprocessing rule*, r_ℓ, is an operation that changes [redacted] Description), and also gives a simple example of how each preprocessing rule works.

Elementary Pattern-based Preprocessing: focuses on reducing the number of tokens (e.g. punctuation removal) and the variation (e.g. capitalization normalization) in tokens by searching for known patterns in tokens that may indicate noise in the context of topic identification. It also includes rules that join existing tokens to improve the semantic quality of the token (e.g. n-gram creation). Typically, these rules are implemented using regular expressions. Two rules - the hashtag removal and the user removal rules - within the elementary pattern-based preprocessing category are specific to Twitter data sets since both have special meanings on that platform. The rules in this category tend to be the basic, standard ones that researchers generally apply.

Dictionary-based Preprocessing: focuses on using a predefined dictionary (typically manually created) to remove tokens (e.g. stopword removal), stan- [redacted]

Statistical Preprocessing: computes statistics about tokens using information about the entire collection to determine tokens that should be maintained or removed. In this class of preprocessing, we consider two rules: collection term frequency cleaning and TF-IDF cleaning. Collection term frequency cleaning refers to removing terms that have a particularly low frequency in a data set (minimum DF), or a particularly high one (maximum DF). TF-IDF (Term-Frequency, Inverse Document Frequency) looks at term relevance in a collection and removes tokens with a low relevance.

Defining this preprocessing taxonomy provides us with a second way to interpret the collection of rules that are most and least beneficial for topic modeling preprocessing.

Table 1: Preprocessing Rules.

Rule	Rule Description	Example Tokens
Elementary Pattern-based Preprocessing		
URL Removal	Removal of tokens containing URLs	Removed Token: http://aaa.com/index.html
Capitalization Normalization	Make all tokens lowercase or uppercase	Original Token: Tree Final Token: tree
Punctuation Removal	Removal of all tokens that are punctuation	Removed Token: !
Hashtag Removal	Removal of tokens beginning with the hashtag (#) sign	Removed Token: #ilovechocolate
User Removal	Removal of tokens beginning with the @ sign	Removed Token: @hillary
Malformed Word Removal	Removal of tokens accidentally created because of other rules	Removed Token: http
N-gram Creation	N tokens are joined together to form a new token	Original: i love cats Bigrams: {i love}, {love cats}
Dictionary-based Preprocessing		
Stopword Removal	Removal of common words that do not add content value	Removed Token: the, is, am
Emoji Removal	Removal of emoji and emoticon tokens	Removed Token: :)
Synonym Matching	Replace tokens that match a synonym in a given dictionary	Synonym: obama=barack obama= barack
Whitelist Cleaning	Retain only tokens that appear on a pre-created list	Whitelist: ['covid', 'masks', 'vaccine', 'pandemic']
Natural Language Preprocessing		
Lemmatization	Shorten a token down to its lemma using NLP	Original Token: better, Final Token: good
Stemming	Shorten a token down to its base by removal of suffixes	Original Token: giving, Final Token: giv
Part of Speech (POS) Removal	Removal of tokens that are tagged as a certain part of speech	Remove all adjectives
Statistical Preprocessing		
Collection Term Frequency Cleaning	Removal of tokens with a low frequency count in the data set	Remove tokens that appear less than α times
TF-IDF Cleaning	Removal of tokens with a low TF-IDF score	Remove tokens with a TF-IDF score below β

[redacted]

al. (Singh et al., 2020). This data set contains over 500,000 tweets.

The second data set, collected from Reddit, contains posts about the 2020 United States Presidential Election. This data set was collected using the pushshift.io library (Pushshift.io, 2021). Reddit posts were collected from subreddits related to U.S. politics and the election from September to election day on the first week of November 2020. This data set has over 1 million posts.

