Natural Language Engineering (2023), **29**, pp. 509–553
doi:10.1017/S1351324922000213

CAMBRIDGE UNIVERSITY PRESS

SURVEY PAPER

Comparison of text preprocessing methods

Christine P. Chai

Microsoft Corporation, One Microsoft Way, Redmond, WA 98052, USA
E-mail: chrchai@microsoft.com

(Received 11 October 2019; revised 13 April 2022; accepted 18 April 2022; first published online 13 June 2022)

Abstract

Text preprocessing is not only an essential step to prepare the corpus for modeling but also a key area that directly affects the natural language processing (NLP) application results. For instance, precise tokenization increases the accuracy of part-of-speech (POS) tagging, and retaining multiword expressions improves reasoning and machine translation. The text corpus needs to be appropriately preprocessed before it is ready to serve as the input to computer models. The preprocessing requirements depend on both the nature of the corpus and the NLP application itself, that is, what researchers would like to achieve from analyzing the data. Conventional text preprocessing practices generally suffice, but there exist situations where the text preprocessing needs to be customized for better analysis results. Hence, we discuss the pros and cons of several common text preprocessing methods: removing formatting, tokenization, text normalization, handling punctuation, removing stopwords, stemming and lemmatization, n-gramming, and identifying multiword expressions. Then, we provide examples of text datasets which require special preprocessing and how previous researchers handled the challenge. We expect this article to be a starting guideline on how to select and fine-tune text preprocessing methods.

Keywords: Data preprocessing; Parsing; Text data mining

1. Introduction

Text mining can provide valuable insights, but the text data need to be adequately preprocessed first, just like numerical data (Kalra and Aggarwal 2018). Real-world data are dirty (Hernández and Stolfo 1998), so data scientists spend more than half of the time preprocessing and organizing the data (Gabernet and Limburn 2017; CrowdFlower 2017). For example, Twitter data contain HTML tags and user mentions, so researchers have to remove the formatting from the data before feeding the corpus into any text model (Angiani *et al.* 2016). Many text analysis models deal with words (Aggarwal and Zhai 2012; Kutuzov *et al.* 2017); hence, breaking down the text into words (i.e., tokenization) is also a necessary preprocessing step (Karthikeyan *et al.* 2020). Text preprocessing refers to these operations that prepare the corpus for analysis (Anandarajan, Hill, and Nolan 2019). Text preprocessing methods are not just essential to natural language processing (NLP), but they have actual implications to the modeling results (Samad, Khounviengxay, and Witherow 2020), just like raw data with errors can distort the regression output (Chai 2020) .

Text preprocessing also has a quantitative impact on the natural language applications. Forst and Kaplan (2006) showed that precise tokenization increased the coverage of grammars in German from 68.3 percent to 73.4 percent. Gomes, Adán-Coello, and Kintschner (2018) also showed that text preprocessing can boost the accuracy by more than 20 percent in sentiment analysis of social media data. Zhou *et al.* (2019) improved hypoglycemia detection by filtering stopwords and signaling medications in clinical notes in the US, which increased the F1 score by between 5.3 percent and 7.4 percent. According to Camacho-Collados and Pilehvar (2018),

there is a high variance in model performance (± 2.4 percent on average) depending on the text preprocessing method, especially with smaller sizes of training data. Trieschnigg, Kraaij, and de Jong (2007) discovered that different tokenization choices can result in a variability of more than 40 percent in the precision of biomedical document retrieval. The variance is large enough for researchers to choose a different model for better performance, while the real issue is choosing the appropriate methods in text preprocessing. Cohen, Hunter, and Pressman (2019) found out that in clinical text mining, tokenization choices can make the difference between getting publishable results or not, which indirectly contribute to the problem of false discoveries (Leek and Jager 2017). These examples provide the evidence that text preprocessing plays a much more important role than most people have realized (Hickman *et al*. 2020). Researchers need to make decisions in working with a dataset, and Nugent (2020) pointed out that human subjective decisions are as important as the machine learning algorithm itself.

Nevertheless, text preprocessing is more complex and difficult than it seems. Text contains many kinds of lexical information as described in the book *The Lexicon* (Ježek 2016), such as concept, grammar, syntax, and morphology. For example, grammar may not be important in topic modeling or text classification, but grammar is essential to end-user applications like question-answering or summarization (Torres-Moreno 2014). Different types of text corpora require different preprocessing methods, so text preprocessing is not a one-size-fits-all process (Yuhang, Yue, and Wei 2010; Denny and Spirling 2018). Recent advances of pretrained language models like Bidirectional Encoder Representations from Transformers (BERT) (Devlin *et al*. 2018) has brought NLP to an unprecedented level (Wang, Gu, and Tang 2020), but preprocessing the text corpus is still necessary to get the data ready for the input (Kaviani and Rahmani 2020; Armengol Estapé 2021). Example preprocessing operations include text normalization and unpacking hashtags (Polignano *et al*. 2019). There are still many decisions to be made, because the number of possible models grows exponentially with the abundance of hyperparameters in neural networks. With eight binary hyperparameters, the number of possible models is as high as $2^8 = 256$ (Dodge *et al*. 2019). For instance, do we choose uniform or term frequency-inverse document frequency (TF-IDF) weights? Do we retain multiword expressions? If yes, what is the cutoff frequency? It is practically infeasible to try every single combination to find the best-performing model, so researchers should narrow down the search space, that is, find which preprocessing choices are more appropriate for the target application.

Therefore, we would like to discuss various text preprocessing methods by summarizing the commonly used practices and pointing out their limitations. These methods include removing formatting, tokenization, text normalization, handling punctuation, removing stopwords, stemming and lemmatization, n-gramming, and identifying multiword expressions. Figure 1 shows a common order of application of the text preprocessing modules, but in some cases, punctuation is handled (or even removed) during the tokenization stage (Welbers, Van Atteveldt, and Benoit 2017; Mullen *et al*. 2018). Researchers have performed text normalization and punctuation handling in either order (Bollmann 2019; Zupon, Crew, and Ritchie 2021), so we list the two modules in parallel in the diagram.

Our objective is to serve a variety of NLP applications and provide researchers guidance on selecting preprocessing methods for their text corpus. Kathuria, Gupta, and Singla (2021) also created a description of common text preprocessing techniques, but their main goal is to compare various open-source text mining tools such as Weka, Rapid Miner, R, and Python Jupyter Notebook. Many survey papers in text preprocessing are focused on a specific NLP application, such as text classification (HaCohen-Kerner, Miller, and Yigal 2020) or sentiment analysis in Brazilian Portuguese (Cirqueira *et al*. 2018). Other relevant survey papers like Vijayarani, Ilamathi, and Nithya (2015) and Nayak *et al*. (2016) seem to discuss the detailed implementation of text preprocessing, rather than the potential impact on the NLP applications.

In this article, the text preprocessing methods described are primarily for English, but some methods also apply to other languages. For example, Kannan and Gurusamy (2014) showed that

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

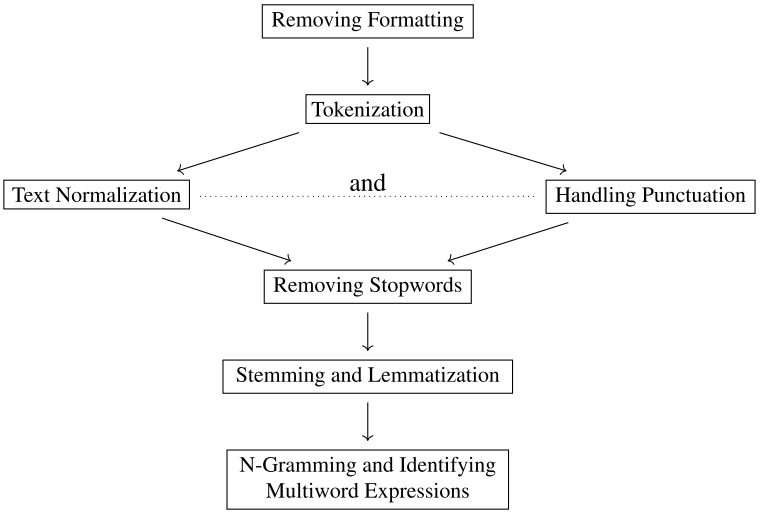

Figure 1. A common order of application of the text preprocessing modules.

English and French can be tokenized in similar ways due to the space-delimited nature in both languages. Here, we explicitly specify the language name to acknowledge the importance of data statement for NLP (Bender and Friedman 2018). There are more than 7000 languages in the world,[1] but English accounts for the vast majority of NLP research (Bender 2011; Mieke 2016). Munro (2015) pointed out that many researchers fail to name the language of their data, which is obviously English. We would like to be inclusive and do not assume that English is the default language studied in this field.

We also encourage researchers to use "text preprocessing" when referring to operations covered in this review article, unless this term is explicitly defined otherwise. Many researchers use "text cleaning" and "text preprocessing" interchangeably (Kunilovskaya and Plum 2021), but we adopt the latter to highlight the goal of getting the text corpus ready for input to NLP models (Aye 2011). Kadhim (2018) extended the term "text preprocessing" to the operations of converting text to numerical data (e.g., TF-IDF), and they clearly stated that the aim is to represent each document as a feature vector of individual words. We agree that this aspect is equally important, but "data representation" is a more appropriate term (Dařena 2019). Finally, we make a distinction between "text preprocessing" and "data preprocessing", where the latter refers to a broader range of data transformations, including scaling and feature selection on numerical/vectorized representations of text (García *et al.* 2016). One example is that Al Sharou, Li, and Specia (2021) uses "data preprocessing" to indicate the handling of all nonstandard appearance of language units (e.g., casing, hashtags, code-switching, emoji, URL, and punctuation). Another example is the Keras `preprocessing` module,[2] which organizes files in a tree of folders into the Keras internal dataset format by creating batches of text, including truncating the text to a preset maximum length. This belongs to "data preprocessing", but not "text preprocessing".

The rest of the paper is organized as follows. In Section 2, we outline the NLP-related application types—information extraction, end-user applications, and building block applications. In Section 3, we review and evaluate several commonly used text preprocessing practices. In Section 4, we provide examples from three types of specialized text corpora—technical datasets, social media data, and text with numerical ratings. In Section 5, we conclude by reemphasizing the

[1] https://www.ethnologue.com/.
[2] https://keras.io/api/preprocessing/.

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

importance of text preprocessing. General text preprocessing methods have merits, but for further improvement of data quality, text preprocessing needs to be tailored to the particular dataset.

2. NLP application types

NLP applications are generally divided into three types: information extraction, end-user applications, and building block applications (Jusoh 2018). Information extraction retrieves useful information from the text corpus (Tang *et al*. 2008); such applications include information retrieval, topic modeling, and classification tasks (Albalawi, Yeap, and Benyoucef 2020). End-user applications take input directly from human users and provide output to them (Shaikh *et al*. 2019). These require more comprehension in machine reading; examples are machine translation, question-answering, reasoning, text summarization, and sentence condensation (Zeng *et al*. 2020). Building block applications enhance the performance of the first two types of NLP applications (Taboada *et al*. 2013), and common building blocks in NLP are part-of-speech (POS) tagging, named entity recognition (NER), and dependency parsing (Alonso, Gómez-Rodríguez, and Vilares 2021). Our discussion on text preprocessing methods in this article is built to serve a wide range of NLP applications. We start with an information extraction standpoint because its text preprocessing methodology is relatively straightforward (Adnan and Akbar 2019a), then we explain why some preprocessing methods are inappropriate for end-user applications. We also explain how text preprocessing contributes to the success of building block NLP applications, and eventually to the text model performance (Liu and Özsu 2009).

For information extraction, most text preprocessing methods would suffice for constructive results (Allahyari *et al*. 2017; Kalra and Aggarwal 2018), but customized preprocessing methods can further improve the performance of information extraction (Adnan and Akbar 2019b). As an example of a successful preprocessing application, Yazdani *et al*. (2020) built an automated misspelling correction system for Persian clinical text, with up to 90 percent detection rate and 88 percent correction accuracy. Using the Sastrawi library stemmer[3] also improves the exact match rate in Indonesian to 92 percent, compared with 82 percent by using the Porter stemmer (Rosid *et al*. 2020). If the goal of information extraction is to reveal the semantic structure of text for further end-user applications, the preprocessing methods also need to cater to the latter (Grishman 2015).

For end-user applications, text preprocessing is still crucial but **how** the methods are implemented is of paramount importance (Kulkarni and Shivananda 2019; Chang *et al*. 2020). Tokenizing the corpus can identify words, and sentence splitting can find sentence boundaries (Zhang and El-Gohary 2017). Such segmentation of the corpus text is helpful in machine comprehension, especially for question-answering and text summarization (Widyassari *et al*. 2022). Stemming and stopword removal are useful to narrow down the search space, but the system needs to output full sentences to respond to the end user (Babar and Patil 2015; Lende and Raghuwanshi 2016). However, applying some preprocessing methods in the wrong way can be detrimental to end-user applications. For instance, removing punctuation too early from the corpus will result in failure to identify sentence boundaries, leading to inaccurate translation (Peitz *et al*. 2011). In multilingual question-answering, the mix of different languages requires special handling in the preprocessing phase, otherwise the system will have a large number of out-of-vocabulary (OOV) words from the default single language (Loginova, Varanasi, and Neumann 2018).

For building block applications, adequate text preprocessing is necessary to leverage these NLP building blocks to their full potential (Thanaki 2017; Sarkar 2019), while improper choices in text preprocessing can hinder their performance (Reber 2019). For example, the accuracy of POS tagging can generally be improved through spelling normalization (Schuur 2020), especially in historical texts where archaic word forms are mapped to modern ones in the POS training

[3] https://pypi.org/project/Sastrawi/.

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

database (Bollmann 2013). NER can benefit from the detection of multiword expressions, since an entity often contains more than one word (Tan and Pal 2014; Nayel *et al*. 2019). On the other hand, tokenization errors can lead to difficulties in NER (Akkasi, Varoğlu, and Dimililer 2016). For instance, "CONCLUSIONGlucose" should be segmented as "conclusion" and "glucose", but splitting on letter case change will result in a partial entity "lucose". Finally, although removing stopwords is helpful in information retrieval (El-Khair 2017), this preprocessing method hurts dependency parsing (Fundel, Küffner, and Zimmer 2007) because it may destroy the dependencies between entities such as prepositions (Agić, Merkler, and Berović 2013).

We also briefly explain the training of word embeddings because the process is similar to the text mining tasks (Jiang *et al*. 2015; Ye *et al*. 2016). Word2vec (Mikolov *et al*. 2013) creates a vector representation to intuitively measure the similarity between words. Both continuous bag-of-words model and continuous skip-gram model are used to predict the nearby words given the current word. GloVe (Pennington, Socher, and Manning 2014) leverages the conditional probability for word frequency in a word–word co-occurrence matrix, and the dimensionality reduction contributes to better performance. Embeddings from Language Model (Peters *et al*. 2018) uses Long Short-Term Memory (LSTM) to capture context-dependent word meanings, allowing for richer word representations. BERT (Devlin *et al*. 2018) is the state-of-the-art language representation model, and it pretrains deep bidirectional representations in more than 100 languages. BERT uses a masked language model for bidirectional conditioning and predicts the next sentence for question-answering. BERT also supports cased and uncased versions of models (Kitaev, Cao, and Klein 2019; Ji, Wei, and Xu 2020), and we will discuss more about letter case normalization in Section 3.3.

Nevertheless, word embeddings with neural networks are not a cure-all solution for NLP applications (Abraham *et al*. 2018; Agre, van Genabith, and Declerck 2018) for two reasons: the first reason is the necessary text preprocessing, and the second reason is the limitations of word embeddings themselves. Segmenting text into words (i.e., tokenization) is a prerequisite of creating word embeddings (Kudo and Richardson 2018). In text ranking with BERT, document preprocessing reduces the data size of potentially relevant information in the corpus, making computationally expensive models scalable (Lin, Nogueira, and Yates 2020). Woo, Kim, and Lee (2020) also validated combinations of text preprocessing techniques to optimize the performance of sentence models. But even with the best intention and preparation, word embeddings still have limitations in applications such as reasoning (i.e., natural language inference) (Zhou *et al*. 2020). Word embeddings also face difficulties in low-resource scenarios (Hedderich *et al*. 2020) such as minority languages (e.g., Tibetan) (Congjun and Hill 2021), due to an insufficient corpus on the language itself. Finally, potential bias in the data can propagate to the word embeddings, leading to unintended consequences such as unfair or discriminatory decisions (Papakyriakopoulos *et al*. 2020; Basta, Costa-jussà, and Casas 2021).

3. Commonly used text preprocessing practices

Extensive information is available for commonly used text preprocessing practices, including books and programming documentation (Lambert 2017). Open-source tools in Python include the natural language toolkit NLTK (Bird, Loper, and Klein 2009) and `scikit-learn` (Pedregosa *et al*. 2011) for machine learning. Both packages have been continuously maintained and updated over the past decade. In addition to Python, R is also a popular tool in text modeling, with the book *Text Mining with* R*: A Tidy Approach* (Silge and Robinson 2017). These resources provide guidance to alleviate the pain of text preprocessing, but manual work is still required even with the aid of integrated software like H2O.ai[4] or Microsoft Azure Machine Learning Studio. Many parameter settings are available for fine-tuning, and for best results, different types of text corpora

[4] http://docs.h2o.ai/driverless-ai/latest-stable/docs/userguide/nlp.html.

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

require different preprocessing methods (Yuhang *et al*. 2010). However, to the best of our knowledge, there is not a set of comprehensive guidelines that can advise researchers on which text preprocessing practices to apply to a brand-new text corpus.

Hence, we would like to review some text preprocessing techniques and evaluate their strengths and weaknesses in terms of NLP so that researchers can determine which methods are most appropriate for their text data. For each method, we start with a brief description, explain its advantages and applications, and then discuss potential issues and situations when they are of concern. We can attempt to propose remedies, but a trade-off always exists between undercorrection and overcorrection. Each text corpus is different, and the goal of text mining also varies by project. This section is not a step-by-step execution guide on text preprocessing. We try to keep the methods sequential, but the methods discussed here do not have to be executed in the same order as in Figure 1. For instance, the information in punctuation is essential for question-answering (Ferret *et al*. 2002) and sentiment detection (Rosenthal and McKeown 2013), so for these tasks we should keep the punctuation in the corpus until much later stages.

3.1 Removing formatting

Removing formatting in text usually needs to be done before any other preprocessing. If the data came from web scraping, the text would contain HTML markup, which needs to be removed first. For example, the original text string with HTML tags can be " actual content ", and we want the "actual content" without the tags. The Python library `BeautifulSoup` (version 4.9.1 by Richardson 2020) is a popular tool for removing HTML tags, and the command to import this library is `from bs4 import BeautifulSoup`. In addition to the official documentation,[5] the book *Website Scraping with Python* (Hajba 2018) also contains a full chapter on using the `BeautifulSoup` library to extract and navigate content in HTML format. For the R community, the R package `textclean` (Rinker 2018b) is also available for removing formatting, such as the function replace_html. This package also replaces common web symbols with their text equivalents, such as "&cent" to "cents" and "&pound" to "pounds". Finally, regular expressions can remove a wide range of text patterns, such as a person's email signatures and "[8:05 AM]" in chat messages. Most programming languages support regular expressions, and manual preprocessing offers greatest flexibility in removing formatting. However, manual preprocessing using regular expressions is not only time-consuming but also error-prone (Shalaby, Zadrozny, and Jin 2019). This can easily introduce unwanted and unexpected artifacts to a corpus (or some parts of it). Hence, we recommend doing so only when the patterns cannot be handled by standard libraries, which are more rigorously tested for correctness (Goyvaerts and Levithan 2012).

3.2 Tokenization

Tokenization decomposes each text string into a sequence of words (technically tokens) for computational analysis (Singh and Saini 2014; Al-Khafaji and Habeeb 2017), and the choices in tokenization are more important than many researchers have realized (Habert *et al*. 1998; Verspoor *et al*. 2009). Given the sentence "I downloaded this software on-the-fly", how many words will it contain after tokenization? The question boils down to whether "on-the-fly" is regarded as a compound word or separated into three words "on", "the", and "fly". Although a white space between two words is often used as an explicit delimiter (Webster and Kit 1992), we should not simply tokenize on white space (Clough 2001). According to Barrett and Weber-Jahnke (2011), there is not a universal tokenization method for English texts, not to mention other languages with different linguistic features. The Python NLTK tokenizer package is a useful tool to separate a text string into word tokens, but researchers still have to make subjective

[5] https://www.crummy.com/software/BeautifulSoup/.

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

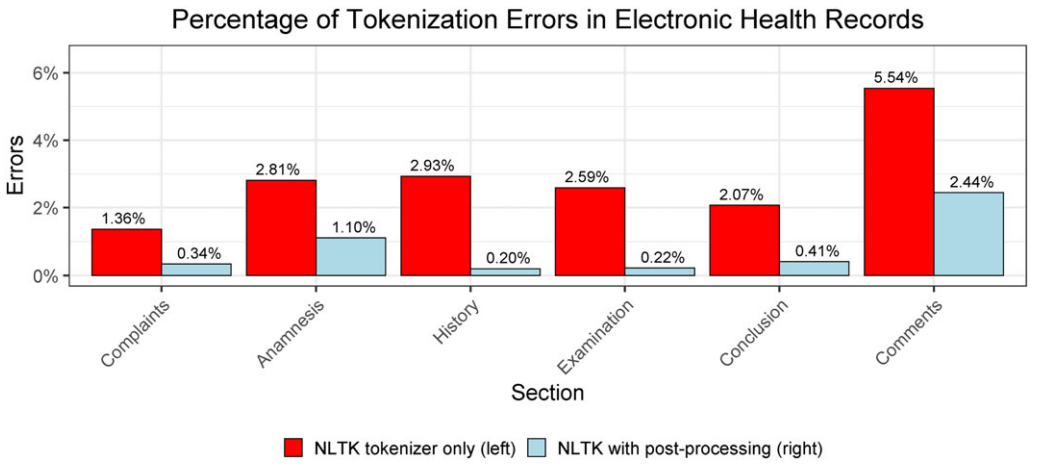

Figure 2. Postprocessing after using the NLTK tokenizer reduces the tokenization errors. The data are from Table 1 in Grön and Bertels (2018).

decisions based on the corpus (Nugent 2020). Trieschnigg *et al.* (2007) showed that these subjective decisions, including tokenization, can contribute much more to model performance than the algorithm choice.

