Guidelines in Selecting Appropriate Text Preprocessing Methods

Christine P. Chai

Microsoft
chrchai@microsoft.com

Conference on Statistical Practice (CSP) 2021

Slides on GitHub: `https://tinyurl.com/csp-2021-chai`

Why is **data** preprocessing important?

- Data scientists spend more than 50% of their work time on data preprocessing, i.e., preparing the data for analysis.

  – *The New York Times* (Lohr, 2014)

- Although time-consuming, data preprocessing is worth the efforts. Real-life data are messier than people think! (Chai, 2020) Adequately preprocessed data are key to success in modeling.

- Data can be structured (numeric) and/or unstructured (text).
- There is no one-size-fits-all solution for data preprocessing!

Why is **text** preprocessing important?

- “Merrill Lynch recently estimated that more than 80% of all potentially useful business information is unstructured data.” (Gharehchopogh and Khalifelu, 2011)

- Text data also require appropriate preprocessing, to prepare the corpus for machine learning and text mining models. (Kalra and Aggarwal, 2018)

- Webscraped text often contains lots of HTML formats.
  $\Rightarrow$ Remove using Python `BeautifulSoup` or R `textclean`.
- Application-specific formats: e.g. [10:23 PM] in chat messages
  $\Rightarrow$ Remove these formats using regular expressions.

Text preprocessing is more important than it seems!

- Text preprocessing details affect **reproducibility.** (Roy et al., 2018)
- Removing stopwords: Which words are regarded as stopwords? Obvious words: “I”, “the”, etc. What about non-obvious words? e.g. Is “algorithm” a stopword in a computer science corpus?

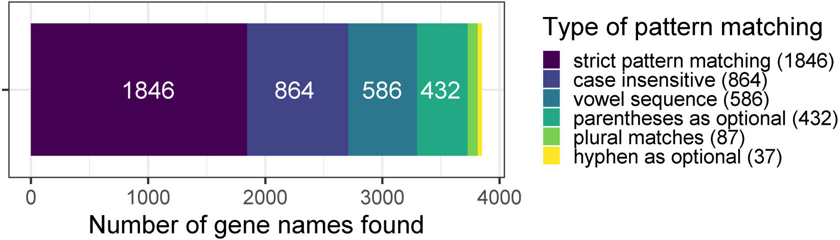
- Trieschnigg et al. (2007): **Tokenization decisions** can contribute more to system performance than the text model itself.

- Text string `NF-`κ` B/CD28-responsive` can be tokenized in many ways, resulting in different precision for document retrieval:
  - `nf `κ` b cd responsive`: 32% precision
  - `NF-`κ` B/CD28-responsive` (non-whitespace): 17%
  - `nf kappa nfkappa b cd 28 respons bcd28respons`: 40%

Tokenization Choices Matter: Another Example

- Tokenization choices of the gene name *alpha-2-macroglobulin* lead to different results of pattern matching in biomedical text.

Official gene name: alpha-2-macroglobulin



Data from Table 2 in (Cohen et al., 2002)

Text Preprocessing: Resources $\neq$ Guidelines

- Lots of resources:
  - NLTK in Python (Natural Language Toolkit) (Bird et al., 2009)
  - R packages: `stringr`, `quanteda`, `textclean`, `tm`, etc.
  - SAS Text Miner, Microsoft Azure Machine Learning Studio
- But most are for implementation (**how**), not for guidelines (**why**).
- Unanswered questions include and are not limited to:
  - Which methods are better for which text applications?
  - How to improve the current text preprocessing pipeline?
  - What are the potential risks of a specific method?

Structure of this talk: Text Preprocessing Methods

- Discuss several commonly-used methods
  Explain the advantages and potential issues
- Review specific examples of text analysis
  How preprocessing decisions contribute to the modeling
- Discussion and key takeaways

- But before we get into the methods, let's talk about the recent advances in natural language processing.

Recent Advances in Natural Language Processing

- Lots of pretrained word embeddings for text models: `word2vec` (Mikolov et al., 2013), `GloVe` (Pennington et al., 2014), `BERT` (Devlin et al., 2018), `ELMo` (Peters et al., 2018), etc.
- Word embeddings map words to vectors using a large corpus.
- Helpful for text models and support in multiple languages.

- We still need to learn text preprocessing methods, because **word embeddings are not a panacea**, i.e., have some limitations.
  - Tokenization is required to break a text string into tokens (words).
  - Some minority languages (e.g. Tibetan) are not supported yet.[1]
  - To add extra words (e.g. company-specific acronyms), we may need to retrain the word embeddings. (Wilson et al., 2020)

[1] `https://github.com/google-research/bert/blob/master/multilingual.md`

Tokenization: Implementation

- Convert a text string into a sequence of words (tokens)
- Whitespace is not the only separator. (Clough, 2001)
- Tools: Python `nltk.tokenize`, R `str_split` in `stringr`

- Example 1: “Statistics is fun” ⇒ “Statistics”, “is”, “fun”
- Example 2: “I downloaded this software on-the-fly”
  Does “on-the-fly” count as one word or three words?