The third data set contains comments collected from the Hacker News platform (Moody, 2016). Hacker News is a technology and entrepreneurship news site that allows users to comment and discuss articles. Collected by Moody for testing lda2vec (Moody, 2016), comments were only col-

[redacted]

library SciKit Learn (Buitinck et al., 2013). It entails tokenization, punctuation removal, capitalization normalization, stopword removal, and the removal of tokens that appear in less than five documents.[2] Our lightweight configuration consists of the following rules: 1. URL removal, 2. Punctuation removal, 3. Capitalization normalization, 4. Stopword removal.

The difference between the lightweight configuration and the second baseline is that we drop frequency thresholding and introduce of URL removal. The heavyweight configuration consists of all of the rules in the lightweight configuration, plus: 1. Short word removal, 2. Lemmatization, 3. N-gram creation.

[2]The minimum number of documents varies by author, but usually lies between 2 and 10.

Table 2: Data Set Properties.

Data Set	# Docs	# Tokens	Tokens/Doc	Stopwords/Doc	Unique Tokens	Token Frequency
Hacker News	1,165,421	80,604,631	69.16	28.95	1,613,253	49.96
Reddit	1,022,481	28,947,427	28.31	11.54	296,132	97.75
Twitter	565,182	12,826,812	22.70	6.46	558,189	22.98

For n-gram creation, we used a minimum frequency of 512 for n-grams to replace their component words. To choose the threshold for n-gram creation, we tested values ranging from 64 to 1024 (powers of 2), and found that 512 offered the best balance between speed and number of n-grams created. If the threshold is set too low, it will take too long to create n-grams and too many n-grams will be created. If the threshold is set too high, few or no n-grams will be created. Both configurations for the Twitter data set, which we consider to be a special case, also include hashtag removal.

[redacted] fore and after certain preprocessing steps. A higher average frequency of tokens indicates that preprocessing rules are successfully removing noisy or less frequent tokens without having to use a minimum frequency threshold such as in baseline 2. Furthermore, smaller vocabularies coupled with higher average token frequency make for better topic modeling conditions. A smaller vocabulary (filled with good content words) means that less memory is required to train a topic model. Topic models also have fewer words to choose from, meaning that they will be less likely to make mistakes. A higher average token frequency means that relationships between words can be more easily reinforced in the topic-word distribution, leading to more accurate and coherent topics.

Extrinsic Evaluation Methods: We use the downstream task of topic modeling to evaluate the effect of preprocessing on data quality. We use LDA (Blei et al., 2003),[3] DMM (Yin and Wang, 2014), and GPUDMM (Li et al., 2016).[4] These three topic models represent different approaches to generative topic modeling, as discussed in Section 2. In order to evaluate the quality of topic modeling results, we use topic coherence and topic diversity.

Topic coherence, or a model's ability to produce easily interpreted topics, can be computed using normalized pointwise mutual information (NPMI) (Lau et al., 2014). NPMI uses word co-occurrences to cap- [redacted] els, we employ topic diversity. Topic diversity is the fraction of unique words in the top 20 words of all topics in a topic set (Dieng et al., 2019b; Churchill and Singh, 2020). High topic diversity indicates that a model was successful in finding unique topics, while low diversity indicates that a model found a small number of topics multiple times.

5.4 Reddit Case Study

Table 3 shows the data quality statistics for each configuration on the Reddit data set. The 'Avg. Freq.'

[3] specifically the Mallet implementation of LDA (McCallum, 2002)

[4] the implementations from the Short Text Topic Model survey (Qiang et al., 2019)

Table 3: Data Quality Statistics for each preprocessing configuration on the Reddit data set.

Configuration	# Tokens	Unique Tokens	Avg. Freq.
Baseline 1	28,947,427	296,132	97.75
Baseline 2	15,267,929	246,307	61.99
Lightweight	14,053,743	**51,399**	**273.42**
Heavyweight	**11,776,937**	326,874	36.02

column shows the average frequency of a token in the data set after being preprocessed using the configuration. The Reddit data quality shows that the configuration with the smallest vocabulary is the lightweight configuration. This seems counter-intuitive at first, because the heavyweight configuration includes the entire lightweight configuration, and the heavyweight configuration has over two million fewer tokens in total than the lightweight configuration.