According to Cohen *et al.* (2019), tokenization options can make the difference between getting a publishable result or not. For example, the type-token ratio is used to measure lexical richness, which indicates the quality of vocabulary in a corpus (Malvern and Richards 2012). Type-token ratio is calculated by dividing the number of types (distinct words) by the number of tokens (total words). But the terms "type" and "token" are loosely defined, and questions arise when we make decisions to count "on-the-fly" as one word or three words. Then, some researchers try various definitions of "type" and "token" and select one that produces a statistically significant result (Cohen *et al.* 2019). Such strategy is essentially p-hacking and hurts reproducibility in research (Head *et al.* 2015). A better practice is to be clear and consistent with the definitions of these terms in the tokenization process. Most existing literature does not specify the tokenization methods in detail, including some papers with actual implications for diagnosis of serious neurological and psychiatric disorders (Posada *et al.* 2017; Nasir *et al.* 2020). Therefore, we need to think carefully about the potential impact from NLP to the published results, especially in clinical settings. We will further discuss the tokenization challenges for biomedical text in Section 3.2.1 and for various natural languages in Section 3.2.2.

3.2.1 Tokenization for biomedical text

In this section, we use biomedical text as an example of how domain-dependent tokenization can be done and the impact it has on downstream tasks. Biomedical text can be regarded as a sublanguage because it is substantially different than documents written in general language. The former contains more syntactic use of word-internal punctuation, such as "/" (forward slash) and "-" (dash) among the biomedical terms (Temnikova *et al.* 2013). This phenomenon applies not only in English but also in other languages such as Bulgarian, French, and Romanian (Névéol *et al.* 2017; Mitrofan and Tufiş 2018). Moreover, Grön and Bertels (2018) showed that many errors in processing biomedical clinical text are due to missing white space or nonstandard use of punctuation, such as "2004:hysterectomie" and "2004 : hysterectomie". A custom script for postprocessing can reduce the tokenization errors, after the corpus of electronic health records is divided into words using the NLTK tokenizer. Figure 2 illustrates that the percentage of tokenization errors is reduced in the sections of complaints, anamnesis, history, examination, conclusion,

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

Table 1. Number and percentage of false positives for each type of pattern matching. The data are from Table 3 in Cohen *et al*. (2002), and the rows after the first one refer to the extra tokens discovered beyond strict pattern matching

Type of pattern matching	Tokens Discovered	True Positives	False Positives	Percentage of False positives
Strict pattern matching	500	425	75	15.00%
Case-insensitive	97	72	25	25.77%
Vowel sequence	100	15	85	85.00%
Parentheses as optional	99	93	6	6.06%
Plural matches	86	75	11	12.79%
Hyphen as optional	37	34	3	8.11%

and comments. The data are from Table 1 in Grön and Bertels (2018). The postprocessing includes Greek letter normalization and break point identification (Jiang and Zhai 2007). For instance, "MIP-**1**-α" and "MIP-**1**α" should both tokenize to "MIP 1 alpha". Note that the hyphen does more than separating elements in a single entity, so simply removing it is not always appropriate. The hyphen in parentheses "(-)" can also indicate a knocked-out gene, such as "PIGA (-) cells had no growth advantage" in Cohen *et al*. (2005).

Trieschnigg *et al*. (2007) also pointed out that tokenization decisions can contribute more to system performance than the text model itself. The text string "NF-κ B/CD28-responsive" has at least 12 different tokenization results, depending on the preprocessing techniques used. The various approaches in tokenization resulted in up to 46 percent difference in the precision for document retrieval. The baseline version "nf κ b cd responsive" keeps only lowercase letters without numbers, and this achieves 32 percent precision in document retrieval. Another version "NF-κ B/CD28-responsive" has only 17 percent, where the text string is separated by white space without further normalization. A custom tokenizer result "nf kappa nfkappa b cd 28 respons bcd28respons" has 40 percent precision, which is the highest among the 12 combinations attempted by Trieschnigg *et al*. (2007). The last version replaces Greek letters with their English names and stems the word from "response" to "respons". This version also regards "nf", "kappa", and "nfkappa" as different tokens to increase the likelihood of getting a match. The same applies to the separate tokens "b","cd", "28", "respons", and the combined token "bcd28respons". In addition to precision, we should also examine the recall to capture as many relevant gene names as they exist in the biomedical corpus.

Biomedical information retrieval often incorporates approximate string matching (a.k.a. fuzzy matching) for gene names (Morgan *et al*. 2008; Cabot *et al*. 2016), because this allows small variations to be considered as the same gene name. Verspoor, Joslyn, and Papcun (2003) discovered that approximately 6 percent of gene oncology terms are exact matches in the biomedical text, showing the necessity of non-exact matches to find the remaining 94 percent. For example, Figure 3 depicts that tokenization choices of the gene name "alpha-2-macroglobulin" lead to different results of pattern matching in biomedical text. With a strict pattern matching heuristic, there were 1846 gene names found in the corpus. Then, more flexible pattern matching methods can find additional gene names and increase the recall. For instance, the case-insensitive heuristic found an extra 864 gene names, and the vowel sequence heuristic[6] discovered an extra 586 matches. The data in Figure 3 are from Table 2 in Cohen *et al*. (2002), and the concern of low

[6]The vowel sequence heuristic maps each vowel sequence to one or more of any vowel to reduce the dialectal differences in gene names. For example, "h**e**moglobin" and "h**ae**moglobin" are regarded as the same gene under this heuristic.

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

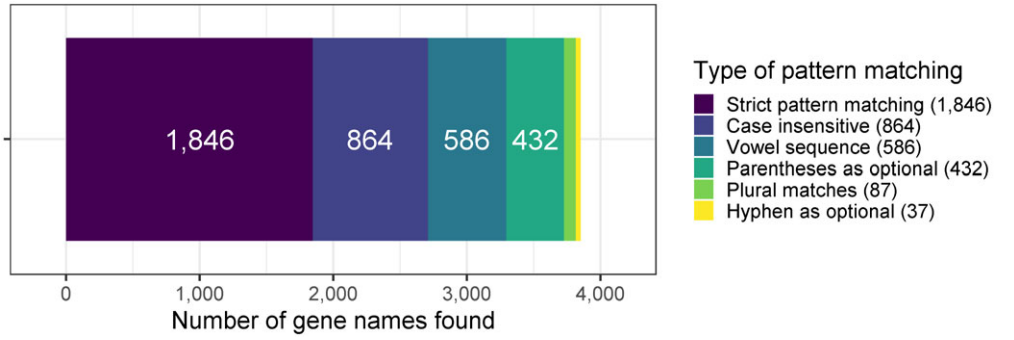

Figure 3. Tokenization decisions in the gene "alpha-2-macroglobulin" lead to different pattern matching results. The data are from Table 2 in Cohen *et al.* (2002).

precision (i.e., false positives) can be mitigated by discarding weaker matches. For each type of pattern matching used, Table 1 lists the number and percentage of false positives in the discovered tokens. The data are from Table 3 in Cohen *et al.* (2002), and the rows after the first one refer to the extra tokens discovered beyond strict pattern matching. Most pattern matching schemes generated fewer than one-third of false positives, except that the vowel sequence heuristic produced 85% of false positives.

Approximate string matching in tokenization is also helpful in NER of biomedical and chemical terms (Akkasi *et al.* 2016; Bhasuran *et al.* 2016; Kaewphan *et al.* 2017), but the tokenization methods still have room for improvement. Both Cohen *et al.* (2004) and Groza and Verspoor (2014) show that choices in handling punctuation affect the tokenization results and eventually affect NER as well. Term variation can easily result in poor results of biomedical concept recognition, especially in noncanonical forms. A slight change such as "apoptosis induction" versus "induction of apoptosis" can result in the two entities being assigned into different equivalence classes (Cohen *et al.* 2010).

Pretrained word embeddings for biomedical text are gradually increasing in popularity (Wang *et al.* 2018), and one example is the BioBERT (Lee *et al.* 2020), which is pretrained on PubMed abstracts and PMC (PubMed Central) articles. Since different language models have different requirements on letter casing and punctuation, the BERT model (Devlin *et al.* 2018) supports multiple variants of input catering to various NLP applications (Ek, Bernardy, and Chatzikyriakidis 2020). Nevertheless, researchers still need to tokenize biomedical text into words before they can leverage pretrained embeddings and do it in accordance with the preprocessing protocol used in preparing the embedding training corpus (Corbett and Boyle 2018; Pais *et al.* 2021). Another challenge in leveraging these pretrained word embeddings is that word vectors may not reflect the internal structure of the words. For instance, the related words "**delta**proteobacteria" and "**beta**proteobacteria" should be close (but not too close) in the embedding space; however, this relationship is not accounted for in the word2vec or GloVe models. Zhang *et al.* (2019) proposed BioWordVec to leverage the subword information in pretraining, allowing the model to better predict new vocabulary from the subwords in the corpus.

3.2.2 Tokenization for various natural languages

Beyond general English and biomedical texts, tokenizing non-English corpora also has challenges that can affect the text mining performance. Most languages have compound terms that span multiple tokens, such as "sin embargo" (however) in Spanish and "parce que" (because) in French (Grana, Alonso, and Vilares 2002). Habert *et al.* (1998) applied eight different tokenization methods on a French corpus of size 450,000 and attempted to concatenate the compound terms (see

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

Section 3.7 for multiword expressions). The number of words in the corpus differed by more than 10 percent, while the number of sentences could range from approximately 14,000 to more than 33,000. French also has contractions similar to English (e.g. "don't" means "do not") (Hofherr 2012). The article "l" (the) can be attached to the following noun, such as "l'auberge" (the hostel). The preposition "d" (of) works similarly, such as "noms d'auberge" (names of hostels). More details on handling word-internal punctuation will be discussed in Section 3.4.2.

Consider the case of Arabic tokenization. In Arabic, a single word can be segmented into at most four independent tokens, including prefix(es), stem, and suffix(es) (Attia 2007). One example is the Arabic word "وكتابنا" ("and our book", or "wktAbnA" as the Buckwalter transliteration). This word is separated into three tokens: the prefix "و" "w" (and), the stem "كتاب" "ktAb" (book), and a possessive pronoun "نا" "nA" (our) (Abdelali *et al*. 2016). Note that Arabic writes from right to left, so the first token starts from the rightmost part of the word (Aliwy 2012). Most text preprocessing systems now have a simple configuration for right-to-left languages (Litvak and Vanetik 2019), and the Python NLTK module `nltk.corpus.reader.udhr` supports major right-to-left languages including Arabic and Hebrew.[7]

It is challenging to tokenize CJK languages because they use characters rather than letters, and each language contains thousands of characters (Zhang and LeCun 2017). As a result, CJK tokenizers often encounter the OOV problem, and it is possible to get OOV characters as well as OOV words (Moon and Okazaki 2021). Plus, CJK languages do not contain white space as obvious word boundaries in the corpus (Moh and Zhang 2012). Researchers have attempted to mitigate these problems by borrowing information from a parallel corpus of another language, commonly in multilingual corpora for translation (Luo, Tinsley, and Lepage 2013; Zhang and Komachi 2018). Thanks to recent advances in neural networks and pretrained models like BERT (Devlin *et al*. 2018), there has been progress in identifying CJK words that span multiple characters (Hiraoka, Shindo, and Matsumoto 2019). Moreover, tokenization results can be improved by leveraging subword information within the same language (Moon and Okazaki 2020), and even subword pooling from other similar languages (Ács, Kádár, and Kornai 2021). Tokenization in multiple languages helps not only in machine translation (Domingo *et al*. 2018) but also in adversarial text generation (Li *et al*. 2020).

3.3 Text normalization

After the corpus is tokenized into words, text normalization is often the next step in preprocessing (Zhang *et al*. 2013). Text normalization is defined as mapping noncanonical language to standardized writing (Lusetti *et al*. 2018), and this consolidates text signals and decreases sparsity of the search space (Bengfort, Bilbro, and Ojeda 2018). Text normalization is especially useful in dealing with large amounts of nonstandard writings, such as social media data (Baldwin and Li 2015; Lourentzou, Manghnani, and Zhai 2019) and speech-to-text output in automatic speech recognition (Yolchuyeva, Németh, and Gyires-Tóth 2018; Chua *et al*. 2018). One example is automatic correction of misspellings (Tan *et al*. 2020), for example "mountian" becomes "mountain"; otherwise, these misspellings will greatly increase the number of OOV words (Piktus *et al*. 2019). Appropriate letter casing is also beneficial in NER, because it is easier to recognize named entities in this form (Bodapati, Yun, and Al-Onaizan 2019). Generally speaking, stemming and lemmatization (Section 3.6) both belong to the text normalization area (Korenius *et al*. 2004; Samir and Lahbib 2018), and the most common form of letter case normalization is converting the entire corpus to lowercase letters (Manning, Raghavan, and Schütze 2008). This is not only due to its prevalence and availability in programming languages (Thanaki 2017) but also due to its demonstrated success in text classification performance (HaCohen-Kerner *et al*. 2020). But despite its

[7] https://www.nltk.org/_modules/nltk/corpus/reader/udhr.html.

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

popularity, converting everything to lowercase can be problematic in certain NLP applications like text summarization and sentence boundary disambiguation, where uppercase at the beginning of the text may indicate the start of a sentence (Abdolahi and Zahedh 2017).

The main advantages of letter case normalization are consistency and consolidation of word variation (Şeker and Eryiğit 2012). For instance, "Large" and "large" would be recognized as the same word because the letter case does not change the meaning of the word (Bokka *et al.* 2019; Rahm and Do 2000). Some obviously proper nouns can also benefit from the lowercasing; for example "Europe" and "europe" refers to the exact same thing, so the two tokens can be consolidated into one. Note that different languages have different capitalization schemes; for example, German capitalizes all nouns, not just proper nouns as in English (Labusch, Kotz, and Perea 2022). In biomedical data, case and format normalization improves the recall of finding a match of gene names, because a gene often has multiple variations of its name (Section 3.2.1 and Cohen *et al.* 2002). Even in neural networks, conversion to lowercase keeps the text feature extraction simple and reduces the number of distinct tokens (Preethi 2021). Lowercasing is also used in information retrieval because search queries do not have accurate capitalization, so the query cannot rely on capital letters to match against a corpus (Barr, Jones, and Regelson 2008). Especially in many speech recognition systems, the user's input does not have inherent capitalization (Beaufays and Strope 2013), which often results in all lowercase words in the query.

However, converting to lowercase can result in loss of semantic information where the letter capitalization indicates something other than the lowercase word. Some acronyms have different meanings than the same spelling in all lowercase—for example, "US" (United States) and "us" (a first-person plural pronoun). In addition to acronyms, many nouns have different meanings when they are in proper case, compared with in lowercase. Examples include last names such as Bush, Cooper, and Green. As a result, lowercasing is not very useful in NER, because the text model needs to identify proper nouns from the letter cases (Campbell *et al.* 2016). The lack of letter case information also contributes to the POS mistagging rate (Foster *et al.* 2011). In these NLP applications, truecasing (Lita *et al.* 2003) is needed to achieve a balance between consolidating word variations and distinguishing proper nouns (Duran *et al.* 2014). Moreover, words in ALL-CAPS can also be used to emphasize a strong emotion, so the capitalization is related to higher sentiment intensity. For example, the comment "The conference is AWESOME!" conveys a stronger emotion than "The conference is awesome!". Hutto and Gilbert (2014) compared the ratings on a Likert scale (from positive to negative) from comments on social media and discovered that the comments with ALL-CAPS express higher sentiment intensity than the ones without.

Handling accented words in languages like French and Spanish faces similar issues as converting letters to lowercase (Zweigenbaum and Grabar 2002). Both text preprocessing steps normalizes the corpus at the character level, that is, character normalization. De-accenting works like lowercasing; the process replaces all accented characters (e.g., "àáâä") with non-accented ones (e.g., "a"). This is helpful in information retrieval because words in queries may not be appropriately accented (Grabar *et al.* 2003), but some semantic information is unavoidably lost. For instance, consider the Spanish word "té" (tea) with an accent and the word "te" (reflexive pronoun of "you") without an accent. Removing the accent will map both words to the same token "te", and the information of "tea" disappears. How to re-accent words from an unaccented corpus has been a challenging research problem (Zweigenbaum and Grabar 2002b; Novák and Siklósi 2015).

3.4 Handling punctuation

Although many researchers remove punctuation in a text corpus at the beginning of text preprocessing (Kwartler 2017), punctuation conveys information beyond the words and should not always be ignored. Punctuation separates word strings to clarify meaning and conveys semantic information to human readers (Baldwin and Coady 1978). Here is an example from Truss

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

(2006): "Go, get him doctors!" means telling someone to find doctors for a male patient, because the comma separates the two clauses "Go" and "get him doctors". In comparison, "Go get him, doctors!" means telling the doctors to catch a guy, because the clause "Go get him" is a command directed at the doctors. In NLP, punctuation also provides syntactic information for parsing, such as identifying complex sentences (Jones 1994) and tokenizing biomedical terms (Díaz and López 2015). For example, a string "cd28-responsive" can be parsed into "cd28" and "responsive" (split words on the dash), or in a single token "cd28-responsive" (Trieschnigg *et al*. 2007).

According to *Corpus Linguistics* (Lüdeling and Kytö 2008), punctuation is generally divided into three categories: sentence-final punctuation, sentence-internal punctuation, and word-internal punctuation.

(1) Sentence-final punctuation indicates the end of a sentence, such as a period, an exclamation mark, and a question mark.
(2) Sentence-internal punctuation are used in the middle of a sentence, including commas, parentheses, semicolons, colons, etc.
(3) Word-internal punctuation exists within a word, and examples include apostrophes and dashes.

These categories are neither mutually exclusive nor exhaustive; that is, a punctuation mark can belong to multiple categories or neither. For instance, languages such as Spanish have sentence-initial punctuation, which indicates sentence boundaries like sentence-final punctuation. One example is the "¿" (the inverted question mark) in the sentence "¿Dónde está el baño?" (Where is the bathroom?) But "¿" can also be used as sentence-internal punctuation, for example "Ana, ¿qué haces hoy?" (Ana, what are you doing today?) (Miller 2014). English has complexity in punctuation as well. The period often marks the end of a sentence, but it can also be used in word-internal abbreviations, such as "e.g." and "U.S.A." The apostrophe can exist in both word-internal (e.g., "don't") and sentence-final (the end of a single quotation mark). The overlap can cause ambiguity in text segmentation, that is, tokenizing the corpus to detect words and sentences (Grefenstette and Tapanainen 1994; Huang and Zhang 2009).

The level of importance of breaking the corpus into sentences varies in NLP applications. For instance, text classification and topic modeling focus on the text documents, so each sentence itself is less of a concern (Blei, Ng, and Jordan 2003; Korde and Mahender 2012). But on the other hand, certain end-user applications rely heavily on the sentences from the corpus breakdown: text summarization needs sentence extraction as a prerequisite (Patil *et al*. 2015); machine translation of documents requires sentence segmentation (Matusov *et al*. 2005; Kim and Zhu 2019); and question-answering utilizes the syntactic information from sentences (Li and Croft 2001). When breaking the corpus into sentences is important, this is often done first before we can continue the rest of preprocessing, such as identifying the entity's POS from a sentence. Breaking into sentences is harder than it appears, due to not only the multiple functions of the period but also the non-standard usage of punctuation in informal text (Villares-Maldonado 2017). Methods for sentence boundary disambiguation include maximum entropy approach (Reynar and Ratnaparkhi 1997; Le and Ho 2008), decision tree induction (Kiss and Strunk 2006; Wong, Chao, and Zeng 2014), and other trainable sentence segmentation algorithms (Indurkhya and Damerau 2010; Sadvilkar and Neumann 2020).

We need to distinguish separating punctuation from strings (Section 3.4.1) and removing punctuation (Section 3.4.2). Separating punctuation from strings means that we split a string into words or shorter strings based on the punctuation. Splitting a string into words on punctuation is essentially tokenization (see Section 3.2). This can be straightforward for a simple sentence like "I have a dog." or "Cindy is my sister, and David is my brother." But there is confusion in splitting the sentence "Then Dr. Smith left." The first period is word-internal punctuation, while the second

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

period is sentence-final punctuation. In the biomedical domain, tokenizing the sentence "I studied E. coli in a 2.5 percent solution." will cause problems because the period following the uppercase "E" indicates an acronym rather than the end of the sentence. Also, "E. coli" is a biomedical term (Arens 2004). When we separate a string into shorter strings via punctuation, discourse parsing is also of interest to understand the hierarchical and syntactic structure of the text string (Soricut and Marcu 2003; Peldszus and Stede 2015). On the other hand, removing punctuation means that we permanently eliminate punctuation from the corpus. Then, the corpus is reduced to words and numbers, and each token is separated by blank space. The reduced dataset may be easier to analyze in some cases, but researchers should be cautious in removing punctuation because the lost information will not return to the corpus later. When punctuation contains emotional context such as repeated exclamation marks "!!!" (Liu 2015), removing punctuation can be detrimental in sentiment analysis, especially in social media data (Koto and Adriani 2015).

3.4.1 Separating punctuation from strings

Researchers often need to separate punctuation from strings in text preprocessing—not only to obtain the words but also to retrieve the syntactic information conveyed in the punctuation (Nunberg 1990; Briscoe 1996). For example, in the sentence "We sell cars, scooters, and bikes.", the two commas separate the noun objects and the period indicates the end of the sentence. This shows that punctuation provides grammatical information to POS tagging (Olde *et al.* 1999). Note that inconsistent use of punctuation can be worse than no punctuation (Bollmann 2013), and in this case, discarding punctuation is preferable. Furthermore, using punctuation to separate text into shorter strings is helpful in machine translation, especially for long and complicated sentences (Yin *et al.* 2007). Parsing punctuation as part of input not only improves the quality of translation between European languages (Koehn, Arun, and Hoang 2008) but also provides bilingual sentence alignment for a Chinese-English corpus (Yeh 2003).