- We need to **decide which characters count as separators.**
- Examine abbreviations and non-standard punctuation usage, especially in biomedical text. (Díaz and López, 2015)
- And most importantly, be clear and consistent with the definitions.

Tokenization: Need Precise Definitions

- **Lexical richness** measures the quality of vocabulary in a corpus (Malvern and Richards, 2012). It can be defined by

$$\textbf{type-token ratio} = \frac{\text{the number of types (distinct words)}}{\text{the number of tokens (total words)}}$$

- But the terms “type” and “token” are loosely defined.
  e.g. “don’t” vs “do not”, “New_York” vs “New York”

- Some researchers try various definitions and select one that produces a statistically significant result. (Cohen et al., 2019)
- **P-hacking hurts reproducibility in research!** (Head et al., 2015)
- Good practice: Give the precise definition of “type” and “token”

Tokenization: Compound Words

- Compound words in languages with whitespace between tokens
  - “White House” in English, “sin embargo” (however) in Spanish, “parce que” (because) in French (Barrett and Weber-Jahnke, 2011)
  - In Arabic, a word has up to four independent tokens. (Attia, 2007)

- Traditional approach: Find the terms that are statistically and practically significant (Blei and Lafferty, 2009)
- Modern approach: Leverage contextualized word representations to find the compound words (Shwartz and Dagan, 2019)

- CJK (Chinese, Japanese, Korean) do not have whitespaces, and a word can also contain multiple tokens.
- Possible to train a neural network model on known words, and update the model based on new information. (Hiraoka et al., 2019)

Handling Punctuation

- Easiest way is to remove all punctuation from the text corpus.
  - Acceptable for information retrieval & topic modeling, where we focus on text rather than sentence (Korde and Mahender, 2012).
  - Inappropriate for applications that require sentence segmentation!

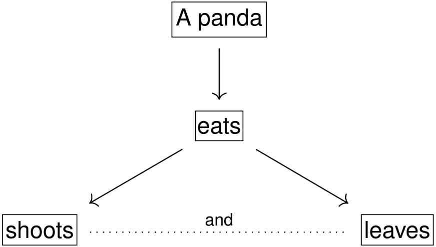
- **Punctuation provides syntactic information for parsing.**
- Categories: sentence-final, sentence-internal, word-internal
- Part-of-speech tagging needs to identify sentence boundaries first.
- Sentences are necessary for end-use applications, such as: **text summarization, machine translation, question answering.** (Patil et al., 2015; Kim and Zhu, 2019; Li and Croft, 2001)

Punctuation Serves as Discourse Markers

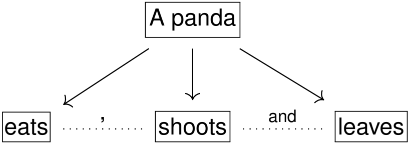
Discourse parsing needs punctuation to identify relations between text.

(Ji and Eisenstein, 2014)

- “A panda eats shoots and leaves.” (without comma)



- “A panda eats, shoots and leaves.” (with comma)



– *Eats, Shoots & Leaves: The zero tolerance approach to punctuation* by Truss (2004)

Removing Punctuation

- If we decide to remove punctuation from the corpus, using regular expressions is straightforward.
- Python and R both provide the list of punctuation symbols:

  `!"#$\%&'()*+,-./:;<=>?@[\]^_`{|}~`

- Beware of word-internal punctuation for contractions
- Predefined list to split contractions[2] (e.g. "don't" ⇒ "do not")
- Python `pycontractions` provides higher precision with context e.g. Does "I'd" map to "I would" or "I had"? (Beaver, 2019)

- Consider retaining emoticons (:-), :p) with Python NLTK's `TweetTokenizer()`, especially in social media data

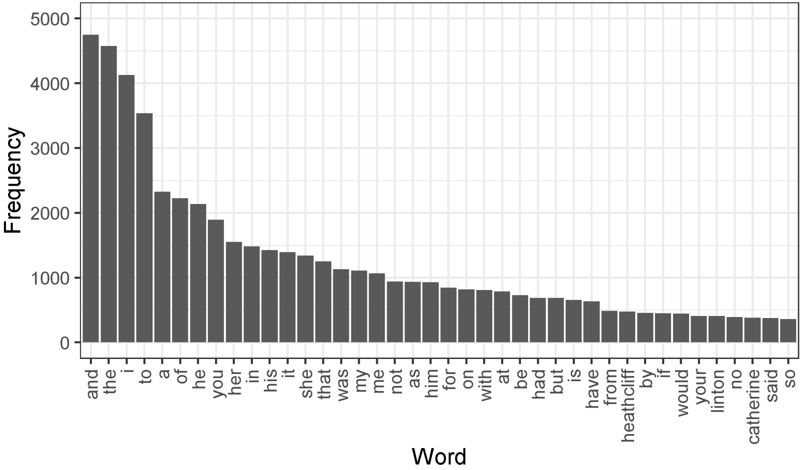
[2] https://gist.github.com/nealrs/96342d8231b75cf4bb82