The difference is n-gram creation. N-gram creation is one of the few preprocessing rules that adds

[redacted]

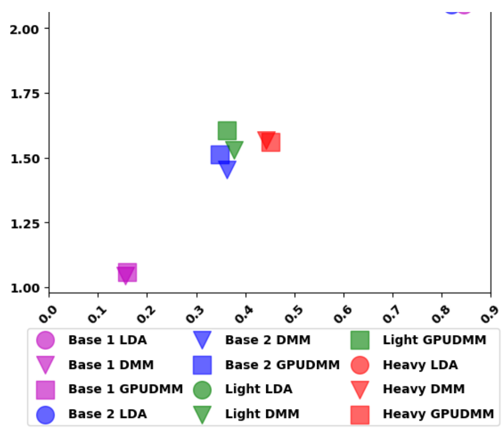

Figure 3: Topic Coherence (y) and Diversity (x) Scores on the Reddit Data Set.

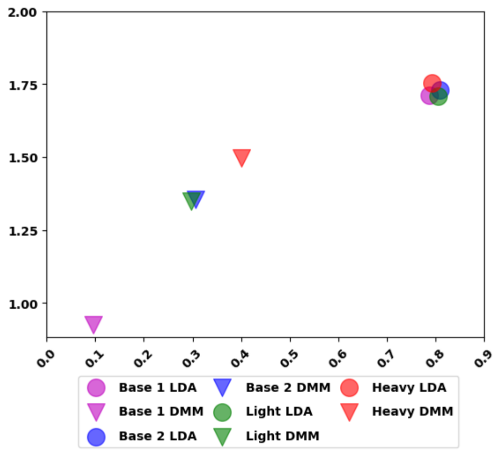

Figure 4: Topic Coherence (y) and Diversity (x) Scores on the Hacker News Data Set.

resented by triangles, and GPUDMM is represented

[redacted]

Table 4 shows the data quality statistics for each configuration on the Hacker News data set. Hacker News, despite having only about 140,000 more documents than the Reddit data set, has over 80 million tokens, making documents over 69 tokens on average. This difference in initial data size leads to different results in data quality after configurations are applied. The heavyweight configuration can only reduce the total number of tokens to 34 million, still over five million more tokens than the unprocessed Reddit data set. Second, the lightweight configuration fails to significantly lower the number of unique tokens compared to baseline 2. In fact, the two configurations lead to nearly identical data quality statistics.

Figure 4 shows the topic coherence and diversity scores for LDA and DMM on each preprocessing configuration. GPUDMM is not shown because,

Table 4: Data Quality Statistics for each preprocessing configuration on the Hacker News data set.

Configuration	# Tokens	Unique Tokens	Avg. Freq.
Baseline 1	80,604,631	1,613,253	49.96
Baseline 2	39,943,951	135,754	294.24
Lightweight	39,991,122	**134,609**	**297.09**
Heavyweight	**34,374,771**	1,196,609	28.72

Table 5: Data Quality Statistics for each preprocessing configuration on the Twitter data set.

Configuration	# Tokens	Unique Tokens	Avg. Freq.
Baseline 1	12,826,812	558,189	22.98
Baseline 2	7,983,422	505,170	15.80
Lightweight	7,270,421	**62,879**	**115.63**
Heavyweight	**6,323,070**	337,741	18.72

as a word embedding aided model that relies heavily on memory, it failed to complete on a server with 77GB of memory due to the size of the Hacker

[redacted]

figuration. Again, the heavyweight configuration produces the least number of total tokens, but a large vocabulary (although not as large as in Hacker News). Figure 5 shows the topic coherence and diversity scores for each topic model and configuration combination. We see similar results to those of Reddit, indicating that they have similar characteristics relative to Hacker News. However, in the Twitter data set, there is a clear benefit to thorough preprocessing for every model including LDA. Every configuration improves over baseline 1 in terms of coherence for LDA and both metrics for DMM and GPUDMM.