We would like to highlight the application of discourse parsing, because this is a difficult research problem central to many tasks such as language modeling, machine translation, and text categorization (Joty *et al.* 2019). Discourse parsing needs the punctuation information to identify the relations between segments of text (Ji and Eisenstein 2014). Punctuation marks can serve as discourse markers to separate a text string into shorter parts, along with many conjunction words (Marcu 2000). A sentence with or without sentence-internal punctuation can have different meanings, and here is an example from Truss (2004): "A panda eats shoots and leaves." means that a panda eats shoots and also eats leaves. But the sentence with a comma, "A panda eats, shoots and leaves.", means that a panda eats something, shoots a gun, and leaves the scene. Figure 4(a) and (b) depict the two hierarchical structures, respectively. The syntactic structure from punctuation feeds into discourse structure (Venhuizen *et al.* 2018). For more complicated text structures, Ghosh *et al.* (2011) leveraged a cascade of conditional random fields to automate discourse parsing, based on different sets of lexical, syntactic, and semantic characteristics of the text.

Another issue we would like to discuss is the challenge of parsing nonstandard use of punctuation in social media text (Farzindar and Inkpen 2020). Text messages often contain repeated punctuation such as "!!" and "??", and these symbols are typically used as an emphasis of emotion, without adding extra lexical information (Liu and Jansen 2013). This emotional context is helpful in sentiment analysis (Rosenthal and McKeown 2013), while in many other applications, it is acceptable to map repeated punctuation to a single one (Verma and Marchette 2019). We also need to be careful in processing emoticons because each of them should be assigned to a token, rather than being separated into characters (Rahman 2017). We recommend starting with a predefined list of emoticons to identify them in the text corpus, and one example list is available on Wikipedia.[8] Emoticons that contain all punctuation are easier to detect, such as ":)" and ":-(".

[8] https://en.wikipedia.org/wiki/List_of_emoticons.

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

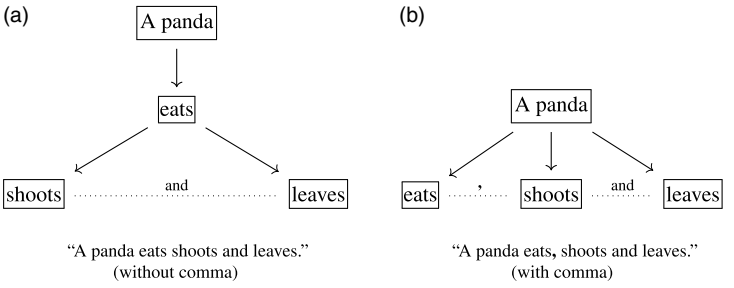

Figure 4. Hierarchical structure of a sentence with or without a comma. The sentence example is from Truss (2004).

On the other hand, emoticons that contain both letters and punctuation are relatively harder to detect, such as ":p" and ";D" (Clark and Araki 2011). In addition, the symbol @ can be part of an emoticon "@_@" or can be used as a mention to another user on social media. Similarly, the symbol # can be part of another emoticon "#)" or serve as the start of a hashtag. More about mentions (@username) and hashtags (#hashtag) will be discussed in Section 4.2.

Finally, if researchers are unsure whether the information from punctuation would be needed in their NLP work, we recommend separating punctuation from strings first. In this way, researchers still have the option to remove the punctuation tokens later. In comparison, if researchers remove punctuation from the corpus in the beginning, it is much harder to restore the punctuation unless they are willing to restart from the raw data.

3.4.2 Removing punctuation

Removing punctuation generally simplifies the text analysis and allows researchers to focus on words in the text corpus (Gentzkow, Kelly, and Taddy 2017; Denny and Spirling 2018). According to Carvalho, de Matos, and Rocio (2007), removing punctuation improves the performance of information retrieval. In NLP applications where punctuation is not of primary interest, it is easier to remove punctuation from the corpus than to normalize the nonstandard usage of punctuation for consistency (Cutrone and Chang 2011; Fatyanosa and Bachtiar 2017). This is why many introductory books on NLP include removing punctuation as an early step of text preprocessing (Dinov 2018; Kulkarni and Shivananda 2019).

Moreover, many text mining models do not handle nonword characters well, so removing these characters beforehand is helpful in such situations. For instance, word2vec (Mikolov *et al.* 2013) takes a text corpus as input and produces efficient word representations in vector space, but word2vec does not support punctuation due to the optimized design toward words. Even when a text mining model allows nonword characters, it treats a word with punctuation as a different token than the same word without punctuation. Hence, the punctuation symbols can hurt the performance of modern parsers. The problem is that when punctuation exists in the input text, many parsers heavily rely on this feature, making it difficult to handle sentences with nonstandard use of punctuation. As a result, prediction suffers from punctuation inconsistency, and previous researchers have referred to the situation as "a nightmare at test time" (Søgaard, de Lhoneux, and Augenstein 2018). Removing punctuation as feature deletion improves the robustness of a model (Globerson and Roweis 2006).

Nevertheless, removing punctuation also removes the underlying semantic information from the corpus, which can have a negative effect on some NLP applications. In sentiment analysis, emotions in a sentence can be expressed from different punctuation, so removing punctuation would have a negative effect on the analysis results (Effrosynidis, Symeonidis, and Arampatzis 2017). For example, "He liked the cake!" is different than "He liked the cake?". Generally,"!"

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

expresses a stronger emotion, while "?" often reverses the intention of the sentence. Wang *et al.* (2014a) built a model using both words and punctuation to identify product reviews as positive or negative, and the model performed better than using only words without punctuation. In digital forensics, punctuation is also an important feature for authorship identification. The ALIAS (Automated Linguistic Identification and Assessment System) methodology performs many computation linguistic calculations to distinguish text written by different authors, and one aspect is the frequency and types of punctuation (Craiger and Shenoi 2007). A sentence in quotation marks can refer to a conversation or a quote from existing literature, so the writing style in that sentence can be highly different from the author's usual style. This information can be used to determine who said what (Pareti *et al.* 2013), and Thione and van den Berg (2007) developed a US patent to use the quotation marks in text for speaker attribution.

If researchers decide to remove punctuation from the text corpus, a straightforward way to do so is using regular expressions. This is part of string processing in almost all programming languages. A starting list is the ASCII printable characters,[9] which consist of a white space, punctuation, uppercase English letters, lowercase English letters, and the numbers 0–9. Note that different languages have different space characters (e.g., Chinese vs. English) (Craven 2004). In Python, the code `from string import punctuation` provides a string of commonly used punctuation symbols, including the open and close brackets. In R, the regular expression `[:punct:]` from the function `grep` also gives the same set of punctuation symbols as Python does. The R package `tm` (Feinerer and Hornik 2018) has a function `removePunctuation` for easy implementation.

We need to be careful in treating word-internal punctuation, especially contractions (e.g., "you're" vs. "you are"). Although most contractions drop out during the stopword removal phase since they are stopwords, some contractions have non-ignorable semantic meaning (e.g., "n't" means "not"). Python NLTK includes the package `contractions` to split contracted words, and GutenTag[10] as an extension toolkit makes it possible to preserve within-word hyphenations and contractions (Brooke, Hammond, and Hirst 2015). We can also map contractions into their non-contracted forms (e.g., map "don't" into "do not") using a predefined list,[11] and this is beneficial to negation handling (Anderwald 2003). For higher precision of contraction-splitting, we can use the Python library `pycontractions` (Beaver 2019) to incorporate context to determine the original form of ambiguous contractions. For example, the sentence "I'd like to know how I'd done that!" contains two "I'd"—the first one is from "I would" and the second one is from "I had". However, many researchers remove punctuation without handling the contractions (Battenberg 2012; Soriano, Au, and Banks 2013; Xue *et al.* 2020), so the demonstrations in this article do not include the contraction-mapping step.

3.5 Removing stopwords

Stopwords refer to the words that do not distinguish one text document from another in the corpus (Ferilli, Esposito, and Grieco 2014), and this concept was first developed by Hans Peter Luhn (1959). Examples include "the" and "or" because they are extremely common across documents, leading to little distinction among each document. Note that every stopword still has semantic content; for example, "the person" has a slightly different meaning than "a person". It is inappropriate to say that stopwords are meaningless, just because their content does not differentiate text documents in the corpus. Stopwords are usually removed in the text preprocessing stage (Rajaraman and Ullman 2011), so that text models can focus on the distinctive words for better performance (Babar and Patil 2015; Raulji and Saini 2016). Otherwise, these nondistinctive words

[9] https://theasciicode.com.ar/ascii-printable-characters/dot-full-stop-ascii-code-46.html.
[10] http://www.cs.toronto.edu/~jbrooke/gutentag/.
[11] https://gist.github.com/nealrs/96342d8231b75cf4bb82.

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

(i.e., stopwords) with high number of occurrences may distort the results of a machine learning algorithm (Armano, Fanni, and Giuliani 2015), especially in information retrieval (Zaman, Matsakis, and Brown 2011) and topic modeling (Wallach 2006). Removing stopwords greatly reduces the number of total words ("tokens") but not significantly reduces the number of distinct words, that is, vocabulary size (Manning *et al*. 2008). Hvitfeldt and Silge (2021) showed that even removing a small number of stopwords can moderately decrease the token count in the corpus, without a large influence on text mining models.

Nevertheless, stopwords are crucial for some NLP applications that require reasoning of text. In dependency parsing, stopwords are informative in understanding the connection between words in a sentence (Elming *et al*. 2013; Poria *et al*. 2014). For instance, the words "from" and "to" in "from Seattle to Houston" show the relationship between the nouns "Seattle" and "Houston" (Nivre 2005). Moreover, syntactic analysis of stopwords enables author identification (Arun, Suresh, and Madhavan 2009), and patterns of stopword usage can also be examined to detect plagiarism (Stamatatos 2011). An example of stopword-related feature is how often one uses pronouns to refer to a specific name (Sánchez-Vega *et al*. 2019).

Stopwords can be divided into two categories—general and domain-specific. General stopwords include prepositions (e.g., "at"), pronouns (e.g., "you"), determiners (e.g., "the"), auxiliaries (e.g., "was"), and other words with relatively less semantic meaning. A predefined stopword list provides general stopwords that can be removed from the corpus in preprocessing (Nothman, Qin, and Yurchak 2018). On the other hand, domain-specific stopwords are words with non-ignorable semantic meaning in general, but they appear in almost every document in the specific domain. For example, the words "method", "algorithm", and "data" are considered domain-specific stopwords in a text corpus of machine learning papers (Fan, Doshi-Velez, and Miratrix 2017). In addition to predefining a list of domain-specific stopwords for removal, we can also leverage the TF-IDF scores to identify the corpus-specific stopwords. These words typically have a high TF and/or a low IDF, making them too prevalent to distinguish one document from another in the corpus (Kaur and Buttar 2018). Other methods to automatically identify domain-specific stopwords include entropy (information theory) (Makrehchi and Kamel 2008; Gerlach, Shi, and Amaral 2019) and Kullback–Leibler divergence (Lo, He, and Ounis 2005; Sarica and Luo 2020).

3.5.1 Existing stopword lists

According to Manning *et al*. (2008), most researchers use a predefined stopword list to filter out the general stopwords, and the list typically includes fewer than 1000 words in their surface forms. For instance, "have" and "has" count as two separate stopwords in the list. Shorter lists include the Python NLTK (127 English stopwords) (Bird *et al*. 2009) and the R package `stopwords` (175 stopwords) (Benoit, Muhr, and Watanabe 2017). For longer lists, the Onix Text Retrieval Toolkit (Lextek International n.d) provides two versions—one with 429 words and the other with 571 words, where both lists are available in the R package `lexicon` (Rinker 2018a). Although Atwood (2008) optimized the SQL query performance for information retrieval by removing the 10,000 most common English words from their corpus, this is an extreme case and we do not recommend removing so many stopwords. Schofield, Magnusson, and Mimno (2017) showed that a list of approximately 500 stopwords would already remove 40–50 percent of the corpus. Zipf (1949) showed that the distribution of words is right-skewed, and that the word with the nth highest frequency is expected to appear only $1/n$ times as likely as the word with the highest frequency. Figure 5 is an illustration of the Zipf's law, assuming that the most frequent word appears 1 million times in a hypothetical corpus. Zhang (2008) and Corral, Boleda, and Ferrer-i Cancho (2015) also show that stopwords are often the most common words in the corpus, which applies to most languages like English, French, or Finnish.

An existing list often serves as a starting point for removing stopwords, and many researchers modify the list before use (Blanchard 2007; Heimerl *et al*. 2014), in order to optimize text mining

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

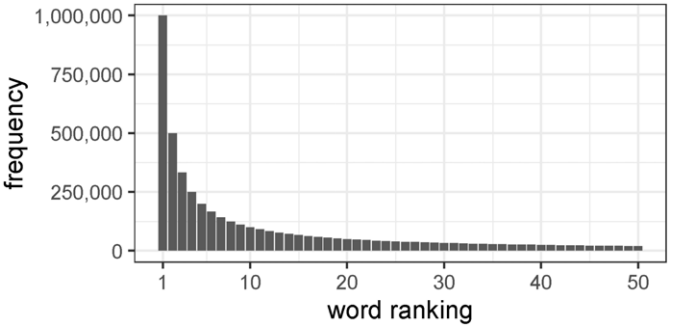

Figure 5. Illustration of the Zipf's law (Zipf 1949).

results (Amarasinghe, Manic, and Hruska 2015). For instance, since the list contains several words for negation (e.g., "no", "not"), removing these words would alter the meaning of the context. Then the negation words should be excluded from the stopword list, and more details are discussed in Section 3.5.2. In addition to standard stopword lists, alternative options include publicly available e-books, such as *Alice's Adventures in Wonderland* by Lewis Carroll, which can be used as a benchmark corpus to filter out nontechnical words from a technical database (Brooke *et al*. 2015). Project Gutenberg[12] contains more than 60,000 e-books, whose US copyrights have expired.[13] Even better, this corpus is directly available in the R package `gutenbergr` (Robinson 2018). Good candidates for a stopword list should be a book of general content, so the technical words in the database would be more likely to stay. Since a book usually contains thousands of words, using one book as a stopword list would be sufficient. Choosing multiple books provides limited benefits because most general words (e.g., "where" and "say") have already been removed by the first book.

There is a trade-off between using shorter stopword lists and longer ones, and it is up to the researchers to determine which works best for their text dataset. Using a shorter stopword list preserves more semantic information in the corpus, but the vocabulary size reduction may be insufficient. Using a longer stopword list further reduces the data, but words with potentially important information may also be accidentally eliminated from the corpus. Since words removed at the beginning would not return to the scope of data analysis, we need to be more cautious in using a longer stopword list (Fox 1989). Therefore, we recommend starting from a short stopword list to retain potentially important "stopwords". If the text mining results contain too many irrelevant words, we can gradually remove them by adding more words to the stopword list. This is a reversible process because if we find key insights are missing in the new version of the corpus, we can always select an earlier version with a smaller amount words removed.

3.5.2 Known issues with removing stopwords

Even in NLP applications where removing stopwords is safe and appropriate, we need to remember that the removed stopwords are permanently excluded from the corpus and will not re-enter the model (Yang and Wilbur 1996). Therefore, we have to beware of unintentional removal of important text information. Some phrases contain only stopwords, and the whole phrase is removed if we filter out stopwords at the beginning. Examples include the band "The The" and the phrase "to be or not to be" from *Hamlet*. Multiword expressions consisting of both stopwords and non-stopwords face a slightly better situation. For example, the novel title "Gone with the

[12] http://www.gutenberg.org/.
[13] https://www.copyright.gov/circs/circ15a.pdf.

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

Wind" becomes "Gone Wind" after we remove stopwords—the phrase still exists, but it is no longer understandable.

Being able to find stopwords is crucial for titles, especially in a search engine for media and/or e-commerce content (Lazarinis 2007). Although information retrieval generally ignores stopwords (Singhal 2001), Google Search allows the user to put a phrase in quotation marks for exact matching,[14] and so do most other search engines (White and Morris 2007; Wilson 2009). Hence in text preprocessing, when the subject indicates that the corpus may have stopword-formed phrases, n-gramming should be done first to preserve the semantic information of these phrases. Moreover, if the NLP application involves dependency parsing or semantic reasoning, we have to be careful in removing stopwords, especially prepositions (Hartrumpf, Helbig, and Osswald 2006). For example, the sentence "The box is in the house, and the toy is in the box." will mean something different if we remove both "in"s. Hansen *et al.* (2018) suggested not removing stopwords at all in syntax dependency parsing, because they may contain important syntactic information for sentence structure.

Finally, researchers should exclude "no" and "not" from the stopword list, because filtering out negation terms often reverts the meaning of the whole phrase. For example, the sentence "Zach doesn't know physics." becomes "zach know physics" (opposite meaning) after we remove the stopwords. It is possible to retain semantic information through antonym replacement or bigram creation. For instance, "not good" can be replaced with "bad" or "not_good" (as a bigram) (Perkins 2014; Chai 2017). However, neither approach is perfect because the scope of "not" may be different than the word immediately following the negation (Councill, McDonald, and Velikovich 2010). Consider the sentence "He was not driving the car and she left to go home." If we simply invert "not drive", we lose the context that someone else may be driving the car instead (Fancellu, Lopez, and Webber 2016). Automatically detecting the scope of negation is a challenging problem (Blanco and Moldovan 2011), and recent advances include using deep learning approaches (Qian *et al.* 2016) and bidirectional LSTM (Fancellu *et al.* 2017; Morante and Blanco 2021).

3.6 Stemming and lemmatization

Both stemming and lemmatization normalize words to their base forms to reduce the vocabulary size of a text corpus (Manning *et al.* 2008). For example, "enjoys" and "enjoyed" are assigned to the same token "enjoy". This decreases the space of word representation and improves the performance not only in information retrieval (Rajput and Khare 2015) but also in automatic text summarization (Torres-Moreno 2012). Liu *et al.* (2012) created a lemmatizer tool for biomedical text and achieved an accuracy of 97.5 percent in a given collection of biomedical articles. Stemming and lemmatization have gained success in many NLP applications (Khyani *et al.* 2020), but as a caveat, researchers are making a trade-off between semantic information and signal consolidation (Tabii *et al.* 2018).

3.6.1 Comparison of stemming and lemmatization

Before we compare stemming and lemmatization, we need to distinguish between inflectional and derivational morphology. Inflectional morphology refers to the word modification for grammatical purposes (e.g., verb tense, singular or plural noun form), without changing the word's meaning or POS (Baerman 2015). When we normalize the word "enjoyed" to its base form "enjoy", we are removing the inflectional morpheme from the word. On the other hand, derivational morphology adds a morpheme to the word and changes its semantic meaning and/or its POS (Crystal 1999). When we normalize the word "enjoyable" (adjective) to its base form "enjoy" (verb), the removed suffix is derivational because it changed the word from a verb to an adjective.

[14] http://www.googleguide.com/quote_operator.html.

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

Stemming uses rules to reduce each word to its root (stem), and this strips off both inflectional and derivational variations. For English words, a commonly used stemming method is the Porter algorithm (Porter 1980), along with the R package `SnowballC` (Bouchet-Valat 2019).[15] This algorithm implements suffix stripping rules for word reduction, such as replacing "ies" with "i" and removing the last "s" before a consonant other than "s" (making a plural noun singular). The Porter algorithm also uses regular expressions and determines which rule takes priority. For instance, "hopping" becomes "hop", but "falling" becomes "fall", not "fal". Due to the removal of suffixes, some tokens are not valid words and look like typos, such as "enjoi" (enjoy) and "challeng" (challenge). Other stemming algorithms include the Lovins stemmer (Lovins 1968) and the Lancaster stemmer (Paice and Hooper 2005).

Lemmatization groups the inflectional forms of each word to its lemma (Sinclair 2018), but lemmatization also considers the POS of the word (Bergmanis and Goldwater 2018) (e.g., noun, verb, adjective, or adverb). The `WordNetLemmatizer` package in Python NLTK analyzes the POS tag in a sentence input and lemmatizes each word, using the English lexical database WordNet.[16] The lemmatizer can also be used on isolated words by mapping to a single, most common lemma, or provide different outputs based on the specified POS tag. For instance, the word "leaves" can have the lemma "leave" (verb) or "leaf" (noun). Another example is mapping "better" and "best" to the same lemma "good", which would not have been picked up by a stemmer. Lemmatization is especially useful in parsing morphologically rich languages, whose grammatical relations (e.g., subject, verb, object) are indicated by changes to the words (Tsarfaty *et al*. 2010). Examples of such languages include Turkish, Finnish, Slovene, and Croatian (Gerz *et al*. 2018).

Lemmatization has a higher time complexity than stemming, but given modern big data technology, we do not need to be overly concerned about computational speed. Assume that the text corpus contains N words, and that each sentence has M words on average. Stemming takes $O(N)$ time because the algorithm processes each word exactly once throughout the corpus. In comparison, lemmatization takes $O(MN)$ time because the algorithm reads in the whole sentence to identify the context and POS tag beforehand. A compromise is context-free lemmatization, which produces the appropriate lemmas without using the whole context (Namly *et al*. 2020). By leveraging existing inflection tables, this method has achieved some success in English, Dutch, and German (Nicolai and Kondrak 2016). Nevertheless, the actual computation time is often acceptable for either method. Mubarak (2017) lemmatized 7.4 million Arabic (a morphologically rich language) words on a personal laptop within 2 minutes, indicating that the computational speed is usually fast. Note that English has only eight inflectional morphemes,[17] and many researchers still use stemming for simplicity (Kao and Poteet 2007; Meyer, Hornik, and Feinerer 2008).

3.6.2 Known issues

Neither stemming nor lemmatization is perfect. Both methods suffer from over-consolidation of words—assigning two word forms with clearly different meanings to the same token, such as mapping both "computer" and "computation" to the token "comput" (Jivani 2011). A known way to avoid over-consolidation is to use the lempos format (lemma with POS tag, e.g., "set_NOUN"), as implemented by Akhtar, Sahoo, and Kumar (2017) and Ptaszynski *et al*. (2019). Moreover, some words require context to determine its true base form or lemma. A fundamental question to ask is whether we should perform stemming and/or lemmatization in the first place. The answer would be "yes" in most cases, but depending on the goal of the text data mining project, we may have to reconsider the pros and cons of consolidating word forms before making the decision.

A risk of consolidating word forms is the loss of sentiment information (Haddi, Liu, and Shi 2013; Potts nd). Words of the same root can have different sentiments, but they are mapped to

[15] http://snowball.tartarus.org/algorithms/porter/stemmer.html.
[16] https://wordnet.princeton.edu/.
[17] http://www.arts.uwaterloo.ca/~raha/306a_web/EnglishInflectionalAffixes.pdf.