Removing Stopwords: Overview

- Stopwords are the words that **do not distinguish one document from another in the corpus.** (Ferilli et al., 2014)
  - General: prepositions (“at”), pronouns (“you”), articles (“the”), etc.
  - Domain-specific: “method”, “problem”, “algorithm” in a corpus of machine learning papers (Fan et al., 2017)
- **Don’t just regard stopwords as “noise”.**
  Every word has semantic meaning, no matter how little.
- Stopwords are identified from a predefined list, or from corpus:
  - TF-IDF (term frequency – inverse document frequency)
  - Entropy (information theory) (Gerlach et al., 2019)
  - Kullback-Leibler (KL) divergence (Sarica and Luo, 2020)
- Especially useful for text corpora with technical content

English Classic Novel: *Wuthering Heights*

- Most frequent words are stopwords: and, the, I, etc.
- Zipf's law: The frequency distribution is right-skewed. (Zipf, 1949)



Existing Stopword Lists

- Python NLTK (127 English words, supports 23 languages)
- R package `stopwords` (175 English words, 231 German words)
- List of $\sim$ 500 removes 40-50% of corpus. (Schofield et al., 2017)

- Project Gutenberg: 60,000+ publicly available e-books[3]
- Choose an e-book of general content (e.g. *Alice's Adventures in Wonderland*) to filter out non-technical words from a technical database. (Brooke et al., 2015)

- **Recommendation: Start with a smaller stopword list, then gradually add more stopwords if necessary.**
- Words removed do not return to the text model later.

[3] `https://www.gutenberg.org/`

Removing Stopwords: Known Issues

- Beware: Unintentional removal of important information
  e.g. “The The” (a band), “to be or not to be” from *Hamlet*
  e.g. “Chemistry is not easy.” ⇒ “Chemistry easy.” (Really?)
- Most stopword lists contain negation (“no”, “not”, “isn’t”, etc.).
  At a minimum, remove these terms to **preserve negation**!

- Stopwords have syntactic information for **dependency parsing.**[4]
- Patterns of stopword usage help in **authorship attribution** (Arun et al., 2009) and **plagiarism detection** (Stamatatos, 2011).
- Feature: How often pronouns are used for a specific name [5]
- Feature: A person ⇒ “he/she” or “he (she)” or else?

[4] (Elming et al., 2013; Poria et al., 2014)

[5] (Sánchez-Vega et al., 2019)

Stemming and Lemmatization: Overview

- Both stemming and lemmatization normalize words to their base forms for vocabulary consolidation. (Manning et al., 2008)
- A word may have inflectional or derivational morphemes.
  - Inflectional: enjoy (verb) → enjoy**s**, enjoy**ed** (verb)
  - Derivational: enjoy (verb) → enjoy**able** (adjective)
  - Derivations change the word's part-of-speech; inflections do not.[6]
- Vocabulary consolidation decreases noise in the corpus e.g. worry, worries, worried, worrying → worri
- Improves performance for **text classification** (Biba and Gjati, 2014) and **information retrieval** (Rajput and Khare, 2015)
- But risks losing **semantic information in word variations** Some sentiments and named entities may be destroyed (Bao et al., 2014; Cambria et al., 2017)

[6] https://blogs.umass.edu/anyman/files/2020/04/Inflectional-vs.-Derivational-Morphemes.pdf

Stemming and Lemmatization: Comparison

- Stemming uses suffix rules to return each word to its root (stem), and removes both inflectional and derivational variations.
- Stemming algorithms: Porter (1980), Lovins (1968), Lancaster[7]
- Fast and requires less context, but creates tokens (not real words) e.g. challenge → **challeng**, president → **presid**

- Lemmatization maps a word's inflected forms to its lemma, and considers the part-of-speech. (Bergmanis and Goldwater, 2018)
- Tool: Python NLTK `WordNetLemmatizer`
- Isolated word: return the most common lemma or multiple options
- Helpful in morphologically rich languages (e.g. Turkish, Finnish), where word changes indicate subject / verb / object / etc. (Tsarfaty et al., 2010)

[7] (Paice and Hooper, 2005)

Stemming and Lemmatization: Potential Risks

- **Over-consolidation**: Words of different meanings → same token
  "computer", "computation" → "comput" (R package `SnowballC`)
- Some words have multiple lemmas, and context is needed.
  "leaves" → "leave" (verb, or noun as days off from a job)
  "leaves" → "leaf" (noun, as on the tree)

- **Sentiment analysis**: Positive & negative words grouped together[8]
  "objective" (positive), "objection" (negative) → "object"
- **Named entity recognition**: Entities may become unrecognizable
  "Guns N' Roses" → "gun n rose" (Cambria et al., 2017)