In order to show the flexibility of the preprocessing pipeline, we delved deeper into configurations for the Twitter data set. What if we could reduce the size of the vocabulary to a number similar to that of the

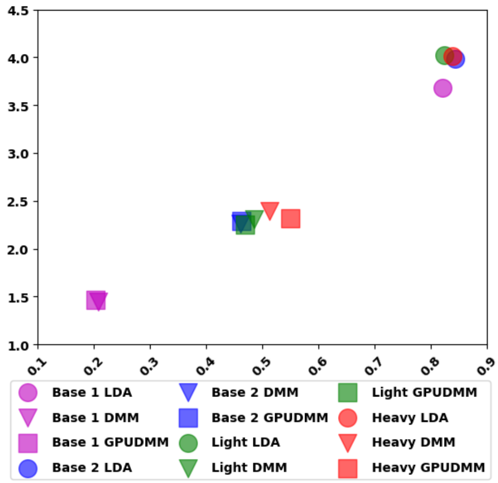

Figure 5: Topic Coherence (y) and Diversity (x) Scores on the Twitter Data Set.

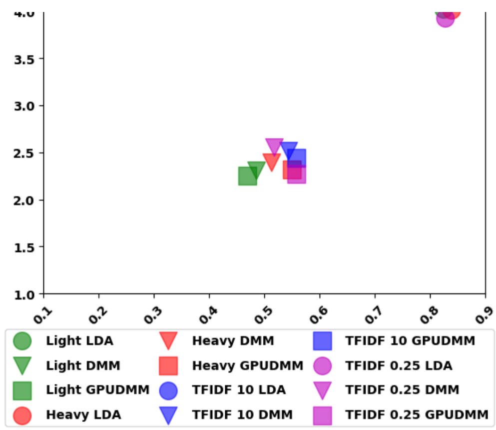

Figure 6: Topic Coherence (y) and Diversity (x) Scores on Twitter Data Set, using TF-IDF rule.

rule with a threshold of 10, 1, 0.5, and 0.25.

Table 6 shows the data quality metrics of each of these configurations compared to the original heavyweight configuration. The thresholds of 10, 1, and 0.5 were all too high, and produced very small vocabularies compared to the lightweight configuration. However, the threshold of 0.25 produced a vocabulary that is similar to that of the lightweight. The total number of tokens is similar for thresholds between 1 and 0.25, so the real difference in data quality exists between threshold 10 and the rest. We can see that while the threshold of 0.25 produces a similar size vocabulary to the lightweight configuration, its average token frequency is about 20% lower. With the preprocessing pipeline, we were able to quickly tailor the data qualities to our desired levels, allowing us to get to topic modeling faster.

Figure 6 shows the results when using the TF- [redacted] that there is truly one best configuration. Data sets can be preprocessed with a set of safe preprocessing rules, but there might be a better configuration out there that offers some significant improvements in model performance. As we saw in the Twitter Case Study, the best configuration might not be one of a few likely choices. In comparing the Hacker News data set to the Reddit and Twitter data sets, we found that what is best for one data set is not necessarily the best for the next data set. However, if we need select a "general purpose" model, LDA typically performs better than DMM and GPUDMM. This is surprising given that the latter two models should in theory be better suited for short texts.

With the textPrep preprocessing pipeline, it is much easier to quickly iterate through preprocessing configurations, assess data quality, and produce better topics. To begin the process of finding a good preprocessing configuration, we recommend an iterative strategy that begins with a configuration similar to the lightweight configuration and stacks or removes one rule at a time until the data quality and vocabulary seems reasonable. The ability to filter by token frequency as in baseline 2 is built into the pipeline, as well as all of the rules that we used in these experiments. textPrep also allows for easy integration of new rules as new social media platforms with new types of text post content emerge.