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

the same token. For example, both "objective" (positive) and "objection" (negative) are mapped to "object" under the Porter stemming algorithm. As a result, we do not know whether the token "object" indicates something positive or negative. Bao *et al.* (2014) also show that both stemming and lemmatization negatively affect Twitter sentiment classification. Even conservative lemmatization can have a negative impact on POS tagging and NER (Cambria *et al.* 2017). Some named entities would not be recognizable after the corpus is lemmatized, such as the American rock band "Guns N' Roses" versus "gun n rose". Therefore, it is better to perform these NLP steps before we lemmatize the corpus.

Another risk is losing the spelling variation as a writing style element, which can be detrimental to authorship attribution (Stamatatos 2008). For instance, spelling discrepancies exist between American English and British English, such as "center" versus "centre" and "canceled" (one "l") versus "cancelled" (two "l"s). Assume we have two documents—one in American English and the other in British English. The two documents should be written by different people, since it is unlikely that someone would use both forms in writing. But after we perform stemming/lemmatization and map each pair to the same token, we cannot distinguish between the two English variations. One potential solution is to identify the words with spelling variations and add an underline character to each word, making the stemmer/lemmatizer "think" they are different tokens. In this way, the algorithm would preserve these words in their original forms.

3.7 N-gramming and identifying multiword expressions

N-gramming is a text preprocessing approach to analyze the phrases consisting of n words in the corpus, that is, n-grams (Dunning 1994). Some n-grams have special meaning (e.g., "New York" and "neighborhood watch"), and they are called multiword expressions (Blei and Lafferty 2009; Masini 2019). The formal definition of multiword expressions is "a class of linguistic forms spanning conventional word boundaries that are both idiosyncratic and pervasive across different languages," as stated by Constant *et al.* (2017). Although some multiword expressions come from n-grams, others are syntactically flexible and may contain different forms that map to the same expression (Sag *et al.* 2002). For instance, the two sentences "Please contact the Customer Help Desk." and "Please contact the Customer Helpdesk." mean the same thing. It is challenging to identify the two multiword expressions "customer help desk" (three words) and "customer helpdesk" (two words), since neither may have enough occurrences in the corpus to be detected. Given the complexity of multiword expressions, Sailer and Markantonatou (2018) regarded them as a difficult problem in NLP applications and computational linguistics. Fortunately, there are programming tools to reduce the burden of n-gramming the corpus. The Python modules include `Phraser` in `gensim` (Řehůřek and Sojka 2010) and `nltk.tokenize.mwe` in NLTK (Bird *et al.* 2009). The R packages include `quanteda` (Benoit *et al.* 2018) and `udpipe` (Wijffels 2021). However, post hoc human judgment is often necessary in identifying the true multiword expressions (Seretan 2011; Poddar 2016).

Retaining multiword expressions is essential because many text mining models rely on the bag-of-words assumption and disregard word order (Wang, McCallum, and Wei 2007), a.k.a. bag-of-words models. Although bag-of-words models are fast and efficient (Joulin *et al.* 2016), their failure to account for word order results in losing semantic meaning (Wallach 2006). An example is "milk chocolate" versus "chocolate milk", where the two phrases consist of the same words but have different meanings. Word order encodes discourse information (Kaiser and Trueswell 2004), especially in free word order languages with significant inflection and discourse functions reflected in the word order (Slioussar 2011). These languages include Finnish, Hindi, Russian, Turkish, and many more (Hoffman 1996). Hence, we need n-gramming to preserve multiword expressions as tokens, so each multiword expression will not be separated in the permutation phase of the model. Das *et al.* (2013) also showed that n-gramming helps in keyword extraction

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

because the approach can include named entities, many of which are multiword expressions such as WHO (World Health Organization).

Multiword expressions are important for a wide variety of NLP tasks, including topic identification (Koulali and Meziane 2011) and machine translation (Lambert and Banchs 2006; Tan and Pal 2014). Detecting multiword expressions is also beneficial to word sense disambiguation, that is, determining which meaning of the word is used in a particular context (Masini 2005; Finlayson and Kulkarni 2011). N-gramming detects and retains many multiword expressions (e.g., "White House" and "New Jersey") before we feed the data into a statistical model, so these multiword expressions will not be separated in the analysis (Bekkerman and Allan 2004). This is a useful approach with demonstrated success in many text mining applications (Doraisamy and Rüger 2003; Los and Lubbers 2019). With the increasing popularity of word embeddings, more advanced methods like deep learning and L1 regularization are available to identify multiword expressions (Berend 2018; Ashok, Elmasri, and Natarajan 2019). We emphasize that n-gramming is not a panacea for retaining multiword expressions or semantic information. Any text preprocessing technique will result in some loss of semantic information from the corpus (Fesseha *et al.* 2021), and in an extreme case, n-gramming even decreased cross-lingual model performance (Kamps, Adafre, and De Rijke 2004).

Before getting into the construction of n-grams and detection of multiword expressions, we also add that n-grams play a major role in creating probabilistic language models (Bengio *et al.* 2003; Qudar and Mago 2020), such as the conditional probability of words occurring together. N-grams can represent linguistic patterns and indicate the perplexity of the language model (Toral *et al.* 2015). Perplexity is defined as the inverse probability of the observed test data (Gamallo, Campos, and Alegria 2017), or the effective number of equally likely words identified by the language model (Lafferty and Blei 2006). The perplexity measure is especially useful in comparing the complexity of multilingual language models (Buck, Heafield, and Van Ooyen 2014; Zeng *et al.* 2017. Note that our focus is on the n-gramming process itself, rather than the maintenance of word order. Recovering Chomsky's Deep Structure across languages is a completely different research field in linguistics (Chomsky 1969; Lan 1996).

3.7.1 Constructing N-grams from a text document

We create n-grams as phrases of n consecutive words in a text document (Gries and Mukherjee 2010), using a moving window of length n across the text, that is, "shingling" (Broder *et al.* 1997; Huston, Culpepper, and Croft 2014). We use the definition of word-based n-grams, which is different than the character-based n-gram definition, which refers to a string of n characters (Houvardas and Stamatatos 2006). Words are also called unigrams, while bigrams and trigrams denote the two-word and three-word phrases in the corpus, respectively.

A document consisting of n words would have n unigrams, $n - 1$ bigrams, and $n - 2$ trigrams. For example, the text "one should be humble while studying advanced calculus" contains eight unigrams, seven bigrams, and six trigrams as below:

- Unigrams: "one", "should", "be", "humble", "while", "studying", "advanced", "calculus"
- Bigrams: "one should", "should be", "be humble", "humble while", "while studying", "studying advanced", "advanced calculus"
- Trigrams: "one should be", "should be humble", "be humble while", "humble while studying", "while studying advanced", "studying advanced calculus"

The ordering of text preprocessing modules is key to success. Converting the string of text into unigrams is equivalent to tokenization (see Section 3.2), and this is a prerequisite for removing stopwords and n-gramming (Putra, Gunawan, and Suryatno 2018; Chaudhary and Kshirsagar 2018). Note that this example inputs a full sentence in raw text, without stopword removal or

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

stemming/lemmatization. As a result, the unigrams contain general stopwords such as "one" and "be", which are not ideal. The unigram "studying" is not reverted to its base form "study", either. That's why removing stopwords and stemming/lemmatization are often performed before n-gramming (Anandarajan *et al*. 2019; Arief and Deris 2021), as illustrated in Figure 1.

In terms of semantic meaning, only the phrase "advanced_calculus" seems special enough to be a multiword expression because it is the name of a college-level class. We need systematic methods to determine which n-grams are multiword expressions and which are not (Baldwin and Kim 2010). Moreover, n-gramming has limitations because some multiword expressions are not *n* consecutive words (i.e., n-grams). They may skip a few words in between and hence are called skip-grams (Guthrie *et al*. 2006). This reflects the challenge of modeling long-distance dependencies in natural languages (Deoras, Mikolov, and Church 2011; Merlo 2019).

In general, multiword expressions should be both practically and statistically significant. Practical significance means that the phrase is of practical interest, that is, it occurs many times in the corpus. Rare phrases should be removed from the n-gram pool (Grobelnik and Mladenic 2004). On the other hand, statistical significance means that the words forming the multiword expression have a relatively high probability to stay together (Brown *et al*. 1992). Thus, a multiword expression should be an n-gram of statistical interest, rather than *n* words that just happen to appear together many times. Sections 3.7.2 and 3.7.3 will discuss the practical and statistical significance requirements for multiword expressions. This is by no means the only way to determine which n-grams are multiword expressions; other methods include finding the longest common subsequence (Duan *et al*. 2006) and using linguistic features of each sentence (Boukobza and Rappoport 2009).

3.7.2 Multiword expressions: Practical significance

For practical significance, the easiest way is to set a minimum frequency and consider any n-gram to be a multiword expression if its number of occurrences is higher than the cutoff (Wei *et al*. 2009). Since general stopwords have been removed from the corpus, uninformative bigrams (e.g., "you are", "is an") are less likely to appear. The article "a", "an", or "the" often precedes a noun, and such bigrams (e.g., "the artist") are not helpful, either. Setting a cutoff frequency ensures that the multiword expressions are of practical interest, and this approach has proven success in computational linguistics (Pantel and Pennacchiotti 2006; Lahiri and Mihalcea 2013). As a note of caution, Wermter and Hahn (2005) showed that a high number of occurrences does not always mean the n-gram is a term, that is, a multiword expression with special meaning. An example is in the biomedical field, the phrase "t cell response" has a higher frequency than "long terminal repeat", but the latter is a term while the former is not.

The cutoff frequency can be determined empirically, and Microsoft Azure Machine Learning Studio recommends a starting value of 5 as the minimum frequency for multiword expressions (Microsoft 2019). Previous researchers had success in using the cutoff frequency of 5, while they analyzed a corpus of 450 political blog posts, with around 289,000 total words at the time when the corpus was ready for n-gramming (Soriano *et al*. 2013; Au 2014). Inevitably, a low threshold of minimum frequency results in a steep increase in computation time as the corpus gets larger due to the need of processing the massive number of low-frequency n-grams (Brants *et al*. 2007). The New York Times Annotated Corpus (Sandhaus 2008) contains more than 1.8 million newspaper articles and more than 1.05 billion words, so Berberich and Bedathur (2013) set the cutoff frequency to 100. The ClueWeb09 dataset (Callan *et al*. 2009) is even larger, with more than 50 million web documents and more than 21.4 billion English words. Accordingly, Berberich and Bedathur (2013) increased the cutoff frequency to 1000. The Google Syntactic Ngrams English corpus (Goldberg and Orwant 2013) consists of more than 345 billion words, and Ng, Bansal, and Curran (2015) set the cutoff frequency to as high as 10,000.

The histograms of n-grams for each *n* are right-skewed (Ha *et al*. 2009; Bardoel 2012). The distribution generally follows the power law (Jean-Baptiste 1916; Newman 2005), just like the

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

Table 2. The n-gram statistics of *Wuthering Heights*

	Once	Twice	3 times	4 times	5 or more	Total distinct
Unigram	3983 (43.27%)	1500 (16.29%)	853 (9.26%)	511 (5.55%)	2360 (25.63%)	9207 (100.00%)
Bigram	48,617 (77.93%)	6765 (10.84%)	2451 (3.93%)	1283 (2.06%)	3271 (5.24%)	62,387 (100.00%)
Trigram	100,158 (93.35%)	4801 (4.47%)	1173 (1.09%)	454 (0.42%)	716 (0.67%)	107,302 (100.00%)
4-gram	117,457 (98.79%)	1191 (1.00%)	152 (0.13%)	50 (0.04%)	45 (0.04%)	118,895 (100.00%)

illustration of Zipf's law (Zipf 1949) in Figure 5. The vast majority of bigrams appear only once, and the phenomenon of trigrams is more extreme. Dressel (2016) demonstrated that only 19 percent of the trigrams occurred more than once in their corpus before stopwords were removed. We also n-grammed the publicly available e-book *Wuthering Heights* from Project Gutenberg (Robinson 2018), which contains 120,772 total words and 9207 total distinct words in the corpus. Table 2 provides the n-gram statistics of *Wuthering Heights*: over a quarter of unigrams (words) appear 5 or more times in the corpus, so it is viable to further examine the frequency trends of the unigrams. For bigrams, more than three-quarters of them appear only once, leaving us with about a quarter of the unique bigrams to analyze. Finally, more than 90 percent of the trigrams and 4-grams appear only once in the corpus, showing that only a small percentage of longer n-grams may be of interest.

Remark. N-grams should not be calculated over sentence boundaries (Buerki 2017), but Huston, Moffat, and Croft (2011) showed that using sentence boundaries as separation can greatly reduce the number of 4-grams produced. Since the majority of longer n-grams will be dropped by the thresholding, empirically it does not matter whether sentence boundaries are considered in the n-gramming process.

3.7.3 Multiword expressions: Statistical significance

For statistical significance, we focus on conditional probability (Jurafsky and Martin 2008)—given one or more words, the probability of another word immediately following the sequence. In mathematical notation, a string of n words is written as $w_1 w_2 \cdots w_n$. For the bigram $w_1 w_2$, the conditional probability of w_2 following w_1 is

$$P(w_2|w_1) = \frac{P(w_1 w_2)}{P(w_1)} = \frac{\text{Count}(w_1 w_2)}{\text{Count}(w_1)}, \tag{1}$$

where the function Count(s) outputs the frequency of the word string s in the corpus.

Assuming the corpus contains N words, we have the probability of getting each word as $P(w) = \text{Count}(w)/N$, and the probability of getting each bigram as $P(wv) = \text{Count}(wv)/(N-1)$. The only difference is that the corpus contains $N-1$ bigrams. When N is sufficiently large, N and $N-1$ are asymptotically equal, so we do not make a distinction in calculating $P(w_2|w_1)$.

To determine whether the bigram $w_1 w_2$ is a multiword expression, we test against the two competing hypotheses:

- H_0: $P(w_2|w_1) \leq P(w_2)$
- H_1: $P(w_2|w_1) > P(w_2)$

The framework is adopted from Chai (2022). This is a one-sided hypothesis test because only a large $P(w_2|w_1)$ is of interest. When the p-value is less than 0.05, we reject H_0 and conclude that $w_1 w_2$ is a multiword expression.

We can extend the bigram testing scheme to n-grams, but the conditional probability of seeing the nth word given the first $n-1$ words, $P(w_n|w_1 \cdots w_{n-1})$, is unreasonable to compute. Chances

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

are high that the $(n-1)$ word phrase appears only once in the corpus, making this conditional probability 100 percent, which is not what we are looking for. Instead, we can approximate the conditional probability with

$$P(w_n|w_1 \cdots w_{n-1}) \approx P(w_n|w_{n-1}) = \frac{\text{Count}(w_{n-1}w_n)}{\text{Count}(w_{n-1})}, \tag{2}$$

where the Markov assumption states that w_n depends on only the previous word w_{n-1} (Fosler-Lussier 1998; Tran and Sharma 2005). Then, the n-gramming question is reduced to whether $w_{n-1}w_n$ is a multiword expression, and the hypothesis testing framework still applies.

Many automated tools are available to perform n-gramming, but most of them simply pull all n-grams from the input text. For example, the function `tokens_ngrams` in the `R` package `quanteda` (Benoit *et al.* 2018) generates the n-grams, but this package does not provide support for how to test n-grams for multiword expressions. It is still up to the researcher to determine which ones are multiword expressions of length n, that is, the n-grams of interest. Fortunately, the implementation of Equation (1) is not difficult for bigrams. The n-gramming results can also be affected by the choices within other text preprocessing operations, such as tokenization, letter case normalization, and stopword removal (Al-Molegi *et al.* 2015; Jimenez *et al.* 2018).

A manual way to create multiword expressions of length 3 or 4 is to run the bigram generation and testing algorithm twice (Henry 2016). If the concatenation of a word and a bigram passes the test, the trigram becomes a multiword expression of length 3. If the concatenation of two bigrams passes the test, the 4-gram becomes a multiword expression of length 4. One drawback is that $w_1w_2w_3$ being a multiword expression does not always imply w_1w_2 and/or w_2w_3 are also multiword expressions. But given the Markov assumption in Equation (2), we can safely presume that counterexamples are rare, that is, w_3 mostly depends on only the previous word w_2.

For an n-gram to be a multiword expression, both practical and statistical significance are essential requirements. Practical significance requires each multiword expression to appear a minimum number of times in the corpus, so the selected n-grams would be presumed to have semantic importance. Statistical significance ensures that the multiword expressions are unlikely to be words co-occurring by chance (Dror *et al.* 2018), but the conditional probability $P(w_2|w_1)$ would not be accurate without w_1w_2 of sufficient frequency in the corpus, that is, practical significance.

3.7.4 N-gramming: Computational issues

Although the number of total n-grams grows linearly with the total number of words in the document, detecting multiword expressions in the corpus is computationally expensive due to the large number of distinct n-grams (McNamee and Mayfield 2007; Vilares *et al.* 2016). If the vocabulary contains V distinct words, the number of distinct unigrams of the corpus has $O(V)$ complexity. As the length n increases, the size of the n-gram pool grows exponentially—the complexity increases to $O(V^2)$ for distinct bigrams and to $O(V^3)$ for distinct trigrams. Nevertheless, the effort needed may not generate a good return of investment, especially for long n-grams. For aggregated statistics of text data, the n-gram frequency is right-skewed and most of the detected n-grams appear only once in the corpus. Accordingly, most studies do not investigate the corpus past 4-grams (Lin and Hovy 2003; Barrón-Cedeño and Rosso 2009), not to mention longer n-grams. As a result, long multiword expressions are difficult to detect in n-gramming (Raff and Nicholas 2018; Raff *et al.* 2019).

The good news is that long multiword expressions may be discovered via researchers' understanding of the data content. The presence of such phrases may follow patterns and/or contain non-ignorable information, and we can apply subject matter knowledge of the corpus to create a list of long n-grams which are likely to be multiword expressions. We should retrieve these phrases directly from the data, rather than shingling the entire corpus to generate all n-grams when n is large. After obtaining the frequency of such phrases, we can determine whether it is appropriate

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

to remove them from the corpus. If the answer is yes, then the text corpus can be further reduced. Below are some scenarios that produce long multiword expressions. Upon detection, these phrases can be easily removed without losing excessive semantic information.

- The sales limitation phrase "must be 18 or older" is a multiword expression in a text corpus of digital advertisements. This phrase indicates the product's legal age range, so the product should not be marketed to underage people such as middle school students. But from a computational advertising standpoint, this multiword expression does not provide any information about the product itself, making it difficult to match ads to users (Soriano *et al*. 2013).
- In a political blog corpus about the 2012 Trayvon Martin shooting incident,[18] the research group at Duke Statistical Science discovered an extremely long multiword expression—because one blogger included the Second Amendment to the United States Constitution in every post as a signature. The statement is "A well regulated Militia, being necessary to the security of a free State, the right of the people to keep and bear Arms, shall not be infringed." Although controversial, this shows the blogger's perspective toward gun laws (Chai 2017).

The decision to apply n-gramming for text preprocessing or not depends on the NLP application. N-gramming is beneficial to applications that focus on words and phrases as consecutive words, such as text classification (Beitzel *et al*. 2007; Narala, Rani, and Ramakrishna 2017) and probabilistic topic modeling (Acree 2016; Luiz *et al*. 2019). N-gramming also identifies multiword expressions, which can enhance the vocabulary database in word embeddings (Goodman, Zimmerman, and Hudson 2020). In machine translation, n-gramming can be used for the hard-to-translate parts, because translation is not a one-to-one mapping between words (Du, Yu, and Zong 2018). But since n-gramming does not provide context to phrases, this approach will be of limited help in logical reasoning (Bernardy and Chatzikyriakidis 2019; Zhou *et al*. 2020). Instead, special techniques are needed for multiword expressions (Rikters and Bojar 2017) and also for proper names (e.g., transliteration from Latin to Cyrillic characters) (Petic and G fu 2014; Mansurov and Mansurov 2021).

4. Dataset examples

In this section, we provide three examples of text corpora which require more advanced preprocessing methods, that is, methods catering to the nature of the text data. The first example is a technical dataset, where removing stopwords should be done after stemming and n-gramming to preserve the technical phrases. The second example is social media data, where removing formatting is of heavy focus and special handling is needed in text normalization. The third example is survey text with numerical ratings, and we should retain negation in every step of text preprocessing, especially during stopword removal. All seven text preprocessing modules in Figure 1 apply, but the discussion will be on the specific adaptation. This is by no means a comprehensive list, but the methods will provide a foundation for unconventional text corpora beyond the three categories described here.

4.1 Technical datasets

For datasets with technical content, we recommend using publicly available e-books on Project Gutenberg as a starting filter to remove stopwords, especially the e-books with general content. Leveraging Project Gutenberg eliminates the subjectivity of deciding which words should be

[18] https://en.wikipedia.org/wiki/Shooting_of_Trayvon_Martin.

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

included in the stopword list, increasing the reproducibility of the work. Since many researchers examine the predefined stopword list and manually add or remove a few words beforehand, it is more difficult to reproduce their results unless they publish the exact stopword list they used in the project (Nothman *et al.* 2018). Existing literature has proposed automated methods to extract domain-specific stopwords with improved text classification results (Ayral and Yavuz 2011; Makrehchi and Kamel 2017). But given the amount of technical expertise and work required, the automation may not be worthwhile, or even feasible (Arora *et al.* 2016). Using an e-book from Project Gutenberg to remove stopwords is an easier way to achieve potentially better results than using a standard stopword list.

An example of a technical corpus is the JSM (Joint Statistical Meetings) abstract collection (Bi 2016), which contains more than 3000 abstracts and 700 sessions from the 2015 JSM.[19] JSM is one of the world's largest conferences in statistics, and the abstracts obviously contain much technical jargon. Common phrases in this dataset include "maximum likelihood estimation" and "time series". Bi (2016) eliminated all HTML formatting and punctuation from the corpus, tokenized the corpus into English words, and converted all letters to lowercase. The stopword removal process for the JSM abstract collection contains three steps:

(1) Stem and n-gram the JSM abstract collection.
(2) Stem and n-gram a publicly available e-book from Project Gutenberg.
(3) Remove anything from (1) that exists in (2).