[8] (Ghazvinian, 2011)

N-Gramming: Retain Word Order

- N-gramming creates n-grams as phrases of *n* consecutive words, with a moving window of length *n* across the text.[9]
- Some n-grams are **multi-word expressions**, and should be retained as a single token. e.g. New Jersey, Black Lives Matter

- Word order is important in languages with few inflections. English: “dog bites man” vs “man bites dog” (Haspelmath, 1996)
- Languages with lots of inflections ⇒ word order rarely matters Latin: Noun as subject has a different form than noun as object. But word order info is still helpful for context. (Beier et al., 2011)

[9] (Gries and Mukherjee, 2010)

Detecting Multi-Word Expressions: Methods

- **Practical significance**: Threshold of raw frequency
  Microsoft Azure ML Studio sets 5 times as the default threshold.[10]
  Threshold of 100 times for a corpus of ∼1 billion words[11]
- **Statistical significance**: Conditional probability (Chai, 2017)
  Bi-gram $w_1 w_2$ is a multi-word expression when $P(w_2|w_1) > P(w_2)$
- **Conditional random fields** (Lafferty et al., 2001; Maldonado et al., 2017)
  Undirected probabilistic graphical model for pattern recognition
- **Retrain word embeddings** to identify multi-word expressions
  Still a growing field in deep learning. (Ashok et al., 2019)
  Tokenizing multi-word expressions does not negatively impact machine translation of single words. (Otani et al., 2020)

[10] `https://tinyurl.com/microsoft-n-grams`

[11] (Berberich and Bedathur, 2013)

Detecting Multi-Word Expressions: Applications

- Bag-of-words models: Disregards grammar and word order
- **Topic modeling, text classification, information retrieval**
- Multi-word expressions produce stronger signals.[12]

- Natural language processing tasks where context is required
- **Word sense disambiguation**: Determine which meaning of the word is used in a given context (Finlayson and Kulkarni, 2011)
- **Machine translation**: Improves translation accuracy (Tan and Pal, 2014) and extends to multi-lingual corpora (Han et al., 2020)
- **Psycholinguistics** (linguistics + psychology): Idiom usage reflects mental representation (Müller, 2011)

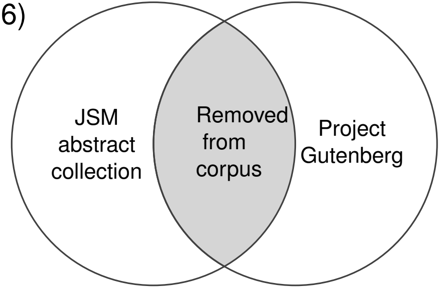
[12] (Blei and Lafferty, 2009)

Example 1: JSM Abstract Dataset – Preprocessing

- Dataset: 3000+ abstracts and 700+ sessions from JSM 2015
  JSM (Joint Statistical Meetings) is a large global conference.
- Lots of technical terms: e.g. maximum likelihood estimation
- **Extract technical terms for conference session scheduling**

Stopword removal process: (Bi, 2016)

1. Stem and n-gram the JSM dataset.
2. Stem and n-gram an e-book from Project Gutenberg.
3. Remove anything from 1. that exists in 2.



Example 1: JSM Abstract Dataset – Applications

- Topic models: Six topics found by LDA (latent Dirichlet allocation)
  Spatial stats, clinical trials, Bayesian, genomics, hypothesis tests
- Conference session scheduling:
  Minimize the overlap of sessions with similar topics
  People interested in a topic can join many relevant sessions
- Abstracts often include keywords from authors, but an automated system is still needed for session scheduling. (Sweeney, 2020)
- **Virtual conferences: Scheduling is still a challenge!** [13]
  Without physical restrictions, but need to account for other factors.
  Availability of on-demand recordings may change the equation.

[13] (Patro et al., 2020)

Example 2: Social Media Data – Preprocessing

- Twitter: Twitter developer API, Python `tweepy`, R `twitteR`
- Webscraping: Python `scrapy`, R `rvest`, webscraper.io, etc.

- Remove HTML tags: Python `BeautifulSoup`, R `textclean`
- Remove leading and trailing whitespaces for each message
- Python NLTK `TweetTokenizer()`[14] retains hashtags (`#hashtag`), mentions (`@username`), and emoticons (`:-D`)

- Text normalization: Consider splitting Internet slang abbreviations e.g. cu = see you, idk = I don't know (Pennell and Liu, 2011)
- Dependency parsing: Beware of nonstandard spellings and punctuation usage (Blodgett et al., 2018)

[14] https://www.nltk.org/api/nltk.tokenize.html

Example 2: Social Media Data – Applications

- **COVID-19**: Information spreading (Cinelli et al., 2020)
  Twitter dataset with daily updates (Banda et al., 2020)[15]
  COVID-Twitter-BERT: Pretrained large corpus (Müller et al., 2020)[16]
- **Analyze the impact of recent events**:
  Black Lives Matter (Giorgi et al., 2020)
  2020 US presidential election (Chen et al., 2020)
  National Disaster Situational Awareness (Karami et al., 2020)
- But applications cannot be done without **text preprocessing**.
  Data need to be preprocessed to be ready for text model input.