7 CONCLUSIONS

In this paper, we present textPrep, a text preprocess- [redacted]

ACKNOWLEDGEMENTS

This work was supported by the National Science Foundation grant numbers #1934925 and #1934494, and by the Massive Data Institute (MDI) at Georgetown University. We would like to thank our funders.

REFERENCES

Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E. D., Gutierrez, J. B., and Kochut, K. (2017). A brief survey of text mining: Classification, clustering and extraction techniques. *arXiv preprint arXiv:1707.02919*.

Bird, S., Klein, E., and Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Blei, D. M. and Lafferty, J. D. (2006). Dynamic topic models. In *International Conference on Machine Learning (ICML)*.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., and Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122.

Bunk, S. and Krestel, R. (2018). Welda: Enhancing

[redacted]

cessed: 2021-03-01.

InternetLiveStats (2021). Twitter usage statistics. http://www.internetlivestats.com/twitter-statistics/. Accessed: 2021-03-01.

Knoblock, C. A., Lerman, K., Minton, S., and Muslea, I. (2003). Accurately and reliably extracting data from the web: A machine learning approach. In *Intelligent exploration of the web*, pages 275–287. Springer.

Lafferty, J. D. and Blei, D. M. (2006). Correlated topic models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 147–154.

Lau, J. H., Newman, D., and Baldwin, T. (2014). Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In *Conference of the European Chapter of the Association for Computational Linguistics*, pages 530–539.

Li, C., Wang, H., Zhang, Z., Sun, A., and Ma, Z. (2016). Topic modeling for short texts with auxiliary word

embeddings. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 165–174.

McCallum, A. K. (2002). Mallet: A machine learning for language toolkit.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Moody, C. E. (2016). Mixing dirichlet topic models and word embeddings to make lda2vec. *CoRR*, abs/1605.02019.

Nigam, K., McCallum, A. K., Thrun, S., and Mitchell, T. (2000). Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2-3):103–134.

Noyes, D. (2020). The top 20 valuable facebook statistics - updated may 2017. https://zephoria.com/top-15-valuable-facebook-statistics/. Accessed: 2021-03-01.

[redacted]

shop on New Challenges for NLP Frameworks, pages 45–50.

Schofield, A., Magnusson, M., and Mimno, D. (2017). Pulling out the stops: Rethinking stopword removal for topic models. In *Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 432–436.

Singh, L., Bansal, S., Bode, L., Budak, C., Chi, G., Kawintiranon, K., Padden, C., Vanarsdall, R., Vraga, E., and Wang, Y. (2020). A first look at covid-19 information and misinformation sharing on twitter.

Srividhya, V. and Anitha, R. (2010). *evaluating preprocessing techniques in text categorization. *International Journal of Computer Science and Application*, 47(11):49–51.

Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M. (2006). Hierarchical dirichlet processes. *Journal of* *the American Statistical Association*, 101(476):1566–1581.

Thompson, L. and Mimno, D. (2018). Authorless topic models: Biasing models away from known structure. In *COLING*.

Vassiliadis, P., Vagena, Z., Skiadopoulos, S., Karayannidis, N., and Sellis, T. (2000). Arktos: A tool for data cleaning and transformation in data warehouse environments. *IEEE Data Engineering Bulletin*, 23(4):42–47.

Wang, X. and McCallum, A. (2006). Topics over time: A non-markov continuous-time model of topical trends. In *ACM International Conference on Knowledge Discovery and Data Mining (KDD)*.

Yin, J. and Wang, J. (2014). A dirichlet multinomial mixture model-based approach for short text clustering. In *ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 233–242.

Zhao, W. X., Jiang, J., Weng, J., He, J., Lim, E.-P., Yan, H., and Li, X. (2011). Comparing twitter and traditional