In Step 1, note that n-gramming is required before we compare the two datasets and remove the words also in Project Gutenberg. Some statistical terminology contains words that may be considered a stopword, so we need to preserve these n-grams first. For example, after the stopword "one" is removed, the term "one sample t-test" is not equivalent to a "sample t-test". Assume the word "two" precedes both terms, then "two one sample t-tests" is different from a "two sample t-test". The former means conducting an independent one-sample t-test on each of the two samples, and the latter means conducting a single t-test to compare the two given samples.

In Step 2, appropriate choices of a filtering book should be something completely unrelated to statistics, such as *Little Busybodies: The Life of Crickets, Ants, Bees, Beetles, and Other Busybodies* by Jeanette Augustus Marks and Julia Moodyand. If the technical dataset includes medical content, such as citation networks from PubMed,[20] then using an e-book about insects may not be appropriate. Otherwise, content related to insect bites would be filtered out by the e-book. A better choice is *Herein is Love* by Reuel L. Howe, which is an interpretation about love from the Bible.

In Step 3, the intersection with the filtering book is removed from the corpus of JSM abstracts. This step removes most of the general content and keeps the technical terms in the corpus. This not only reduces the vocabulary size but also prepares the corpus for topic modeling and conference scheduling optimization. Although most JSM abstracts contain self-identified keywords, an automated system is still needed for conference session scheduling given the large number of abstracts (Sweeney 2020; Frigau, Wu, and Banks 2021). Most conferences have moved to virtual in COVID times, but Patro *et al.* (2020) emphasized that scheduling remains a challenge due to time zones, sign language interpreter hours, etc. Availability of on-demand recordings may change the equation, because participants have the opportunity to watch the recorded talks after the conference.

4.2 Social media data

Social media data contain information from user posts, and text analytics of such data can generate insights of business value and/or research interest (Mostafa 2013). Particularly, Twitter has an

[19] https://ww2.amstat.org/meetings/jsm/2015/.
[20] https://www.ncbi.nlm.nih.gov/pubmed/.

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

abundance of unstructured text data (Kwak *et al*. 2010), and text mining applications include (and are not limited to) opinion mining (Pak and Paroubek 2010), stock market prediction (Bollen, Mao, and Zeng 2011), and even how people communicate under emergency situations (Mendoza, Poblete, and Castillo 2010; Vieweg *et al*. 2010). In addition to the text preprocessing methodology described in Section 3, preprocessing social media text comes with unique challenges due to ungrammatical language, misspellings, and typos (Batrinca and Treleaven 2015). Many resources are available for social media text preprocessing, such as the *Google Refine*[21] desktop application software and the book *Python Social Media Analytics* (Chatterjee and Krystyanczuk 2017). These are excellent and helpful tools, and they describe the whole text preprocessing pipeline.

Therefore, we point out some key differences between preprocessing a traditional text dataset and preprocessing a social media text corpus. Although preprocessing a social media text corpus takes considerable time, the preprocessing step is worth the effort because the resulting corpus leads to successful implementation of text mining algorithms (Irfan *et al*. 2015).

First, a text corpus from social media often contains HTML tags and URL paths which were generated from web scraping. We can use regular expressions to remove the web-generated characters, just like the process of removing punctuation described in Section 3.4.2. One caveat is to consider whether to remove punctuation or remove web-generated characters first, because the order of removing different types of nonword characters matters in the code development. For instance, many URLs start with `https://`, and if we remove the colon (:) first, then the URL prefix would be `https//` instead.

Next, hashtags and mentions in Twitter are extremely helpful in processing tweets, but due to their distinct structures, it is important to identify them as separate entities before we remove all punctuation. Hashtags start with the # sign (e.g., #hashtag), and they often indicate the main terms of the tweet, which are useful in topic identification (Zubiaga *et al*. 2015) and event detection (Wang *et al*. 2014b; Cai *et al*. 2015). Mentions start with the @ symbol (e.g., @username) and refer directly to another Twitter user, making them helpful in NER (Yamada *et al*. 2015; Gorrell *et al*. 2015). All mentions (@) can also be converted into stopwords (Metzler *et al*. 2021) or masked for certain tasks (Abd-Alrazaq *et al*. 2020). If we remove punctuation before processing the hashtags and mentions, the unexpected word in a sentence can mess up NLP models such as POS tagging (Kaufmann and Kalita 2010).

Moreover, text from online social media often contains abbreviations not in a standard English dictionary, such as "cu" (see you) and "idk" (I don't know) (Pennell and Liu 2011; Kozakou 2017). These terms, also known as Internet slang, need to be explicitly defined. Nevertheless, this task is relatively difficult because we require an unconventional dictionary to translate slang words into their proper reference words, such as the *Internet Slang Dictionary* (Jones 2006). But given the fast-growing content on the web, the static dictionaries about Internet slang risk being outdated due to not keeping up with the current trends. It is possible to automatically detect such spelling variants (Barteld 2017), but this is an extensive research topic outside the text preprocessing scope.

After the preprocessing of social media data is complete, we recommend text mining methods which specialize in short texts. Many topic modeling techniques, especially the ones based on the standard latent Dirichlet allocation (LDA), work better for longer documents and have decreased performance over short texts (Li *et al*. 2016). This is due to their probabilistic nature of identifying the topic distribution over words in each document (Chen, Yao, and Yang 2016), which becomes a challenge for shorter documents like social media data (Chen *et al*. 2019). One potential solution is auxiliary word embeddings (Li *et al*. 2017), that is, incorporating external words to the sample space of the dataset. In this way, each word in the short text can be either generated directly from the Dirichlet multinomial distribution or sampled from the estimated probabilities by the word embeddings in the corresponding topic (Nguyen *et al*. 2015). Despite the challenges of preprocessing and analyzing social media data, many researchers have successfully generated insights

[21] https://github.com/OpenRefine/OpenRefine/wiki.

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

from the massive database of information. Sophisticated models can provide recommendations for companies to develop their social media marketing strategy (He, Zha, and Li 2013), and even a simple gender classification of Facebook names can improve an online dating platform's accuracy of targeted advertising (Tang *et al*. 2011).

4.3 Text with numerical ratings

Some datasets contain text records with a numerical rating assigned to each text comment, such as surveys, online reviews, and performance evaluations. The numerical ratings are regarded as metadata, and they should also be incorporated in the data preprocessing phase. There are other types of metadata we can utilize, such as dates, authors, and locations. Nevertheless, we decided to emphasize the numerical ratings in surveys because analysis of survey text is relatively rare, due to its complexity compared with the numerical counterparts (Schuman and Presser 1996; Roberts *et al*. 2014). When preprocessing the text with numerical ratings, we need to ensure correctness of which rating is associated with which text comment and beware of missing cells which can mess up the alignment. But more often than not, the rating errors are made by the respondents (Saal, Downey, and Lahey 1980). Therefore, it is important to identify potential data errors and make correction efforts, so combined analysis of text and numerical data is required.

The problem of text-rating mismatch is especially prevalent in surveys because respondents may get confused with the rating scale (Fisher 2019). A 1–10 rating scale has at least two different meanings: 1 (least important) to 10 (most important) and 1 (most important) to 10 (least important). Appropriate survey design and instructions can mitigate the problem (Fowler 1995; Friedman and Amoo 1999), but text-rating inconsistencies are inevitable in a survey response dataset (Fisher 2013). For example, in an employee satisfaction dataset with the rating scale from 1 (least satisfied) to 10 (most satisfied), a respondent wrote "Love my work – very varied." and gave a "1" rating (Chai 2019). The respondent was obviously confused by the rating scale because the comment shows satisfaction with his/her work.

Preserving negation is essential to the survey text, because removing the negation term will result in an opposite meaning and ambiguity of text-rating assignment. If another respondent wrote "I don't like my job," then the "3" rating is appropriate. But if we accidentally remove "don't" as a stopword, the text comment becomes "I like my job," and the originally correct rating "3" becomes inappropriate. Section 3.5 explains that many existing stopword lists contain negation terms like "no" and "not". Hence, researchers should exclude them from the list so that negation can be retained in the corpus.

A potential solution to validate a text-rating dataset is to estimate the rating from the text using the supervised latent Dirichlet allocation (sLDA) (McAuliffe and Blei 2008), which is a supervised topic model for text mining. Compared with the LDA (Blei *et al*. 2003) for topic modeling, sLDA is supervised because it takes the record labels (i.e., numerical ratings) into account. After we train the sLDA model using the text-rating dataset, we can estimate the rating from a particular text record and compare it with the actual rating (Fisher and Lee 2011). If the estimated rating is very different from the actual rating in the record, then the rating is likely a response error. We should flag the record and reconsider its accuracy.

5. Conclusion

Researchers and practitioners need to know which text preprocessing modules to apply in what order, and whether a generic or specific model is preferable. General text preprocessing methods apply for most datasets, but more content-specific methods are needed to preserve additional semantic information. There is a trade-off between the effort in data preprocessing and the resulting data quality, so advanced text preprocessing methods are considered add-ons to the general

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

methods. Which text preprocessing methods are useful depend on both the corpus and the goal of text mining (White *et al*. 2018). Therefore, we provide some guidelines to select the appropriate text preprocessing methods for a given dataset. For publicly available datasets such as 20 Newsgroups and WebKB (World Wide Knowledge Base),[22] previous researchers have done the preprocessing and published their methodology (Miah 2009; Albishre, Albathan, and Li 2015). But when we receive a whole-new text corpus for a specific application, we have to preprocess the data with little guidance. This is the key application for the compare-and-contrast of various text preprocessing methods.

This article has set up a framework in selecting the appropriate text preprocessing methods for a given corpus. Most of the literature we reviewed here used text corpora written in English, but some concepts are applicable to other languages. For instance, most languages contain stopwords, and their word frequency distributions follow the Zipf's law (Zipf 1949). Nevertheless, the complexity of each necessary text preprocessing step would vary by language. CJK (Chinese, Japanese, and Korean) require more complex tokenizers due to the character-based nature of the languages (Zhang and LeCun 2017), as well as the lack of white space as obvious word boundaries (Moh and Zhang 2012). On the other hand, morphologically rich languages like Turkish and Finnish require more complex stemmers/lemmatizers due to their wide range of derivational and inflectional word variation (Przepiórkowski *et al*. 2012; Nuzumlalı and Özgür 2014).

Acknowledgments. The author is grateful for the support of Microsoft in writing this manuscript. The author would like to thank Prof. David Banks, her PhD advisor at Duke University, who motivated her to do research in text mining. The author would also like to thank the anonymous reviewers for the detailed comments, which greatly improved the manuscript.

Conflicts of interest. The author is employed at Microsoft Corporation, and she completed a PhD in statistical science from Duke University.

References

Abd-Alrazaq A., **Alhuwail D.**, **Househ M.**, **Hamdi M.** and **Shah Z.** (2020). Top concerns of Tweeters during the COVID-19 pandemic: Infoveillance study. *Journal of Medical Internet Research (JMIR)* **22**(4), e19016.

Abdelali A., **Darwish K.**, **Durrani N.** and **Mubarak H.** (2016). Farasa: A fast and furious segmenter for Arabic. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pp. 11–16.

Abdolahi M. and **Zahedh M.** (2017). Sentence matrix normalization using most likely n-grams vector. In *2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI)*. IEEE, pp. 0040–0045.

Abraham A., **Dutta P.**, **Mandal J.K.**, **Bhattacharya A.** and **Dutta S.** (2018). *Emerging Technologies in Data Mining and Information Security*: *Proceedings of IEMIS 2018, Volume* 2, vol. 813. Springer. IEMIS Stands for International Conference on Emerging Technologies in Data Mining and Information Security.

Acree B.D. (2016). *Deep Learning and Ideological Rhetoric*. PhD Dissertation, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA.

Ács J., **Kádár Á.** and **Kornai A.** (2021). Subword pooling makes a difference. arXiv preprint arXiv:2102.10864.

Adnan K. and **Akbar R.** (2019a). An analytical study of information extraction from unstructured and multidimensional big data. *Journal of Big Data* **6**(1), 1–38.

Adnan K. and **Akbar R.** (2019b). Limitations of information extraction methods and techniques for heterogeneous unstructured big data. *International Journal of Engineering Business Management* **11**, 1–23.

Aggarwal C.C. and **Zhai C.** (2012). *Mining Text Data*. Boston, MA, USA: Springer Science & Business Media.

Agić Ž., **Merkler D.** and **Berović D.** (2013). Parsing Croatian and Serbian by using Croatian dependency treebanks. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pp. 22–33.

Agre G., **van Genabith J.** and **Declerck T.** (2018). *Artificial Intelligence: Methodology, Systems, and Applications: 18th International Conference, AIMSA 2018, Varna, Bulgaria, 12–14 September 2018, Proceedings*, vol. 11089. Springer.

Akhtar A.K., **Sahoo G.** and **Kumar M.** (2017). Digital corpus of Santali language. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, pp. 934–938.

[22] http://www.cs.cmu.edu/~webkb/.

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

Akkasi A., **Varoğlu E.** and **Dimililer N.** (2016). ChemTok: A new rule based tokenizer for chemical named entity recognition. *BioMed Research International* **2016**, 1–9.

Al-Khafaji H.K. and **Habeeb A.T.** (2017). Efficient algorithms for preprocessing and stemming of tweets in a sentiment analysis system. *International Organization of Scientific Research – Journal of Computer Engineering* **9**(3), 44–50.

Al-Molegi A., **Alsmadi I.**, **Najadat H.** and **Albashiri H.** (2015). Automatic learning of Arabic text categorization. *International Journal of Digital Contents and Applications* **2**(1), 1–16.

Al Sharou K., **Li Z.** and **Specia L.** (2021). Towards a better understanding of noise in natural language processing. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP)*, pp. 53–62.

Albalawi R., **Yeap T.H.** and **Benyoucef M.** (2020). Using topic modeling methods for short-text data: A comparative analysis. *Frontiers in Artificial Intelligence* **3**, 42.

Albishre K., **Albathan M.** and **Li Y.** (2015). Effective 20 newsgroups dataset cleaning. In *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, vol. 3. IEEE, pp. 98–101.

Aliwy A.H. (2012). Tokenization as preprocessing for Arabic tagging system. *International Journal of Information and Education Technology* **2**(4), 348–353.

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.

Alonso M.A., **Gómez-Rodríguez C.** and **Vilares J.** (2021). On the use of parsing for named entity recognition. *Applied Sciences* **11**(3), 1090.

Amarasinghe K., **Manic M.** and **Hruska R.** (2015). Optimal stop word selection for text mining in critical infrastructure domain. In 2015 *Resilience Week (RWS)*. IEEE, pp. 1–6.

Anandarajan M., **Hill C.** and **Nolan T.** (2019). Text preprocessing. In *Practical Text Analytics*. Springer, pp. 45–59.

Anderwald L. (2003). *Negation in Non-Standard British English: Gaps, Regularizations and Asymmetries*. London, UK: Routledge.

Angiani G., **Ferrari L.**, **Fontanini T.**, **Fornacciari P.**, **Iotti E.**, **Magliani F.** and **Manicardi S.** (2016). A comparison between preprocessing techniques for sentiment analysis in Twitter. In *Proceedings of the 2nd International Workshop on Knowledge Discovery on the WEB (KDWeb)*.

Arens R. (2004). A preliminary look into the use of named entity information for bioscience text tokenization. In *Proceedings of the Student Research Workshop at HLT-NAACL 2004*, pp. 37–42. HLT-NAACL stands for Human Language Technologies – North American Chapter of the Association for Computational Linguistics.

Arief M. and **Deris M.B.M.** (2021). Text preprocessing impact for sentiment classification in product review. In *2021 Sixth International Conference on Informatics and Computing (ICIC)*. IEEE, pp. 1–7.

Armano G., **Fanni F.** and **Giuliani A.** (2015). Stopwords identification by means of characteristic and discriminant analysis. In *ICAART (2)*, pp. 353–360. ICCART stands for International Conference on Agents and Artificial Intelligence.

Armengol Estapé J. (2021). *A Pipeline for Large Raw Text Preprocessing and Model Training of Language Models at Scale*. Master's Thesis, Universitat Politècnica de Catalunya, Barcelona, Spain.

Arora C., **Sabetzadeh M.**, **Briand L.** and **Zimmer F.** (2016). Extracting domain models from natural-language requirements: Approach and industrial evaluation. In *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*, pp. 250–260.

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*. IEEE, pp. 192–196.

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*. Springer, pp. 295–302.

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

Atwood J. (2008). Stack Overflow Podcast Episode 32. Available from: https://stackoverflow.blog/2008/12/04/podcast-32/.

Au T.C. (2014). *Topics in Computational Advertising*. PhD Dissertation, Duke University, Durham NC, USA.

Aye T.T. (2011). Web log cleaning for mining of web usage patterns. In *2011 3rd International Conference on Computer Research and Development*, vol. 2. IEEE, pp. 490–494.

Ayral H. and **Yavuz S.** (2011). An automated domain specific stop word generation method for natural language text classification. In *2011 International Symposium on Innovations in Intelligent Systems and Applications*. IEEE, pp. 500–503.

Babar S. and **Patil P.D.** (2015). Improving performance of text summarization. *Procedia Computer Science* **46**, 354–363.

Baerman M. (2015). *The Oxford Handbook of Inflection*. Oxford, UK: Oxford University Press.

Baldwin R.S. and **Coady J.M.** (1978). Psycholinguistic approaches to a theory of punctuation. *Journal of Reading Behavior* **10**(4), 363–375.

Baldwin T. and **Kim S.N.** (2010). Multiword expressions. *Handbook of Natural Language Processing* **2**, 267–292.

Baldwin T. and **Li Y.** (2015). An in-depth analysis of the effect of text normalization in social media. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 420–429.

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*. Springer, pp. 615–624.

Bardoel T. (2012). Comparing n-gram frequency distributions. Technical report, Tilburg center for Cognition and Communication (TiCC), Tilburg University, Tilburg, Netherlands.

Barr C., **Jones R.** and **Regelson M.** (2008). The linguistic structure of English web-search queries. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pp. 1021–1030.

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.

Barrón-Cedeño A. and **Rosso P.** (2009). On automatic plagiarism detection based on n-grams comparison. In *European Conference on Information Retrieval*. Springer, pp. 696–700.

Barteld F. (2017). Detecting spelling variants in non-standard texts. In *Proceedings of the Student Research Workshop at the Fifteenth Conference of the European Chapter of the Association for Computational Linguistics*, pp. 11–22.

Basta C., **Costa-jussà M.R.** and **Casas N.** (2021). Extensive study on the underlying gender bias in contextualized word embeddings. *Neural Computing and Applications* **33**(8), 3371–3384.

Batrinca B. and **Treleaven P.C.** (2015). Social media analytics: A survey of techniques, tools and platforms. *AI & Society* **30**(1), 89–116.

Battenberg E. (2012). Ratings prediction using linear regression on text reviews. Technical report, Berkeley Institute of Design (BID), University of California – Berkeley, Berkeley CA, United States.

Beaufays F. and **Strope B.** (2013). Language model capitalization. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, pp. 6749–6752.

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

Beitzel S.M., **Jensen E.C.**, **Lewis D.D.**, **Chowdhury A.** and **Frieder O.** (2007). Automatic classification of web queries using very large unlabeled query logs. *ACM Transactions on Information Systems (TOIS)* **25**(2), 1–29.

Bekkerman R. and **Allan J.** (2004). Using bigrams in text categorization. Technical report, IR-408, Center of Intelligent Information Retrieval, University of Massachusetts at Amherst, Amherst MA, United States.

Bender E.M. (2011). On achieving and evaluating language-independence in NLP. *Linguistic Issues in Language Technology* **6**(3), 1–26.

Bender E.M. and **Friedman B.** (2018). Data statements for natural language processing: Toward mitigating system bias and enabling better science. *Transactions of the Association for Computational Linguistics* **6**, 587–604.

Bengfort B., **Bilbro R.** and **Ojeda T.** (2018). *Applied Text Analysis with Python: Enabling Language-Aaware Data Products with Machine Learning*. Sebastopol, CA, USA: O'Reilly Media Inc.

Bengio Y., **Ducharme R.**, **Vincent P.** and **Jauvin C.** (2003). A neural probabilistic language model. *Journal of Machine Learning Research* **3**, 1137–1155.

Benoit K., **Muhr D.** and **Watanabe K.** (2017). *stopwords: Multilingual Stopword Lists*. R package version 0.9.0. Available from: https://CRAN.R-project.org/package=stopwords.

Benoit K., **Watanabe K.**, **Wang H.**, **Nulty P.**, **Obeng A.**, **Müller S.** and **Matsuo A.** (2018). quanteda: An R package for the quantitative analysis of textual data. Journal of Open Source Software 3(30), 774. Available from: https://quanteda.io.

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

Berend G. (2018). L1 regularization of word embeddings for multi-word expression identification. *Acta Cybernetica* **23**(3), 801–813.

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*, vol. 1, pp. 1391–1400.

Bernardy J.-P. and **Chatzikyriakidis S.** (2019). What kind of natural language inference are NLP systems learning: Is this enough? In *ICAART (2)*, pp. 919–931. ICAART stands for International Conference on Agents and Artificial Intelligence.

Bhasuran B., **Murugesan G.**, **Abdulkadhar S.** and **Natarajan J.** (2016). Stacked ensemble combined with fuzzy matching for biomedical named entity recognition of diseases. *Journal of Biomedical Informatics* **64**, 1–9.

Bi Y. (2016). *Scheduling Optimization with LDA and Greedy Algorithm*. Master's Thesis, Duke University, Durham NC, United States. LDA stands for latent Dirichlet allocation.

Bird S., **Loper E.** and **Klein E.** (2009). *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. USA: O'Reilly Media Inc.

Blanchard A. (2007). Understanding and customizing stopword lists for enhanced patent mapping. *World Patent Information* **29**(4), 308–316.

Blanco E. and **Moldovan D.** (2011). Some issues on detecting negation from text. In *Proceedings of the Twenty-Fourth International Florida Artificial Intelligence Research Society (FLAIRS) Conference*.

Blei D.M. and **Lafferty J.D.** (2009). Visualizing topics with multi-word expressions. arXiv preprint arXiv:0907.1013.

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

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

Bodapati S., **Yun H.** and **Al-Onaizan Y.** (2019). Robustness to capitalization errors in named entity recognition. arXiv preprint arXiv:1911.05241.