[15] https://github.com/thepanacealab/covid19_twitter

[16] https://github.com/digitalepidemiologylab/covid-twitter-bert

Example 3: Text with Numerical Ratings – Data

- Employee satisfaction survey: (Chai, 2019)
  Ratings from **1 (least satisfied)** to **10 (most satisfied)**
  Text comment to explain why you made the rating
- Best: Combined analysis in text and numerical data

- Need to **preserve negation terms** while removing stopwords.
  "No clear path to promotion" ⇒ "Clear path promotion"?
  Negation terms should be excluded from the stopword list.

- Data often contain possible errors.
  "Love my work – very varied." ⇒ rating 1?
  Common reason: **Confusion with rating scale** (Fisher, 2013)
  The respondent meant 10 (most satisfied) but reverted the scale.

Example 3: Text with Numerical Ratings – Modeling

- How to **detect potential rating errors** in the survey?
- **Method 1: Leverage the text response** (i.e., text mining)
  Predict the rating from the text comment with supervised methods
  Large discrepancy between predicted rating and actual rating
  $\Rightarrow$ Flag as potential error and examine the record
- **Method 2: Leverage other ratings** in the same dataset
  Predict the overall rating from the ratings of each item
  e.g. EM (expectation maximization) algorithm (Fisher and Lee, 2011)
  Also check for large discrepancies between predicted and actual

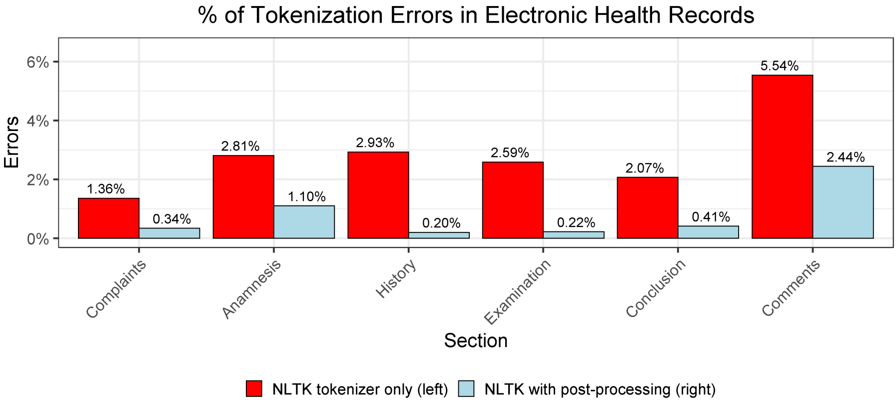
Example 4: Biomedical Data

- Pretrained word embeddings for biomedical text
  Gradually increasing in popularity (Wang et al., 2018)
  BioBERT (Lee et al., 2020): Pretrained on PubMed and PMC
- Challenge: Word vectors don't reflect internal structure of words.
  e.g. "**delta**proteobacteria" and "**beta**proteobacteria" are related.
  $\Rightarrow$ Leverage subword info. in pretraining (Zhang et al., 2019)

- Tokenizing biomedical text includes **Greek letter normalization** and **break point identification** (Jiang and Zhai, 2007)
- Both MIP-**1**-α and MIP-**1**α should tokenize to `MIP 1 alpha`
- Text string NF-κ B/CD28-responsive (Trieschnigg et al., 2007)
  - nf κ b cd responsive: 32% precision in document retrieval
  - `nf kappa nfkappa b cd 28 respons bcd28respons`: 40%

Example 4: Biomedical Data

- NLTK tokenizer + post-processing ⇒ fewer tokenization errors



Data from Table 1 in (Grön and Bertels, 2018)

Discussion

- Different applications require different preprocessing methods.
- Bag-of-words models: Acceptable to remove punctuation, remove stopwords, stem and lemmatize the corpus.
    - **Topic modeling, text classification, information retrieval**, etc.
- Other applications that require more context: Need to reconsider whether each text preprocessing step is appropriate.
    - **Dependency parsing, machine translation, text summarization, question answering**, etc.
- Detecting multi-word expressions is a challenging problem, even with the help of word embeddings. (Park and Tsvetkov, 2019)

Key Takeaways

- Text preprocessing prepares the corpus for analysis, so this directly affects data quality in modeling.
- Document the decisions made in the process (Nugent, 2020)
- **Exclude negation terms from the stopword list!** Don't accidentally remove negation from the corpus.
- Word embeddings are useful, but not a panacea solution. A non-trivial prerequisite: Tokenize the corpus into words
- Need to learn text preprocessing basics to make better decisions.

Acknowledgments

- I am grateful for the support of **Microsoft**.
- I'd like to thank **Prof. David Banks**, my PhD advisor at Duke University, who motivated me to do text preprocessing research.
- I also thank my friend **Sourav Sen** for feedback on the slides.
- I declare that there is no conflict of interest.

References I

Arun, R., Suresh, V., and Madhavan, C. V. (2009). Stopword graphs and authorship attribution in text corpora. In *2009 IEEE International Conference on Semantic Computing*, pages 192–196. IEEE.

Ashok, A., Elmasri, R., and Natarajan, G. (2019). Comparing different word embeddings for multiword expression identification. In *International Conference on Applications of Natural Language to Information Systems*, pages 295–302. Springer.

Attia, M. (2007). Arabic tokenization system. In *Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources*, pages 65–72.

Banda, J. M., Tekumalla, R., Wang, G., Yu, J., Liu, T., Ding, Y., and Chowell, G. (2020). A large-scale COVID-19 Twitter chatter dataset for open scientific research – An international collaboration. *arXiv preprint arXiv:2004.03688*.

Bao, Y., Quan, C., Wang, L., and Ren, F. (2014). The role of pre-processing in Twitter sentiment analysis. In *International Conference on Intelligent Computing*, pages 615–624. Springer.

References II

Barrett, N. and Weber-Jahnke, J. (2011). Building a biomedical tokenizer using the token lattice design pattern and the adapted Viterbi algorithm. *BMC Bioinformatics*, 12(3):S1.

Beaver, I. (2019). pycontractions 2.0.1. Python library. Available from: `https://pypi.org/project/pycontractions/`.

Beier, C., Hansen, C., Lai, I.-w., and Michael, L. (2011). Exploiting word order to express an inflectional category: Reality status in iquito. *Linguistic Typology*, 15(1):65–99.

Berberich, K. and Bedathur, S. (2013). Computing n-gram statistics in MapReduce. In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 101–112.

Bergmanis, T. and Goldwater, S. (2018). Context sensitive neural lemmatization with Lematus. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 1391–1400.

Bi, Y. (2016). Scheduling optimization with LDA and greedy algorithm. Master's thesis, Duke University. LDA stands for latent Dirichlet allocation.

References III

Biba, M. and Gjati, E. (2014). Boosting text classification through stemming of composite words. In *Recent Advances in Intelligent Informatics*, pages 185–194. Springer.

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. (2009). Visualizing topics with multi-word expressions. *arXiv preprint arXiv:0907.1013*.

Blodgett, S. L., Wei, J., and O'Connor, B. (2018). Twitter universal dependency parsing for African-American and mainstream American English. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425.

Brooke, J., Hammond, A., and Hirst, G. (2015). GutenTag: An NLP-driven tool for digital humanities research in the Project Gutenberg corpus. In *Proceedings of the Fourth Workshop on Computational Linguistics for Literature*, pages 42–47.

Cambria, E., Poria, S., Gelbukh, A., and Thelwall, M. (2017). Sentiment analysis is a big suitcase. *IEEE Intelligent Systems*, 32(6):74–80.

References IV

Chai, C. P. (2017). *Statistical Issues in Quantifying Text Mining Performance*. PhD dissertation, Duke University.

Chai, C. P. (2019). Text mining in survey data. *Survey Practice*, 12(1):1–13.

Chai, C. P. (2020). The importance of data cleaning: Three visualization examples. *CHANCE*, 33(1):4–9.

Chen, E., Deb, A., and Ferrara, E. (2020). #Election2020: The first public Twitter dataset on the 2020 US presidential election. *arXiv preprint arXiv:2010.00600*.

Cinelli, M., Quattrociocchi, W., Galeazzi, A., Valensise, C. M., Brugnoli, E., Schmidt, A. L., Zola, P., Zollo, F., and Scala, A. (2020). The COVID-19 social media infodemic. *Scientific Reports*, 10(1):1–10.

Clough, P. (2001). A Perl program for sentence splitting using rules. *University of Sheffield*.

Cohen, K. B., Acquaah-Mensah, G. K., Dolbey, A. E., and Hunter, L. (2002). Contrast and variability in gene names. In *Proceedings of the ACL-02 Workshop on Natural Language Processing in the Biomedical Domain*, volume 3, pages 14–20. Association for Computational Linguistics (ACL).

References V

Cohen, K. B., Hunter, L. E., and Pressman, P. S. (2019). P-hacking lexical richness through definitions of “type” and “token”. *Studies in Health Technology and Informatics*, 264:1433–1434.

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

Díaz, N. P. C. and López, M. J. M. (2015). An analysis of biomedical tokenization: Problems and strategies. In *Proceedings of the Sixth International Workshop on Health Text Mining and Information Analysis*, pages 40–49.