Bokka K.R., **Hora S.**, **Jain T.** and **Wambugu M.** (2019). *Deep Learning for Natural Language Processing: Solve your Natural Language Processing Problems with Smart Deep Neural Networks*. Birmingham, UK: Packt Publishing Ltd.

Bollen J., **Mao H.** and **Zeng X.** (2011). Twitter mood predicts the stock market. *Journal of Computational Science* **2**(1), 1–8.

Bollmann M. (2013). POS tagging for historical texts with sparse training data. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pp. 11–18.

Bollmann M. (2019). A large-scale comparison of historical text normalization systems. arXiv preprint arXiv:1904.02036.

Bouchet-Valat M. (2019). *SnowballC: Snowball Stemmers Based on the C 'libstemmer' UTF-8 Library*. R package version 0.6.0. Available from: https://CRAN.R-project.org/package=SnowballC.

Boukobza R. and **Rappoport A.** (2009). Multi-word expression identification using sentence surface features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pp. 468–477.

Brants T., **Popat A.C.**, **Xu P.**, **Och F.J.** and **Dean J.** (2007). Large language models in machine translation. Technical report, Google Research.

Briscoe T. (1996). The syntax and semantics of punctuation and its use in interpretation. In *Proceedings of the Association for Computational Linguistics Workshop on Punctuation*, pp. 1–7.

Broder A.Z., **Glassman S.C.**, **Manasse M.S.** and **Zweig G.** (1997). Syntactic clustering of the web. *Computer Networks and ISDN Systems* **29**(8–13), 1157–1166. ISDN stands for Integrated Services Digital Network.

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*, pp. 42–47.

Brown P.F., **Desouza P.V.**, **Mercer R.L.**, **Pietra V. J.D.** and **Lai J.C.** (1992). Class-based n-gram models of natural language. *Computational Linguistics* **18**(4), 467–479.

Buck C., **Heafield K.** and **Van Ooyen B.** (2014). N-gram counts and language models from the common crawl. In LREC (International Conference on Language Resources and Evaluation), pp. 3579–3584.

Buerki A. (2017). Frequency consolidation among word n-grams. In *International Conference on Computational and Corpus-Based Phraseology*. Springer, pp. 432–446.

Cabot C., **Soualmia L.F.**, **Dahamna B.** and **Darmoni S.J.** (2016). SIBM at CLEF eHealth Evaluation Lab 2016: Extracting concepts in French medical texts with ECMT and CIMIND. In *CLEF (Working Notes)*, pp. 47–60. CLEF stands for Conference and Labs of the Evaluation Forum.

Cai H., **Yang Y.**, **Li X.** and **Huang Z.** (2015). What are popular: Exploring Twitter features for event detection, tracking and visualization. In *Proceedings of the 23rd ACM International Conference on Multimedia*, pp. 89–98.

Callan J., **Hoy M.**, **Yoo C.** and **Zhao L.** (2009). The ClueWeb09 dataset. Available from: http://lemurproject.org/clueweb09/.

Camacho-Collados J. and **Pilehvar M.T.** (2018). On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 40–46. EMNLP stands for Conference on Empirical Methods in Natural Language Processing.

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

Campbell W.M., **Li L.**, **Dagli C.**, **Acevedo-Aviles J.**, **Geyer K.**, **Campbell J.P.** and **Priebe C.** (2016). Cross-domain entity resolution in social media. arXiv preprint arXiv:1608.01386.

Carvalho G., **de Matos D.M.** and **Rocio V.** (2007). Document retrieval for question answering: A quantitative evaluation of text preprocessing. In *Proceedings of the ACM First PhD Workshop in CIKM*, pp. 125–130. CIKM stands for Conference on Information and Knowledge Management.

Chai C.P. (2017). *Statistical Issues in Quantifying Text Mining Performance*. PhD Dissertation, Duke University, Durham NC, USA.

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.

Chai C.P. (2022). Word distinctivity – Quantifying improvement of topic modeling results from n-gramming. *REVSTAT–Statistical Journal* **20**(2), 199–220.

Chang J.P., **Chiam C.**, **Fu L.**, **Wang A.Z.**, **Zhang J.** and **Danescu-Niculescu-Mizil C.** (2020). Convokit: A toolkit for the analysis of conversations. arXiv preprint arXiv:2005.04246.

Chatterjee S. and **Krystyanczuk M.** (2017). *Python Social Media Analytics*. Birmingham, UK: Packt Publishing Ltd. Book excerpt. Available from: https://hub.packtpub.com/clean-social-media-data-analysis-python/.

Chaudhary G. and **Kshirsagar M.** (2018). Overview and application of text data pre-processing techniques for text mining on health news Tweets. *Helix* **8**(5), 3764–3768.

Chen Q., **Yao L.** and **Yang J.** (2016). Short text classification based on LDA topic model. In *2016 International Conference on Audio, Language and Image Processing (ICALIP)*. IEEE, pp. 749–753. LDA stands for latent Dirichlet allocation.

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

Chen Y., **Zhang H.**, **Liu R.**, **Ye Z.** and **Lin J.** (2019). Experimental explorations on short text topic mining between LDA and NMF based schemes. *Knowledge-Based Systems* **163**, 1–13. *LDA stands for latent Dirichlet allocation, and NMF stands for non-negative matrix factorization.*

Chomsky N. (1969). *Deep Structure, Surface Structure, and Semantic Interpretation*. Indiana University Linguistics Club.

Chua M., **Van Esch D.**, **Coccaro N.**, **Cho E.**, **Bhandari S.** and **Jia L.** (2018). Text normalization infrastructure that scales to hundreds of language varieties. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Cirqueira D., **Pinheiro M.F.**, **Jacob A.**, **Lobato F.** and **Santana Á.** (2018). A literature review in preprocessing for sentiment analysis for Brazilian Portuguese social media. In *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*. IEEE, pp. 746–749.

Clark E. and **Araki K.** (2011). Text normalization in social media: Progress, problems and applications for a pre-processing system of casual English. *Procedia-Social and Behavioral Sciences* **27**, 2–11.

Clough P. (2001). A Perl program for sentence splitting using rules. Technical report, University of Sheffield, Sheffield, United Kingdom.

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*, vol. 3. Association for Computational Linguistics (ACL), pp. 14–20.

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.

Cohen K.B., **Ogren P.V.**, **Fox L.** and **Hunter L.** (2005). Empirical data on corpus design and usage in biomedical natural language processing. In American Medical Informatics Association (AMIA) Annual Symposium Proceedings, vol. 2005, p. 156.

Cohen K.B., **Roeder C.**, **Baumgartner** Jr **W. A.**, **Hunter L.E.** and **Verspoor K.** (2010). Test suite design for ontology concept recognition systems, pp. 441–446.

Cohen K.B., **Tanabe L.**, **Kinoshita S.** and **Hunter L.** (2004). A resource for constructing customized test suites for molecular biology entity identification systems. In *HLT-NAACL 2004 Workshop: Linking Biological Literature, Ontologies and Databases*, pp. 1–8. HLT-NAACL stands for Human Language Technologies – North American Chapter of the Association for Computational Linguistics.

Congjun L. and **Hill N.W.** (2021). Recent developments in Tibetan NLP. *Transactions on Asian and Low-Resource Language Information Processing* **20**(2), 1–3.

Constant M., **Eryiğit G.**, **Monti J.**, **Van Der Plas L.**, **Ramisch C.**, **Rosner M.** and **Todirascu A.** (2017). Multiword expression processing: A survey. *Computational Linguistics* **43**(4), 837–892.

Corbett P. and **Boyle J.** (2018). Improving the learning of chemical-protein interactions from literature using transfer learning and specialized word embeddings. *Database* **2018**, 1–10.

Corral Á., **Boleda G.** and **Ferrer-i Cancho R.** (2015). Zipf's law for word frequencies: Word forms versus lemmas in long texts. *PLOS One* **10**(7), e0129031.

Councill I., **McDonald R.** and **Velikovich L.** (2010). What's great and what's not: Learning to classify the scope of negation for improved sentiment analysis. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing*, pp. 51–59.

Craiger P. and **Shenoi S.** (2007). *Advances in Digital Forensics III: IFIP International Conference on Digital Forensics, National Center for Forensic Science, Orlando Florida,* 28–31 *January 2007*, vol. 242. Springer. IFIP stands for International Federation for Information Processing.

Craven T.C. (2004). Variations in use of meta tag keywords by web pages in different languages. *Journal of Information Science* **30**(3), 268–279.

CrowdFlower (2017). 2017 Data Science Report. Available from: https://visit.figure-eight.com/rs/416-ZBE-142/images/CrowdFlower_DataScienceReport.pdf.

Crystal, **D.** (1999). *The Penguin Dictionary of Language*. London, UK: Penguin Group.

Cutrone L. and **Chang M.** (2011). Auto-assessor: Computerized assessment system for marking student's short-answers automatically. In *2011 IEEE International Conference on Technology for Education*. IEEE, pp. 81–88.

Dařena F. (2019). VecText: Converting documents to vectors. *IAENG International Journal of Computer Science* **46**(2). *IAENG stands for International Association of Engineers.*

Das B., **Pal S.**, **Mondal S.K.**, **Dalui D.** and **Shome S.K.** (2013). Automatic keyword extraction from any text document using N-gram rigid collocation. *International Journal of Soft Computing and Engineering (IJSCE)* **3**(2), 238–242.

Denny M.J. and **Spirling A.** (2018). Text preprocessing for unsupervised learning: Why it matters, when it misleads, and what to do about it. *Political Analysis* **26**(2), 168–189.

Deoras A., **Mikolov T.** and **Church K.** (2011). A fast re-scoring strategy to capture long-distance dependencies. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pp. 1116–1127.

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.

Daz 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*, pp. 40–49.

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

Dinov I.D. (2018). *Data Science and Predictive Analytics: Biomedical and Health Applications using* R. Ann Arbor, MI, USA: Springer.

Dodge J., **Gururangan S.**, **Card D.**, **Schwartz R.** and **Smith N.A.** (2019). Show your work: Improved reporting of experimental results. arXiv preprint arXiv:1909.03004.

Domingo M., **Garca-Martnez M.**, **Helle A.**, **Casacuberta F.** and **Herranz M.** (2018). How much does tokenization affect neural machine translation? arXiv preprint arXiv:1812.08621.

Doraisamy S. and **Rüger S.** (2003). Robust polyphonic music retrieval with n-grams. *Journal of Intelligent Information Systems* **21**(1), 53–70.

Dressel F. (2016). Distribution of n-grams in English text corpus. Available from: http://rpubs.com/fdd/187848.

Dror R., **Baumer G.**, **Shlomov S.** and **Reichart R.** (2018). The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the Fifty-Sixth Annual Meeting of the Association for Computational Linguistics*, vol. 1, pp. 1383–1392.

Du J., **Yu P.** and **Zong C.** (2018). Towards computing technologies on machine parsing of English and Chinese garden path sentences. In *Proceedings of the Future Technologies Conference*. Springer, pp. 806–827.

Duan J., **Lu R.**, **Wu W.**, **Hu Y.** and **Tian Y.** (2006). A bio-inspired approach for multi-word expression extraction. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*. Association for Computational Linguistics, pp. 176–182. COLING stands for International Conference on Computational Linguistics.

Dunning T. (1994). Statistical identification of language. In *Computing Research Laboratory Technical Memo MCCS 94-273*. New Mexico State University.

Duran M.S., **Avanço L.**, **Alusio S.**, **Pardo T.** and **Nunes M.d.G.V.** (2014). Some issues on the normalization of a corpus of products reviews in Portuguese. In *Proceedings of the 9th Web as Corpus Workshop (WaC-9)*, pp. 22–28.

Effrosynidis D., **Symeonidis S.** and **Arampatzis A.** (2017). A comparison of pre-processing techniques for Twitter sentiment analysis. In *International Conference on Theory and Practice of Digital Libraries*. Springer, pp. 394–406.

Ek A., **Bernardy J.-P.** and **Chatzikyriakidis S.** (2020). How does punctuation affect neural models in natural language inference. In *Proceedings of the Probability and Meaning Conference (PaM 2020)*, pp. 109–116.

El-Khair I.A. (2017). Effects of stop words elimination for Arabic information retrieval: A comparative study. arXiv preprint arXiv:1702.01925.

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*, pp. 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.

Fancellu F., **Lopez A.** and **Webber B.** (2016). Neural networks for negation scope detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 495–504.

Fancellu F., **Lopez A.**, **Webber B.** and **He H.** (2017). Detecting negation scope is easy, except when it isn't. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pp. 58–63.

Farzindar A.A. and **Inkpen D.** (2020). Natural language processing for social media. *Synthesis Lectures on Human Language Technologies* **13**(2), 1–219.

Fatyanosa T.N. and **Bachtiar F.A.** (2017). Classification method comparison on Indonesian social media sentiment analysis. In *2017 International Conference on Sustainable Information Engineering and Technology (SIET)*. IEEE, pp. 310–315.

Feinerer I. and **Hornik K.** (2018). *tm: Text Mining Package*. R package version 0.7.6. Available from: https://CRAN.R-project.org/package=tm.

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.

Ferret O., **Grau B.**, **Hurault-Plantet M.**, **Illouz G.**, **Jacquemin C.**, **Monceaux L.** and **Vilnat A.** (2002). How NLP can improve question answering. *Knowledge Organization* **29**(3–4), 135–155.

Fesseha A., **Xiong S.**, **Emiru E.D.**, **Diallo M.** and **Dahou A.** (2021). Text classification based on convolutional neural networks and word embedding for low-resource languages: Tigrinya. *Information* **12**(2), 52.

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*, pp. 20–24.

Fisher N. (2013). *Analytics for Leaders: A Performance Measurement System for Business Success*. Cambridge, UK: Cambridge University Press.

Fisher N. (2019). A comprehensive approach to problems of performance measurement. *Journal of the Royal Statistical Society: Series A (Statistics in Society)* **182**(3), 755–803.

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.

Forst M. and **Kaplan R.M.** (2006). The importance of precise tokenizing for deep grammars. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*.

Fosler-Lussier E. (1998). Markov models and hidden Markov models: A brief tutorial. *International Computer Science Institute*.

Foster J., **Cetinoglu O.**, **Wagner J.**, **Le Roux J.**, **Hogan S.**, **Nivre**, **J.**, **Hogan**, **D.**, and **Van Genabith**, **J.** 2011. # hardtoparse: POS tagging and parsing the Twitterverse. In *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*. AAAI stands for Association for the Advancement of Artificial Intelligence.

Fowler, **F. J.** 1995. *Improving Survey Questions: Design and Evaluation*, vol. **38**. Thousand Oaks, CA, USA: Sage.

Fox, **C.** 1989. A stop list for general text. In *ACM SIGIR Forum*, vol. **24**, pp. 19–21. ACM. SIGIR stands for Special Interest Group on Information Retrieval.

Friedman H.H. and **Amoo T.** (1999). Rating the rating scales. *Journal of Marketing Management* **9**(3), 114–123.

Frigau L., **Wu Q.** and **Banks D.** (2021). Optimizing the JSM program. *Journal of the American Statistical Association* **00**(0), 1–10. JSM stands for Joint Statistical Meetings.

Fundel K., **Küffner**, **R.** and **Zimmer R.** (2007). RelEx – relation extraction using dependency parse trees. *Bioinformatics* **23**(3), 365–371.

Gabernet A.R. and **Limburn J.** (2017). Breaking the 80/20 rule: How data catalogs transform data scientists' productivity. Available from: https://www.ibm.com/cloud/blog/ibm-data-catalog-data-scientists-productivity.

Gamallo P., **Campos J.R.P.** and **Alegria I.** (2017). A perplexity-based method for similar languages discrimination. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, pp. 109–114.

Garíca S., **Ramírez-Gallego S.**, **Luengo J.**, **Benítez J.M.** and **Herrera F.** (2016). Big data preprocessing: Methods and prospects. *Big Data Analytics* **1**(1), 1–22.

Gentzkow M., **Kelly B.T.** and **Taddy M.** (2017). Text as data. Technical report, National Bureau of Economic Research, Cambridge, MA, USA.

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.

Gerz D., **Vulić I.**, **Ponti E.**, **Naradowsky J.**, **Reichart R.** and **Korhonen A.** (2018). Language modeling for morphologically rich languages: Character-aware modeling for word-level prediction. *Transactions of the Association for Computational Linguistics* **6**, 451–465.

Ghosh S., **Johansson R.**, **Riccardi G.** and **Tonelli S.** (2011). Shallow discourse parsing with conditional random fields. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pp. 1071–1079.

Globerson A. and **Roweis S.** (2006). Nightmare at test time: Robust learning by feature deletion. In *Proceedings of the Twenty-Third International Conference on Machine Learning*. ACM, pp. 353–360.

Goldberg Y. and **Orwant J.** (2013). A dataset of syntactic-ngrams over time from a very large corpus of English books. Technical report, Google Research.

Gomes F.B., **Adán-Coello J.M.** and **Kintschner F.E.** (2018). Studying the effects of text preprocessing and ensemble methods on sentiment analysis of Brazilian Portuguese Tweets. In *International Conference on Statistical Language and Speech Processing*. Springer, pp. 167–177.

Goodman E.L., **Zimmerman C.** and **Hudson C.** (2020). Packet2vec: Utilizing word2vec for feature extraction in packet data. arXiv preprint arXiv:2004.14477.

Gorrell G., **Petrak J.** and **Bontcheva K.** (2015). Using @Twitter conventions to improve #LOD-based named entity disambiguation. In *European Semantic Web Conference*. Springer, pp. 171–186. LOD stands for Linked Open Data.

Goyvaerts J. and **Levithan S.** (2012). *Regular Expressions Cookbook*. Sebastopol, CA, USA: O'Reilly Media Inc.

Grabar N., **Zweigenbaum P.**, **Soualmia L.** and **Darmoni S.** (2003). Matching controlled vocabulary words. In *Studies in Health Technology and Informatics,* pp. 445–450.

Grana J., **Alonso M.A.** and **Vilares M.** (2002). A common solution for tokenization and part-of-speech tagging. In *International Conference on Text, Speech and Dialogue*, vol. 2448. Springer, pp. 3–10.

Grefenstette G. and **Tapanainen P.** (1994). What is a word, what is a sentence? Problems of tokenisation. Technical report, Rank Xerox Research Centre, Grenoble Laboratory, Meylan, France.

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. ICE stands for International Corpus of English.

Grishman R. (2015). Information extraction. *IEEE Intelligent Systems* **30**(5), 8–15.

Grobelnik M. and **Mladenic D.** (2004). Text-mining tutorial. Technical report, Jožef Stefan Institute (JSI), Slovenia.

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.

Groza T. and **Verspoor K.** (2014). Automated generation of test suites for error analysis of concept recognition systems. In *Proceedings of the Australasian Language Technology Association Workshop 2014*, pp. 23–31.

Guthrie D., **Allison B.**, **Liu W.**, **Guthrie L.** and **Wilks Y.** (2006). A closer look at skip-gram modelling. *In LREC (International Conference on Language Resources and Evaluation)*, vol. **6**, pp. 1222–1225.

Ha L., **Hanna P.**, **Ming J.** and **Smith F.** (2009). Extending Zipf's law to n-grams for large corpora. *Artificial Intelligence Review* **32**(1–4), 101–113.

Habert B., **Adda G.**, **Adda-Decker M.**, **de Marëuil P.B.**, **Ferrari S.**, **Ferret O.**, **Illouz G.** and **Paroubek P.** (1998). Towards tokenization evaluation. In *Proceedings of the First International Conference on Language Resources and Evaluation (LREC)*, pp. 427–431.

HaCohen-Kerner Y., **Miller D.** and **Yigal Y.** (2020). The influence of preprocessing on text classification using a bag-of-words representation. *PLOS One* **15**(5), e0232525.

Haddi E., **Liu X.** and **Shi Y.** (2013). The role of text pre-processing in sentiment analysis. *Procedia Computer Science* **17**, 26–32.

Hajba G.L. (2018). Using beautiful soup. In Website Scraping with Python. Springer, pp. 41–96.

Hansen C., **Hansen C.**, **Simonsen J.G.** and **Lioma C.** (2018). The Copenhagen team participation in the check-worthiness task of the competition of automatic identification and verification of claims in political debates of the CLEF-2018 CheckThat! lab. In *CLEF (Working Notes)*. CLEF stands for Conference and Labs of the Evaluation Forum.

Hartrumpf S., **Helbig H.** and **Osswald R.** (2006). Semantic interpretation of prepositions for NLP applications. In *Proceedings of the Third ACL-SIGSEM Workshop on Prepositions*.

He W., **Zha S.** and **Li L.** (2013). Social media competitive analysis and text mining: A case study in the pizza industry. *International Journal of Information Management* **33**(3), 464–472.

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.

Hedderich M.A., **Lange L.**, **Adel H.**, **Strötgen J.** and **Klakow D.** (2020). A survey on recent approaches for natural language processing in low-resource scenarios. arXiv preprint arXiv:2010.12309.

Heimerl F., **Lohmann S.**, **Lange S.** and **Ertl T.** (2014). Word cloud explorer: Text analytics based on word clouds. In *2014 Forty-Seventh Hawaii International Conference on System Sciences*. IEEE, pp. 1833–1842.

Henry T. (2016). Quick ngram processing script. Available from: https://github.com/trh3/NGramProcessing.

Hernández M.A. and **Stolfo S.J.** (1998). Real-world data is dirty: Data cleansing and the merge/purge problem. *Data Mining and Knowledge Discovery* **2**(1), 9–37.

Hickman L., **Thapa S.**, **Tay L.**, **Cao M.** and **Srinivasan P.** (2020). Text preprocessing for text mining in organizational research: Review and recommendations. In Organizational Research Methods, pp. 1–58.

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*, pp. 1620–1629.

Hoffman B. (1996). Translating into free word order languages. In *Proceedings of the 16th Conference on Computational Linguistics-Volume 1*, pp. 556–561.

Hofherr P.C. (2012). Preposition-determiner amalgams in German and French at the syntax-morphology interface. In **Ackema P.** (ed), *Comparative Germanic Syntax: The State of the Art*, Amsterdam, Netherlands. John Benjamins Publishing Company, pp. 99–132.

Houvardas J. and **Stamatatos E.** (2006). N-gram feature selection for authorship identification. In *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*. Springer, pp. 77–86.

Huang H. and **Zhang B.** (2009). Text segmentation. In **Liu L.** and **Özsu M.T.** (eds), Encyclopedia of Database Systems, vol. 6. Springer, pp. 3072–3075.

Huston S., **Culpepper J.S.** and **Croft W.B.** (2014). Indexing word sequences for ranked retrieval. *ACM Transactions on Information Systems (TOIS)* **32**(1), 1–26.

Huston S., **Moffat A.** and **Croft W.B.** (2011). Efficient indexing of repeated n-grams. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, pp. 127–136.

Hutto C. and **Gilbert E.** (2014). VADER: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 8, pp. 216–225. AAAI stands for Association for the Advancement of Artificial Intelligence.

Hvitfeldt E. and **Silge J.** (2021). *Supervised Machine Learning for Text Analysis in* R. New York, NY, USA: Chapman and Hall/CRC.

Indurkhya N. and **Damerau F.J.** (2010). *Handbook of Natural Language Processing*. New York, NY, USA: Chapman and Hall/CRC.

Irfan R., **King**, **C.K.**, **Grages D.**, **Ewen S.**, **Khan S.U.**, **Madani**, **S.A.**, **Kolodziej J.**, **Wang L.**, **Chen D.** and **Rayes A.** (2015). A survey on text mining in social networks. *The Knowledge Engineering Review* **30**(2), 157–170.

Jean-Baptiste E. (1916). Gammes sténographiques. Technical report, Institut Sténographique de France, Paris.

Ježek E. (2016). *The Lexicon: An Introduction*. Oxford, UK: Oxford University Press.

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*, vol. 1, pp. 13–24.

Ji Z., **Wei Q.** and **Xu H.** (2020). BERT-based ranking for biomedical entity normalization. *AMIA Summits on Translational Science Proceedings*, 2020, 269–277. AMIA stands for American Medical Informatics Association.

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

Jiang Z., **Li L.**, **Huang D.** and **Jin L.** (2015). Training word embeddings for deep learning in biomedical text mining tasks. In *2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, pp. 625–628.

Jimenez M., **Maxime C.**, **Le Traon Y.** and **Papadakis M.** (2018). On the impact of tokenizer and parameters on n-gram based code analysis. In *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, pp. 437–448.

Jivani A.G. (2011). A comparative study of stemming algorithms. *International Journal of Computer Technology and Applications (IJCTA)* **2**(6), 1930–1938.

Jones B.E. (1994). Exploring the role of punctuation in parsing natural text. In *Proceedings of the Fifteenth Conference on Computational Linguistics*, vol. 1. Association for Computational Linguistics, pp. 421–425.

Jones R. (2006). *Internet Slang Dictionary*. Durham NC, USA: Lulu.com.

Joty S., **Carenini G.**, **Ng R.** and **Murray G.** (2019). Discourse analysis and its applications. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pp. 12–17.

Joulin A., **Grave E.**, **Bojanowski P.** and **Mikolov T.** (2016). Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759.

Jurafsky D. and **Martin J.H.** (2008). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 2nd Edn., Fort Collins, CO, USA: Prentice Hall Series in Artificial Intelligence.

Jusoh S. (2018). A study on NLP applications and ambiguity problems. *Journal of Theoretical & Applied Information Technology* **96**(6), 1486–1499.

Kadhim A.I. (2018). An evaluation of preprocessing techniques for text classification. *International Journal of Computer Science and Information Security (IJCSIS)* **16**(6), 22–32.

Kaewphan S., **Mehryary F.**, **Hakala K.**, **Salakoski T.** and **Ginter F.** (2017). TurkuNLP entry for interactive Bio-ID assignment. In *Proceedings of the BioCreative VI Workshop*, pp. 32–35.

Kaiser E. and **Trueswell J.C.** (2004). The role of discourse context in the processing of a flexible word-order language. *Cognition* **94**(2), 113–147.

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)*, vol. 14, pp. 71–75.

Kamps J., **Adafre S.F.** and **De Rijke M.** (2004). Effective translation, tokenization and combination for cross-lingual retrieval. In *Workshop of the Cross-Language Evaluation Forum for European Languages*. Springer, pp. 123–134.

Kannan S. and **Gurusamy V.** (2014). Preprocessing techniques for text mining. *International Journal of Computer Science & Communication Networks* **5**(1), 7–16.

Kao A. and **Poteet S.R.** (2007). *Natural Language Processing and Text Mining*. London, UK: Springer Science & Business Media.

Karthikeyan S., **Jotheeswaran J.**, **Balamurugan B.** and **Chatterjee**, **J.M.** (2020). Text mining. In *Natural Language Processing in Artificial Intelligence*, pp. 167–210.

Kathuria A., **Gupta A.** and **Singla R.** (2021). A review of tools and techniques for preprocessing of textual data. In *Computational Methods and Data Engineering*, pp. 407–422.

Kaufmann M. and **Kalita J.** (2010). Syntactic normalization of Twitter messages. In *International Conference on Natural Language Processing*, *Kharagpur, India.*

Kaur J. and **Buttar P.K.** (2018). A systematic review on stopword removal algorithms. *International Journal on Future Revolution in Computer Science & Communication Engineering* **4**(4), 207–210.

Kaviani M. and **Rahmani H.** (2020). EmHash: Hashtag recommendation using neural network based on BERT embedding. In *2020 6th International Conference on Web Research (ICWR)*. IEEE, pp. 113–118.

Khyani D., **Siddhartha B.**, **Niveditha N.** and **Divya B.** (2020). An interpretation of lemmatization and stemming in natural language processing. *Journal of University of Shanghai for Science and Technology* **22**(10), 350–357.

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

Kiss T. and **Strunk J.** (2006). Unsupervised multilingual sentence boundary detection. *Computational Linguistics* **32**(4), 485–525.

Kitaev N., **Cao S.** and **Klein D.** (2019). Multilingual constituency parsing with self-attention and pre-training. arXiv preprint arXiv:1812.11760.

Koehn P., **Arun A.** and **Hoang H.** (2008). Towards better machine translation quality for the German-English language pairs. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pp. 139–142.

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

Korenius T., **Laurikkala J.**, **Järvelin K.** and **Juhola M.** (2004). Stemming and lemmatization in the clustering of Finnish text documents. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, pp. 625–633.

Koto F. and **Adriani M.** (2015). A comparative study on Twitter sentiment analysis: Which features are good? In *International Conference on Applications of Natural Language to Information Systems*. Springer, pp. 453–457.

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

Koulali R. and **Meziane A.** (2011). Topic detection and multi-word terms extraction for Arabic unvowelized documents. In *Asia Information Retrieval Symposium*. Springer, pp. 614–623.
Kozakou E. (2017). *Word Adaptions in the Language of Twitter*. Master's Thesis, Leiden University, Leiden, Netherlands.
Kudo T. and **Richardson J.** (2018). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. arXiv preprint arXiv:1808.06226.
Kulkarni A. and **Shivananda A.** (2019). *Natural Language Processing Recipes*. Berkeley, CA, USA: Springer.
Kunilovskaya M. and **Plum A.** (2021). Text preprocessing and its implications in a digital humanities project. In *Proceedings of the Student Research Workshop Associated with RANLP 2021*, pp. 85–93. RANLP stands for International Conference on Recent Advances in Natural Language Processing.
Kutuzov A., **Fares M.**, **Oepen S.** and **Velldal E.** (2017). Word vectors, reuse, and replicability: Towards a community repository of large-text resources. In *Proceedings of the 58th Conference on Simulation and Modelling*, Linköping, Sweden. Linköping University Electronic Press, pp. 271–276.
Kwak H., **Lee C.**, **Park H.** and **Moon S.** (2010). What is Twitter, a social network or a news media? In *Proceedings of the Nineteenth International Conference on World Wide Web*. ACM, pp. 591–600.
Kwartler T. (2017). *Text Mining in Practice with* R. Hoboken, NJ, USA: John Wiley & Sons.
Labusch M., **Kotz S.A.** and **Perea M.** (2022). The impact of capitalized German words on lexical access. *Psychological Research* **2022**(86), 891–902.
Lafferty J.D. and **Blei D.M.** (2006). Correlated topic models. In Advances in Neural Information Processing Systems, pp. 147–154.
Lahiri S. and **Mihalcea R.** (2013). Using n-gram and word network features for native language identification. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pp. 251–259.
Lambert N.J. (2017). Text mining tutorial. In *Group Processes*. Edinburgh, Scotland, UK: Springer, Cham, pp. 93–117.
Lambert P. and **Banchs R.E.** (2006). Grouping multi-word expressions according to part-of-speech in statistical machine translation. In *Proceedings of the Workshop on Multi-Word-Expressions in a Multilingual Context*.
Lan Y. (1996). Chomskian deep structure and translation. *Perspectives: Studies in Translatology* **4**(1), 103–113.
Lazarinis F. (2007). Evaluating the searching capabilities of e-commerce web sites in a non-English language. *Online Information Review* **31**(6), 881–891.
Le H.P. and **Ho T.V.** (2008). A maximum entropy approach to sentence boundary detection of Vietnamese texts. In *IEEE International Conference on Research, Innovation and Vision for the Future-RIVF 2008*.
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.
Leek J.T. and **Jager L.R.** (2017). Is most published research really false? *Annual Review of Statistics and Its Application* **4**, 109–122.
Lende S.P. and **Raghuwanshi M.** (2016). Question answering system on education acts using NLP techniques. In *2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave)*. IEEE, pp. 1–6.
Lextek International n.d. Onix text retrieval toolkit – API reference. Available from: https://www.lextek.com/manuals/onix/ [last accessed August 2019].
Li C., **Duan Y.**, **Wang H.**, **Zhang Z.**, **Sun A.** and **Ma Z.** (2017). Enhancing topic modeling for short texts with auxiliary word embeddings. *ACM Transactions on Information Systems (TOIS)* **36**(2), 1–30.
Li C., **Wang H.**, **Zhang Z.**, **Sun A.** and **Ma Z.** (2016). Topic modeling for short texts with auxiliary word embeddings. In *Proceedings of the Thirty-Ninth International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, pp. 165–174.
Li L., **Shao Y.**, **Song D.**, **Qiu X.** and **Huang X.** (2020). Generating adversarial examples in Chinese texts using sentence-pieces. arXiv preprint arXiv:2012.14769.
Li X. and **Croft W.B.** (2001). Incorporating syntactic information in question answering. Technical report, Center for Intelligent Information Retrieval, University of Massachusetts at Amherst, Amherst MA, USA.
Lin C.-Y. and **Hovy E.** (2003). Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 150–157.
Lin J., **Nogueira R.** and **Yates A.** (2020). Pretrained transformers for text ranking: BERT and beyond. arXiv preprint arXiv:2010.06467.
Lita L.V., **Ittycheriah A.**, **Roukos S.** and **Kambhatla N.** (2003). tRuEcasIng. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pp. 152–159.
Litvak M. and **Vanetik N.** (2019). *Multilingual Text Analysis: Challenges, Models, and Approaches*. Singapore: World Scientific.
Liu B. (2015). *Sentiment Analysis: Mining Opinions, Sentiments, and Emotions*. Cambridge, UK: Cambridge University Press.
Liu H., **Christiansen T.**, **Baumgartner W.A.** and **Verspoor K.** (2012). BioLemmatizer: A lemmatization tool for morphological processing of biomedical text. *Journal of Biomedical Semantics* **3**(1), 1–29.
Liu L. and **Özsu M.T.** (2009). *Encyclopedia of Database Systems*, vol. **6**. New York, NY, USA: Springer.
Liu Z. and **Jansen B.J.** (2013). Factors influencing the response rate in social question and answering behavior. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, pp. 1263–1274.

Lo R.T.-W., **He B.** and **Ounis I.** (2005). Automatically building a stopword list for an information retrieval system. In Journal on Digital Information Management: Special Issue on the 5th Dutch-Belgian Information Retrieval Workshop (DIR), vol. 5, pp. 17–24.

Loginova E., **Varanasi S.** and **Neumann G.** (2018). Towards multilingual neural question answering. In *European Conference on Advances in Databases and Information Systems*. Springer, pp. 274–285.

Los B. and **Lubbers T.** (2019). Syntax, text type, genre and authorial voice in old English: A data-driven approach. *In Grammar–Discourse–Context: Grammar and Usage in Language Variation and Change*, vol. **23**, p. 49.

Lourentzou I., **Manghnani K.** and **Zhai C.** (2019). Adapting sequence to sequence models for text normalization in social media. In *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 13, pp. 335–345. AAAI stands for Association for the Advancement of Artificial Intelligence.

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

Lüdeling A. and **Kytö M.** (2008). *Corpus Linguistics*, vol. **1**. Berlin, Germany: Walter de Gruyter.

Luhn H.P. (1959). Keyword-in-context index for technical literature (KWIC index). *American Documentation* **11**(4), 288–295.

Luiz O.J., **Olden J.D.**, **Kennard M.J.**, **Crook D.A.**, **Douglas M.M.**, **Saunders T.M.** and **King A.J.** (2019). Trait-based ecology of fishes: A quantitative assessment of literature trends and knowledge gaps using topic modelling. *Fish and Fisheries* **20**(6), 1100–1110.

Luo J., **Tinsley J.** and **Lepage Y.** (2013). Exploiting parallel corpus for handling out-of-vocabulary words. In *Proceedings of the 27th Pacific Asia Conference on Language, Information, and Computation (PACLIC 27)*, pp. 399–408.

Lusetti M., **Ruzsics T.**, **Göhring A.**, **Samardžić T.** and **Stark E.** (2018). Encoder-decoder methods for text normalization. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*. Association for Computational Linguistics, pp. 18–28.

Makrehchi M. and **Kamel M.S.** (2008). Automatic extraction of domain-specific stopwords from labeled documents. In *European Conference on Information Retrieval*. Springer, pp. 222–233.

Makrehchi M. and **Kamel M.S.** (2017). Extracting domain-specific stopwords for text classifiers. *Intelligent Data Analysis* **21**(1), 39–62.

Malvern D. and **Richards B.** (2012). Measures of lexical richness. In *The Encyclopedia of Applied Linguistics*.

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

Mansurov B. and **Mansurov A.** (2021). Uzbek Cyrillic-Latin-Cyrillic machine transliteration. arXiv preprint arXiv:2101.05162.

Marcu D. (2000). *The Theory and Practice of Discourse Parsing and Summarization*. Cambridge, MA, USA: MIT Press.

Masini F. (2005). Multi-word expressions between syntax and the lexicon: The case of Italian verb-particle constructions. *SKY Journal of Linguistics* **18**(2005), 145–173. *SKY stands for Suomen kielitieteellinen yhdistys, from the Linguistic Association of Finland.*

Masini F. (2019). Multi-word expressions and morphology. In *Oxford Research Encyclopedia of Linguistics*.

Matusov E., **Leusch G.**, **Bender O.** and **Ney H.** (2005). Evaluating machine translation output with automatic sentence segmentation. In *International Workshop on Spoken Language Translation (IWSLT)*.

McAuliffe J.D. and **Blei D.M.** (2008). Supervised topic models. In *Advances in Neural Information Processing Systems,* pp. 121–128.

McNamee P. and **Mayfield J.** (2007). N-gram morphemes for retrieval. In *CLEF (Working Notes)*. CLEF stands for the Cross-Language Evaluation Forum workshop.

Mendoza M., **Poblete B.** and **Castillo C.** (2010). Twitter under crisis: Can we trust what we RT? In *Proceedings of the First Workshop on Social Media Analytics*. ACM, pp. 71–79.

Merlo P. (2019). Probing word and sentence embeddings for long-distance dependencies effects in French and English. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 158–172.

Metzler H., **Baginski H.**, **Niederkrotenthaler T.** and **Garcia D.** (2021). Detecting potentially harmful and protective suicide-related content on Twitter: A machine learning approach. arXiv preprint arXiv:2112.04796.

Meyer D., **Hornik K.** and **Feinerer I.** (2008). Text mining infrastructure in R. *Journal of Statistical Software* **25**(5), 1–54.

Miah M. (2009). Improved k-NN algorithm for text classification. In *Proceedings of the 2009 International Conference on Data Mining (DMIN)*. Citeseer, pp. 434–440.

Microsoft (2019). Extract n-gram features from text. Available from: https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/extract-n-gram-features-from-text.

Mieke S.S. (2016). Language diversity in ACL 2004–2016. ACL stands for the annual meeting of the Association for Computational Linguistics. Available from: https://sjmielke.com/acl-language-diversity.htm.

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

Miller J.-A. (2014). Language: Much ado about what? In *Lacan and the Subject of Language*, pp. 21–35.

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

Mitrofan M. and **Tufiş D.** (2018). Bioro: The biomedical corpus for the Romanian language. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Moh T.-S. and **Zhang Z.** (2012). Cross-lingual text classification with model translation and document translation. In *Proceedings of the 50th Annual Southeast Regional Conference*, pp. 71–76.

Moon S. and **Okazaki N.** (2020). Jamo pair encoding: Subcharacter representation-based extreme Korean vocabulary compression for efficient subword tokenization. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pp. 3490–3497.

Moon S. and **Okazaki N.** (2021). Effects and mitigation of out-of-vocabulary in universal language models. *Journal of Information Processing* **29**, 490–503.

Morante R. and **Blanco E.** (2021). Recent advances in processing negation. *Natural Language Engineering* **27**(2), 1–10.

Morgan A.A., **Lu Z.**, **Wang X.**, **Cohen A.M.**, **Fluck J.**, **Ruch P.**, **Divoli A.**, **Fundel K.**, **Leaman R.** and **Hakenberg J.** (2008). Overview of BioCreative II gene normalization. *Genome Biology* **9**(2), 1–19.

Mostafa M.M. (2013). More than words: Social networks' text mining for consumer brand sentiments. *Expert Systems with Applications* **40**(10), 4241–4251.

Mubarak H. (2017). Build fast and accurate lemmatization for Arabic. arXiv preprint arXiv:1710.06700.

Mullen L.A., **Benoit K.**, **Keyes O.**, **Selivanov D.** and **Arnold J.** (2018). Fast, consistent tokenization of natural language text. *Journal of Open Source Software* **3**(23), 655.

Munro R. (2015). Language at ACL this year. ACL stands for the annual conference of the Association for Computational Linguistics. Available from: http://www.junglelightspeed.com/languages-at-acl-this-year/.

Namly D., **Bouzoubaa K.**, **El Jihad A.** and **Aouragh S.L.** (2020). Improving Arabic lemmatization through a lemmas database and a machine-learning technique. In *Recent Advances in NLP: The Case of Arabic Language*. Springer, pp. 81–100.

Narala S., **Rani B.P.** and **Ramakrishna K.** (2017). Telugu text categorization using language models. Global Journal of Computer Science and Technology **16**(4), 9–13.

Nasir A., **Aslam K.**, **Tariq S.** and **Ullah M.F.** (2020). Predicting mental illness using social media posts and comments. *International Journal of Advanced Computer Science and Applications* **11**(12), 607–613.

Nayak A.S., **Kanive A.**, **Chandavekar N.** and **Balasubramani R.** (2016). Survey on pre-processing techniques for text mining. *International Journal of Engineering and Computer Science* **5**(6) 2319–7242.

Nayel H.A., **Shashirekha H.**, **Shindo H.** and **Matsumoto Y.** (2019). Improving multi-word entity recognition for biomedical texts. arXiv preprint arXiv:1908.05691.

Névéol A., **Robert A.**, **Anderson R.**, **Cohen K.B.**, **Grouin C.**, **Lavergne T.**, **Rey G.**, **Rondet C.** and **Zweigenbaum**, **P.** (2017). CLEF eHealth 2017 multilingual information extraction task overview: ICD10 coding of death certificates in English and French. In *CLEF (Working Notes)*. CLEF stands for Conference and Labs of the Evaluation Forum.

Newman M.E. (2005). Power laws, Pareto distributions and Zipf's law. *Contemporary Physics* **46**(5), 323–351.

Ng D., **Bansal M.** and **Curran J.R.** (2015). Web-scale surface and syntactic n-gram features for dependency parsing. arXiv preprint arXiv:1502.07038.

Nguyen D.Q., **Billingsley R.**, **Du L.** and **Johnson M.** (2015). Improving topic models with latent feature word representations. *Transactions of the Association for Computational Linguistics* **3**, 299–313.

Nicolai G. and **Kondrak G.** (2016). Leveraging inflection tables for stemming and lemmatization. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1138–1147.

Nivre J. (2005). Dependency grammar and dependency parsing. *MSI Report* **5133**(1959), 1–32. MSI report is from Växjö University, Växjö, Sweden.

Nothman J., **Qin H.** and **Yurchak R.** (2018). Stop word lists in free open-source software packages. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pp. 7–12.

Novák A. and **Siklósi B.** (2015). Automatic diacritics restoration for Hungarian. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 2286–2291.

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.

Nunberg G. (1990). *The Linguistics of Punctuation*, vol. **18**. *Center for the Study of Language (CSLI)*, Stanford, CA, USA: Stanford University.

Nuzumlal M.Y. *and* **Özgür A.** *(2014). Analyzing stemming approaches for Turkish multi-document summarization. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 702–706.*

Olde B.A., **Hoeffner J.**, **Chipman P.**, **Graesser A.C.** and **Tutoring Research Group** (1999). A connectionist model for part of speech tagging. In *Florida Artificial Intelligence Research Society (FLAIRS) Conference*, pp. 172–176.

Paice C. and **Hooper R.** (2005). Lancaster stemmer. Available from: http://www.comp.lancs.ac.uk/computing/research/stemming/ [last accessed August 2019].

Pais V., **Ion R.**, **Avram A.-M.**, **Mitrofan M.** and **Tufis D.** (2021). In-depth evaluation of Romanian natural language processing pipelines. *Romanian Journal of Information Science and Technology* **24**(4), 384–401.

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

Pak A. and **Paroubek P.** (2010). Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC)*, pp. 1320–1326.

Pantel P. and **Pennacchiotti M.** (2006). Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the Twenty-First International Conference on Computational Linguistics and the Forty-Fourth Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 113–120.

Papakyriakopoulos O., **Hegelich S.**, **Serrano J.C.M.** and **Marco F.** (2020). Bias in word embeddings. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pp. 446–457.

Pareti S., **O'Keefe T.**, **Konstas I.**, **Curran J.R.** and **Koprinska I.** (2013). Automatically detecting and attributing indirect quotations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 989–999.