Elming, J., Johannsen, A., Klerke, S., Lapponi, E., Alonso, H. M., and Søgaard, A. (2013). Down-stream effects of tree-to-dependency conversions. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 617–626.

Fan, A., Doshi-Velez, F., and Miratrix, L. (2017). Promoting domain-specific terms in topic models with informative priors. *arXiv preprint arXiv:1701.03227*.

References VI

Ferilli, S., Esposito, F., and Grieco, D. (2014). Automatic learning of linguistic resources for stopword removal and stemming from text. *Procedia Computer Science*, 38:116–123.

Finlayson, M. and Kulkarni, N. (2011). Detecting multi-word expressions improves word sense disambiguation. In *Proceedings of the Workshop on Multiword Expressions: From Parsing and Generation to the Real World*, pages 20–24.

Fisher, N. (2013). *Analytics for leaders: A performance measurement system for business success*. Cambridge University Press.

Fisher, N. and Lee, A. (2011). Getting the 'correct' answer from survey responses: A simple application of the EM algorithm. *Australian & New Zealand Journal of Statistics*, 53(3):353–364.

Gerlach, M., Shi, H., and Amaral, L. A. N. (2019). A universal information theoretic approach to the identification of stopwords. *Nature Machine Intelligence*, 1(12):606–612.

References VII

Gharehchopogh, F. S. and Khalifelu, Z. A. (2011). Analysis and evaluation of unstructured data: Text mining versus natural language processing. In *2011 5th International Conference on Application of Information and Communication Technologies (AICT)*, pages 1–4. IEEE.

Ghazvinian, A. (2011). Star quality: Sentiment categorization of restaurant reviews. Technical report, Stanford University.

Giorgi, S., Guntuku, S. C., Rahman, M., Himelein-Wachowiak, M., Kwarteng, A., and Curtis, B. (2020). Twitter corpus of the #blacklivesmatter movement and counter protests: 2013 to 2020. *arXiv preprint arXiv:2009.00596*.

Gries, S. T. and Mukherjee, J. (2010). Lexical gravity across varieties of English: An ICE-based study of n-grams in Asian Englishes. *International Journal of Corpus Linguistics*, 15(4):520–548.

Grön, L. and Bertels, A. (2018). Clinical sublanguages: Vocabulary structure and its impact on term weighting. *Terminology: International Journal of Theoretical and Applied Issues in Specialized Communication*, 24(1):41–65.

Han, L., Jones, G. J., and Smeaton, A. F. (2020). MultiMWE: Building a multi-lingual multi-word expression (MWE) parallel corpora. *arXiv preprint arXiv:2005.10583*.

References VIII

Haspelmath, M. (1996). Word-class-changing inflection and morphological theory. In *Yearbook of Morphology 1995*, pages 43–66. Springer.

Head, M. L., Holman, L., Lanfear, R., Kahn, A. T., and Jennions, M. D. (2015). The extent and consequences of p-hacking in science. *PLOS Biology*, 13(3):e1002106.

Hiraoka, T., Shindo, H., and Matsumoto, Y. (2019). Stochastic tokenization with a language model for neural text classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1620–1629.

Ji, Y. and Eisenstein, J. (2014). Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 13–24.

Jiang, J. and Zhai, C. (2007). An empirical study of tokenization strategies for biomedical information retrieval. *Information Retrieval*, 10(4-5):341–363.

Kalra, V. and Aggarwal, R. (2018). Importance of text data preprocessing & implementation in RapidMiner. In *Proceedings of the First International Conference on Information Technology and Knowledge Management (ICITKM)*, volume 14, pages 71–75.

References IX

Karami, A., Shah, V., Vaezi, R., and Bansal, A. (2020). Twitter speaks: A case of national disaster situational awareness. *Journal of Information Science*, 46(3):313–324.

Kim, K. H. and Zhu, Y. (2019). *Researching Translation in the Age of Technology and Global Conflict: Selected Works of Mona Baker*. Routledge.

Korde, V. and Mahender, C. N. (2012). Text classification and classifiers: A survey. *International Journal of Artificial Intelligence & Applications*, 3(2):85.

Lafferty, J. D., McCallum, A., and Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289.

Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., and Kang, J. (2020). BioBERT: A pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Li, X. and Croft, W. B. (2001). Incorporating syntactic information in question answering. Technical report, Center for Intelligent Information Retrieval at University of Massachusetts – Amherst.

References X

Lohr, S. (2014). For big-data scientists, 'janitor work' is key hurdle to insights. *The New York Times*. Available from: `https://www.nytimes.com/2014/08/18/technology/for-big-data-scientists-hurdle-to-insights-is-janitor-work.html`.

Lovins, J. B. (1968). Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11(1-2):22–31.

Maldonado, A., Han, L., Moreau, E., Alsulaimani, A., Chowdhury, K. D., Vogel, C., and Liu, Q. (2017). Detection of verbal multi-word expressions via conditional random fields with syntactic dependency features and semantic re-ranking. In *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, pages 114–120.