Patil A., **Pharande K.**, **Nale D.** and **Agrawal R.** (2015). Automatic text summarization. *International Journal of Computer Applications* **109**(17), 18–19.

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.

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

Peitz S., **Freitag M.**, **Mauser A.** and **Ney H.** (2011). Modeling punctuation prediction as machine translation. In *International Workshop on Spoken Language Translation* (IWSLT).

Peldszus A. and **Stede M.** (2015). Joint prediction in MST-style discourse parsing for argumentation mining. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 938–948. MST stands for minimum spanning tree.

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)*. IEEE, pp. 5364–5367.

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)*, pp. 1532–1543.

Perkins J. (2014). *Python 3 Text Processing with NLTK 3 Cookbook*. Birmingham, UK: Packt Publishing Ltd.

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.

Petic M. and **Gîfu D.** (2014). Transliteration and alignment of parallel texts from Cyrillic to Latin. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pp. 1819–1823.

Piktus A., **Edizel N.B.**, **Bojanowski P.**, **Grave É.**, **Ferreira R.** and **Silvestri F.** (2019). Misspelling oblivious word embeddings. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3226–3234.

Poddar L. (2016). Multilingual multiword expressions. arXiv preprint arXiv:1612.00246.

Polignano M., **Basile P.**, **De Gemmis M.**, **Semeraro G.** and **Basile V.** (2019). Alberto: Italian BERT language understanding model for NLP challenging tasks based on Tweets. In *6th Italian Conference on Computational Linguistics*, *CLiC-it 2019*, vol. 2481. CEUR Workshop Proceedings, pp. 1–6.

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*. Springer, pp. 113–127.

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

Posada J.D., **Barda A.J.**, **Shi L.**, **Xue D.**, **Ruiz V.**, **Kuan P.-H.**, **Ryan N.D.** and **Tsui F.R.** (2017). Predictive modeling for classification of positive valence system symptom severity from initial psychiatric evaluation records. *Journal of Biomedical Informatics* **75**, S94–S104.

Potts C. (n.d). Sentiment symposium tutorial: Stemming. Available from: http://sentiment.christopherpotts.net/stemming.html [last accessed August 2019].

Preethi V. (2021). Survey on text transformation using Bi-LSTM in natural language processing with text data. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)* **12**(9), 2577–2585.

Przepiórkowski A., **Piasecki M.**, **Jassem K.** and **Fuglewicz P.** (2012). *Computational Linguistics: Applications*, vol. **458**. Heidelberg, Germany: Springer.

Ptaszynski M., **Lempa P.**, **Masui F.**, **Kimura Y.**, **Rzepka R.**, **Araki K.**, **Wroczynski M.** and **Leliwa G.** (2019). Brute-force sentence pattern extortion from harmful messages for cyberbullying detection. *Journal of the Association for Information Systems* **20**(8), 4.

Putra S.J., **Gunawan M.N.** and **Suryatno A.** (2018). Tokenization and n-gram for indexing Indonesian translation of the Quran. In *2018 6th International Conference on Information and Communication Technology (ICoICT)*. IEEE, pp. 158–161.

Qian Z., **Li P.**, **Zhu Q.**, **Zhou G.**, **Luo Z.** and **Luo W.** (2016). Speculation and negation scope detection via convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 815–825.

Qudar M. and **Mago V.** (2020). A survey on language models. Technical report, Lakehead University, Thunder Bay, Ontario, Canada.

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

Raff E., **Fleming W.**, **Zak R.**, **Anderson H.**, **Finlayson B.**, **Nicholas C.** and **McLean M.** (2019). Kilograms: Very large n-grams for malware classification. arXiv preprint arXiv:1908.00200.

Raff E. and **Nicholas C.** (2018). Hash-grams: Faster n-gram features for classification and malware detection. In *Proceedings of the ACM Symposium on Document Engineering 2018*, pp. 1–4.

Rahm E. and **Do H.H.** (2000). Data cleaning: Problems and current approaches. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering* **23**(4), 3–13.

Rahman R. (2017). Detecting emotion from text and emoticon. *London Journal of Research in Computer Science and Technology* **17**(3), 9–13.

Rajaraman A. and **Ullman J.D.** (2011). *Mining of Massive Datasets*. Cambridge, UK: Cambridge University Press.

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.

Raulji J.K. and **Saini J.R.** (2016). Stop-word removal algorithm and its implementation for Sanskrit language. *International Journal of Computer Applications* **150**(2), 15–17.

Reber U. (2019). Overcoming language barriers: Assessing the potential of machine translation and topic modeling for the comparative analysis of multilingual text corpora. *Communication Methods and Measures* **13**(2), 102–125.

Řehůřek R. and **Sojka P.** (2010). Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pp. 45–50. LREC stands for Conference on Language Resources and Evaluation.

Reynar J.C. and **Ratnaparkhi A.** (1997). A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*. Association for Computational Linguistics, pp. 16–19.

Richardson L. (2020). Beautiful Soup 4.9.1. Python library. Available from: https://www.crummy.com/software/BeautifulSoup/.

Rikters M. and **Bojar O.** (2017). Paying attention to multi-word expressions in neural machine translation. arXiv preprint arXiv:1710.06313.

Rinker T.W. (2018a). *lexicon: Lexicon Data*. R package version 1.2.1. Available from: http://github.com/trinker/lexicon.

Rinker T.W. (2018b). *textclean: Text Cleaning Tools*. R package version 0.9.3. Available from: https://github.com/trinker/textclean.

Roberts M.E., **Stewart B.M.**, **Tingley D.**, **Lucas C.**, **Leder-Luis J.**, **Gadarian S.K.**, **Albertson B.** and **Rand D.G.** (2014). Structural topic models for open-ended survey responses. *American Journal of Political Science* **58**(4), 1064–1082.

Robinson D. (2018). *gutenbergr: Download and Process Public Domain Works from Project Gutenberg*. R package version 0.1.4. Available from: https://CRAN.R-project.org/package=gutenbergr.

Rosenthal S. and **McKeown K.** (2013). Columbia NLP: Sentiment detection of subjective phrases in social media. In *Second Joint Conference on Lexical and Computational Semantics (SEM): Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, vol. 2, pp. 478–482.

Rosid M.A., **Fitrani A.S.**, **Astutik I.R.I.**, **Mulloh N.I.** and **Gozali H.A.** (2020). Improving text preprocessing for student complaint document classification using Sastrawi. *In IOP Conference Series: Materials Science and Engineering*, vol. **874**, 012017, Bristol, UK: IOP Publishing.

Saal F.E., **Downey R.G.** and **Lahey M.A.** (1980). Rating the ratings: Assessing the psychometric quality of rating data. *Psychological Bulletin* **88**(2), 413.

Sadvilkar N. and **Neumann M.** (2020). PySBD: Pragmatic sentence boundary disambiguation. arXiv preprint arXiv:2010.09657.

Sag I.A., **Baldwin T.**, **Bond F.**, **Copestake A.** and **Flickinger D.** (2002). Multiword expressions: A pain in the neck for NLP. In *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, pp. 1–15.

Sailer M. and **Markantonatou S.** (2018). *Multiword Expressions: Insights from a Multi-Lingual Perspective*. Berlin, Germany: Language Science Press.

Samad M.D., **Khounviengxay N.D.** and **Witherow M.A.** (2020). Effect of text processing steps on Twitter sentiment classification using word embedding. arXiv preprint arXiv:2007.13027.

Samir A. and **Lahbib Z.** (2018). Stemming and lemmatization for information retrieval systems in Amazigh language. In *International Conference on Big Data, Cloud and Applications*. Springer, pp. 222–233.

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.

Sandhaus E. (2008). The New York Times annotated corpus. *Linguistic Data Consortium, Philadelphia* **6**(12), e26752. Available from: https://catalog.ldc.upenn.edu/LDC2008T19.

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

Sarkar D. (2019). *Text Analytics with Python: A Practitioner's Guide to Natural Language Processing*. Bangalore, India: Apress.

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*, vol. 2, pp. 432–436.

Schuman H. and **Presser S.** (1996). *Questions and Answers in Attitude Surveys: Experiments on Question Form, Wording, and Context*. Thousand Oaks, CA, USA: Sage.

Schuur Y. (2020). *Normalization for Dutch for Improved POS Tagging*. Master's Thesis, University of Groningen, Groningen, Netherlands.

Şeker G.A. and **Eryiğit G.** (2012). Initial explorations on using CRFs for Turkish named entity recognition. In *Proceedings of COLING 2012*, pp. 2459–2474. CRFs stand for conditional random fields, and COLING stands for International Conference on Computational Linguistics.

Seretan V. (2011). *Syntax-Based Collocation Extraction*, vol. **44**. Heidelberg, Germany: Springer Science & Business Media.

Shaikh A., **More D.**, **Puttoo R.**, **Shrivastav S.** and **Shinde S.** (2019). A survey paper on chatbots. *International Research Journal of Engineering and Technology* (IRJET) **6**(4), 1786–1789.

Shalaby W., **Zadrozny W.** and **Jin H.** (2019). Beyond word embeddings: Learning entity and concept representations from large scale knowledge bases. *Information Retrieval Journal* **22**(6), 525–542.

Silge J. and **Robinson D.** (2017). *Text Mining with* R: *A Tidy Approach*. Sebastopol, CA, USA: O'Reilly Media Inc.

Sinclair J. (2018). *Collins English Dictionary*, 13th Edn. New York NY, USA: HarperCollins.

Singh V. and **Saini B.** (2014). An effective tokenization algorithm for information retrieval systems. In *First International Conference on Data Mining*, pp. 109–119.

Singhal A. (2001). Modern information retrieval: A brief overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering* **24**(4), 35–43.

Slioussar N. (2011). Processing of a free word order language: The role of syntax and context. *Journal of Psycholinguistic Research* **40**(4), 291–306.

Søgaard A., **de Lhoneux M.** and **Augenstein I.** (2018). Nightmare at test time: How punctuation prevents parsers from generalizing. arXiv preprint arXiv:1809.00070.

Soriano J., **Au T.** and **Banks D.** (2013). Text mining in computational advertising. *Statistical Analysis and Data Mining: The ASA Data Science Journal* **6**(4), 273–285. ASA stands for American Statistical Association.

Soricut R. and **Marcu D.** (2003). Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 228–235.

Stamatatos E. (2008). Author identification: Using text sampling to handle the class imbalance problem. *Information Processing & Management* **44**(2), 790–799.

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, Bergen, Norway.

Tabii Y., **Lazaar M.**, **Al Achhab M.** and **Enneya N.** (2018). *Big Data, Cloud and Applications (BDCA): Third International Conference, BDCA 2018, Kenitra, Morocco, 4–5 April 2018, Revised Selected Papers*, vol, 872. Springer.

Taboada M., **Meizoso M.**, **Martínez D.**, **Riano D.** and **Alonso A.** (2013). Combining open-source natural language processing tools to parse clinical practice guidelines. *Expert Systems* **30**(1), 3–11.

Tan F., **Hu Y.**, **Hu C.**, **Li K.** and **Yen K.** (2020). TNT: Text normalization based pre-training of transformers for content moderation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4735–4741.

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*, pp. 201–206.

Tang C., **Ross K.**, **Saxena N.** and **Chen R.** (2011). What's in a name: A study of names, gender inference, and gender behavior in Facebook. In *International Conference on Database Systems for Advanced Applications*. Springer, pp. 344–356.

Tang J., **Hong M.**, **Zhang D.L.** and **Li J.** (2008). Information extraction: Methodologies and applications. In Emerging Technologies of Text Mining: Techniques and Applications. USA: IGI Global, pp. 1–33.

Temnikova I., **Nikolova I.**, **Baumgartner** Jr **W.A.**, **Angelova G.** and **Cohen K.B.** (2013). Closure properties of Bulgarian clinical text. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)*, pp. 667–675.

Thanaki J. (2017). *Python Natural Language Processing*. Birmingham, UK: Packt Publishing Ltd.

Thione G. and **van den Berg M.** (2007). Systems and methods for structural indexing of natural language text. Google Patents. US Patent Application No. 11/405,385.

Toral A., **Pecina P.**, **Wang L.** and **Van Genabith J.** (2015). Linguistically-augmented perplexity-based data selection for language models. *Computer Speech & Language* **32**(1), 11–26.

Torres-Moreno J.-M. (2012). Beyond stemming and lemmatization: Ultra-stemming to improve automatic text summarization. arXiv preprint arXiv:1209.3126.

Torres-Moreno J.-M. (2014). *Automatic Text Summarization*. Hoboken, NJ, USA: John Wiley & Sons.

Tran D. and **Sharma D.** (2005). Markov models for written language identification. In *Proceedings of the 12th International Conference on Neural Information Processing*, pp. 67–70.

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*, pp. 803–804.

Truss L. (2004). *Eats, Shoots & Leaves: The Zero Tolerance Approach to Punctuation*. New York, NY, USA: Penguin.

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

Truss L. (2006). *Eats, Shoots & Leaves: Why, Commas Really do make a Difference!* New York, NY, USA: 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*. Association for Computational Linguistics, pp. 1–12. NAACL-HLT stands for the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.

Venhuizen N.J., **Bos J.**, **Hendriks P.** and **Brouwer H.** (2018). Discourse semantics with information structure. *Journal of Semantics* **35**(1), 127–169.

Verma R.M. and **Marchette D.J.** (2019). *Cybersecurity Analytics*. New York, NY, USA: CRC Press.

Verspoor C.M., **Joslyn C.** and **Papcun G.J.** (2003). The gene ontology as a source of lexical semantic knowledge for a biological natural language processing application. In SIGIR Workshop on Text Analysis and Search for Bioinformatics, pp. 51–56.

Verspoor K., **Dvorkin D.**, **Cohen K.B.** and **Hunter L.** (2009). Ontology quality assurance through analysis of term transformations. *Bioinformatics* **25**(12), i77–i84.

Vieweg S., **Hughes A.L.**, **Starbird K.** and **Palen L.** (2010). Microblogging during two natural hazards events: What Twitter may contribute to situational awareness. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, pp. 1079–1088. SIGCHI stands for Special Interest Group on Computer-Human Interaction.

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

Vilares J., **Vilares M.**, **Alonso M.A.** and **Oakes M.P.** (2016). On the feasibility of character n-grams pseudo-translation for cross-language information retrieval tasks. *Computer Speech & Language* **36**, 136–164.

Villares-Maldonado R. (2017). A corpus-based analysis of non-standard English features in the microblogging platform Tumblr. *EPiC Series in Language and Linguistics* **2**, 295–303.

Wallach H.M. (2006). Topic modeling: Beyond bag-of-words. In *Proceedings of the Twenty-Third International Conference on Machine Learning*. ACM, pp. 977–984.

Wang H., **Liu L.**, **Song W.** and **Lu J.** (2014a). Feature-based sentiment analysis approach for product reviews. *Journal of Software* **9**(2), 274–279.

Wang W., **Gu Z.** and **Tang K.** (2020). An improved algorithm for Bert. In *Proceedings of the 2020 International Conference on Cyberspace Innovation of Advanced Technologies*, pp. 116–120.

Wang X., **McCallum A.** and **Wei X.** (2007). Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*. IEEE, pp. 697–702.

Wang X., **Tokarchuk L.** and **Poslad S.** (2014b). Identifying relevant event content for real-time event detection. In *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014),* pp. 395–398.

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.

Webster J.J. and **Kit C.** (1992). Tokenization as the initial phase in NLP. In *Proceedings of the 14th Conference on Computational Linguistics*, pp. 1106–1110.

Wei Z., **Miao D.**, **Chauchat J.-H.**, **Zhao R.** and **Li W.** (2009). N-grams based feature selection and text representation for Chinese text classification. *International Journal of Computational Intelligence Systems* **2**(4), 365–374.

Welbers K., **Van Atteveldt W.** and **Benoit K.** (2017). Text analysis in R. *Communication Methods and Measures* **11**(4), 245–265.

Wermter J. and **Hahn U.** (2005). Paradigmatic modifiability statistics for the extraction of complex multi-word terms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pp. 843–850.

White L., **Togneri R.**, **Liu W.** and **Bennamoun M.** (2018). *Neural Representations of Natural Language*, vol. **783**. Singapore: Springer.

White R.W. and **Morris D.** (2007). Investigating the querying and browsing behavior of advanced search engine users. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 255–262.

Widyassari A.P., **Rustad S.**, **Shidik G.F.**, **Noersasongko E.**, **Syukur A.** and **Affandy A.** (2022). Review of automatic text summarization techniques & methods. *Journal of King Saud University-Computer and Information Sciences* **34**(4), 1029–1046.

Wijffels J. (2021). *udpipe: Tokenization, Parts of Speech Tagging, Lemmatization and Dependency Parsing with the 'UDPipe' 'NLP' Toolkit*. R package version 0.8.8. Available from: https://cran.r-project.org/web/packages/udpipe/udpipe.pdf.

Wilson K.S. (2009). Database search control. US Patent Application No. 12/031,701.

Wong D.F., **Chao L.S.** and **Zeng X.** (2014). iSentenizer-μ: Multilingual sentence boundary detection model. The Scientific World Journal 2014, 1–10.

Woo H., **Kim J.** and **Lee W.** (2020). Validation of text data preprocessing using a neural network model. *Mathematical Problems in Engineering* **2020**, 1–9.

Xue J., **Chen J.**, **Hu R.**, **Chen C.**, **Zheng C.** and **Zhu T.** (2020). Twitter discussions and concerns about COVID-19 pandemic: Twitter data analysis using a machine learning approach. arXiv preprint arXiv: 2005.12830.

https://doi.org/10.1017/S1351324922000213 Published online by Cambridge University Press

Yamada I., **Takeda H.** and **Takefuji Y.** (2015). Enhancing named entity recognition in Twitter messages using entity linking. In *Proceedings of the Workshop on Noisy User-Generated Text*, pp. 136–140.

Yang Y. and **Wilbur J.** (1996). Using corpus statistics to remove redundant words in text categorization. *Journal of the American Society for Information Science* **47**(5), 357–369.

Yazdani A., **Ghazisaeedi M.**, **Ahmadinejad N.**, **Giti M.**, **Amjadi H.** and **Nahvijou A.** (2020). Automated misspelling detection and correction in Persian clinical text. *Journal of Digital Imaging* **33**(3), 555–562.

Ye X., **Shen H.**, **Ma X.**, **Bunescu R.** and **Liu C.** (2016). From word embeddings to document similarities for improved information retrieval in software engineering. In *Proceedings of the 38th International Conference on Software Engineering*, pp. 404–415.

Yeh K.C. (2003). Bilingual sentence alignment based on punctuation marks. In *Proceedings of the ROCLING 2003 Student Workshop*, pp. 303–312. ROCLING stands for Conference on Computational Linguistics and Speech Processing.

Yin D., **Ren F.**, **Jiang P.** and **Kuroiwa S.** (2007). Chinese complex long sentences processing method for Chinese-Japanese machine translation. In *2007 International Conference on Natural Language Processing and Knowledge Engineering*. IEEE, pp. 170–175.

Yolchuyeva S., **Németh G.** and **Gyires-Tóth B.** (2018). Text normalization with convolutional neural networks. *International Journal of Speech Technology* **21**(3), 589–600.

Yuhang Z., **Yue W.** and **Wei Y.** (2010). Research on data cleaning in text clustering. In *2010* International Forum on Information Technology and Applications, vol. 1. IEEE, pp. 305–307.

Zaman A., **Matsakis P.** and **Brown C.** (2011). Evaluation of stop word lists in text retrieval using latent semantic indexing. In *2011 Sixth International Conference on Digital Information Management*. IEEE, pp. 133–136.

Zeng C., **Li S.**, **Li Q.**, **Hu J.** and **Hu J.** (2020). A survey on machine reading comprehension–tasks, evaluation metrics and benchmark datasets. *Applied Sciences* **10**(21), 7640.

Zeng Z., **Xu H.**, **Chong T.Y.**, **Chng E.-S.** and **Li H.** (2017). Improving N-gram language modeling for code-switching speech recognition. In *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, pp. 1596–1601.

Zhang C., **Baldwin T.**, **Ho H.**, **Kimelfeld B.** and **Li Y.** (2013). Adaptive parser-centric text normalization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1159–1168.

Zhang H. (2008). Exploring regularity in source code: Software science and Zipf's law. In *2008 15th Working Conference on Reverse Engineering*. IEEE, pp. 101–110.

Zhang J. and **El-Gohary N.M.** (2017). Integrating semantic NLP and logic reasoning into a unified system for fully-automated code checking. *Automation in Construction* **73**, 45–57.

Zhang L. and **Komachi M.** (2018). Neural machine translation of logographic languages using sub-character level information. arXiv preprint arXiv:1809.02694.

Zhang X. and **LeCun Y.** (2017). Which encoding is the best for text classification in Chinese, English, Japanese and Korean? arXiv preprint arXiv:1708.02657.

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.

Zhou L., **Siddiqui T.**, **Seliger S.L.**, **Blumenthal J.B.**, **Kang Y.**, **Doerfler R.** and **Fink J.C.** (2019). Text preprocessing for improving hypoglycemia detection from clinical notes – A case study of patients with diabetes. *International Journal of Medical Informatics* **129**, 374–380.

Zhou M., **Duan N.**, **Liu S.** and **Shum H.-Y.** (2020). Progress in neural NLP: Modeling, learning, and reasoning. *Engineering* **6**(3), 275–290.

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

Zubiaga A., **Spina D.**, **Marítnez R.** and **Fresno V.** (2015). Real-time classification of Twitter trends. *Journal of the Association for Information Science and Technology* **66**(3), 462–473.

Zupon A., **Crew E.** and **Ritchie S.** (2021). Text normalization for low-resource languages of Africa. arXiv preprint arXiv:2103.15845.

Zweigenbaum P. and **Grabar N.** (2002a). Accenting unknown words: Application to the French version of the MeSH. In *Workshop NLP in Biomedical Applications, EFMI, Cyprus,* 69á. EFMI stands for European Federation for Medical Informatics.

Zweigenbaum P. and **Grabar N.** (2002b). Restoring accents in unknown biomedical words: Application to the French MeSH thesaurus. *International Journal of Medical Informatics* **67**(1–3), 113–126.

Cite this article: Chai CP (2023). Comparison of text preprocessing methods. *Natural Language Engineering* **29**, 509–553. https://doi.org/10.1017/S1351324922000213