Malvern, D. and Richards, B. (2012). Measures of lexical richness. *The encyclopedia of applied linguistics*.

Manning, C., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

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

References XI

Müller, M., Salathé, M., and Kummervold, P. E. (2020). COVID-Twitter-BERT: A natural language processing model to analyse COVID-19 content on Twitter. *arXiv preprint arXiv:2005.07503*.

Müller, P. O. (2011). Multi-word expressions. ingeborg ohnheiser: Word formation, an international handbook of the languages of europe [hsk series].

Nugent, R. (2020). Instead of just teaching data science, let's understand how and why people do it. Symposium on Data Science and Statistics. Abstract available from: `https://ww2.amstat.org/meetings/sdss/2020/onlineprogram/AbstractDetails.cfm?AbstractID=308230`.

Otani, N., Ozaki, S., Zhao, X., Li, Y., St Johns, M., and Levin, L. (2020). Pre-tokenization of multi-word expressions in cross-lingual word embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4451–4464.

Paice, C. and Hooper, R. (2005). Lancaster stemmer.

References XII

Park, C. Y. and Tsvetkov, Y. (2019). Learning to generate word-and phrase-embeddings for efficient phrase-based neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 241–248.

Patil, A., Pharande, K., Nale, D., and Agrawal, R. (2015). Automatic text summarization. *International Journal of Computer Applications*, 109(17).

Patro, G. K., Chakraborty, A., Ganguly, N., and Gummadi, K. P. (2020). On fair virtual conference scheduling: Achieving equitable participant and speaker satisfaction. *arXiv preprint arXiv:2010.14624*.

Pennell, D. and Liu, Y. (2011). Toward text message normalization: Modeling abbreviation generation. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5364–5367. IEEE.

Pennington, J., Socher, R., and Manning, C. D. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

References XIII

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Poria, S., Agarwal, B., Gelbukh, A., Hussain, A., and Howard, N. (2014). Dependency-based semantic parsing for concept-level text analysis. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 113–127. Springer.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3):130–137.

Rajput, B. S. and Khare, N. (2015). A survey of stemming algorithms for information retrieval. *International Organization of Scientific Research – Journal of Computer Engineering*, 17(3):76–80.

Roy, D., Mitra, M., and Ganguly, D. (2018). To clean or not to clean: Document preprocessing and reproducibility. *Journal of Data and Information Quality (JDIQ)*, 10(4):1–25.

Sánchez-Vega, F., Villatoro-Tello, E., Montes-y Gómez, M., Rosso, P., Stamatatos, E., and Villaseñor-Pineda, L. (2019). Paraphrase plagiarism identification with character-level features. *Pattern Analysis and Applications*, 22(2):669–681.

References XIV

Sarica, S. and Luo, J. (2020). Stopwords in technical language processing. *arXiv preprint arXiv:2006.02633*.

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

Shwartz, V. and Dagan, I. (2019). Still a pain in the neck: Evaluating text representations on lexical composition. *Transactions of the Association for Computational Linguistics*, 7:403–419.

Stamatatos, E. (2011). Plagiarism detection using stopword n-grams. *Journal of the American Society for Information Science and Technology*, 62(12):2512–2527.

Sweeney, K. (2020). Unsupervised machine learning for conference scheduling: A natural language processing approach based on latent Dirichlet allocation. Master's thesis, NHH Norwegian School of Economics.

Tan, L. and Pal, S. (2014). Manawi: Using multi-word expressions and named entities to improve machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 201–206.

References XV

Trieschnigg, D., Kraaij, W., and de Jong, F. (2007). The influence of basic tokenization on biomedical document retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 803–804.

Truss, L. (2004). *Eats, Shoots & Leaves: The zero tolerance approach to punctuation*. Penguin.

Tsarfaty, R., Seddah, D., Goldberg, Y., Kübler, S., Candito, M., Foster, J., Versley, Y., Rehbein, I., and Tounsi, L. (2010). Statistical parsing of morphologically rich languages (SPMRL): What, how and whither. In *Proceedings of the NAACL-HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 1–12. Association for Computational Linguistics. NAACL-HLT stands for the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.

Wang, Y., Liu, S., Afzal, N., Rastegar-Mojarad, M., Wang, L., Shen, F., Kingsbury, P., and Liu, H. (2018). A comparison of word embeddings for the biomedical natural language processing. *Journal of Biomedical Informatics*, 87:12–20.

References XVI

Wilson, S., Magdy, W., McGillivray, B., Garimella, K., and Tyson, G. (2020). Urban dictionary embeddings for slang NLP applications. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 4764–4773.

Zhang, Y., Chen, Q., Yang, Z., Lin, H., and Lu, Z. (2019). BioWordVec, improving biomedical word embeddings with subword information and MeSH. *Scientific data*, 6(1):1–9.

Zipf, G. K. (1949). Human behavior and the principle of least effort.