Word Distinctivity – Quantifying Improvement of Topic Modeling Results from N-Gramming *

Authors: CHRISTINE P. CHAI
– Core Services Engineering Operations (CSEO),
Microsoft Corporation,
Redmond WA, United States (cpchai21@gmail.com)

Abstract:

- Text data cleaning is an important but often overlooked step in text mining because it is difficult to quantify the contribution. Therefore, we propose the word distinctivity to measure the improvement of topic modeling results from n-gramming, which preserves special phrases in a corpus. The word distinctivity evaluates the signal strength of a word's topic assignments, and a high distinctivity means a high posterior probability for the word to come from a certain topic. We implemented the latent Dirichlet allocation for topic modeling, and discovered that some special phrases show an increase in word distinctivity, reducing uncertainty in topic identification.

Key-Words:

- *Latent Dirichlet allocation; text mining; topic modeling; n-gramming; data cleaning; quantification.*

AMS Subject Classification:

- 62-07, 62C10, 68U15

*The research was conducted while the author was a PhD student in statistical science at Duke University. The opinions and views expressed in this manuscript are those of the author and do not necessarily state or reflect those of Microsoft.

1. INTRODUCTION

Data cleaning is an essential step to prepare the data for analysis [34, 54], because most datasets in real life contain much noise and missing values [22]. For example, a table consisting of a variable "years of education" may encode a missing value as "99". It is almost impossible for a person to have 99 years of education, and we need to remove this value before running a regression on the data. Otherwise, the regression results would be distorted.

Nevertheless, data cleaning is often overlooked for various reasons. One reason is that early-career professionals focus on the analysis due to the over-emphasis of statistical modeling in graduate school programs [41]. Another reason is that data cleaning is often viewed as a tedious and time-consuming task [43]. According to a report done by CrowdFlower in 2016 [14], most data scientists spend more than half of their work time cleaning and organizing data.

1.1. Text Data Cleaning

The under-appreciation of text data cleaning is a more severe problem because people tend to care less about text data than numerical data [48, 44]. This is unfortunate, although understandable because text data are unstructured and more difficult to analyze than the numerical counterparts [39]. Many characteristics of numerical data are not transferable to text data, such as mean and standard deviation.

Recently, due to the emerging need of text data mining [17], more and more resources are available for text data processing [18, 3, 40]. However, most of them describe text data cleaning as an important step before the analysis, without providing concrete evidence of why this step is crucial. If we can quantify the text data cleaning results, the importance of preprocessing the data is clearly demonstrated. Quantifiable results are published in many different fields to show new research findings, and text data cleaning results should not be an exception.

A unique issue with text data is that many statistical models perform random permuation of words and do not account for word order [55], resulting in confusion and loss of semantic information. For example, the two sentences "the department chair couches offers" and "the chair department offers couches" comprise the exact same words, but with completely different meanings. The first sentence probably came from a university administration report, and the second sentence may be written by a retail store [50].

1.2. N-Gramming and Word Distinctivity

Text data cleaning needs to preserve the word order to mitgate the problem, and one common solution is n-gramming [47, 26], i.e. retaining special phrases in the corpus, such as "white house" and "new york". The phrases are called n-grams [5]. The n-gramming process helps recover semantic information because words within a preserved special phrase are regarded as a single token in text analysis. For example, the two sentences "The white person lives in the house." and "The person lives in the White House." contain the exact same words but have completely different meaning. If the term "White House" is separated as two words, the original meaning is lost.

N-gramming has been widely used in natural language processing, such as machine translation [53], speech recognition [52], and information retrieval [25, 15]. Discussion and comparison of various n-gram sizes (i.e., how many words each n-gram contains) are also extensive in the literature [29, 32].

Nevertheless, few previous studies [50] evaluate the information gain from n-gramming, not to mention quantifying it. Even though many researchers in the text mining field regard n-gramming as a necessity in text data processing, it can be challenging to explain to business stakeholders why n-gramming is worth the time spent. Most business stakeholders would like to see quantifiable results, such as "a 20% increase in model accuracy."

To quantify the improvement of text classification results from n-gramming, we propose the "word distinctivity" as a metric. Word distinctivity refers to how "distinctive" a word is, or in plain language, how likely the word is assigned to a certain topic.

In mathematical terms, word distinctivity is defined as

$$\max_i P(\text{ topic } i|\text{ word } j,\ \text{data }),$$

i.e., taking the maximum probability of how likely a topic i is, given a specific word j and the data.

Here, text classification is also known as topic modeling, the classification process that assigns text documents into various topics.

1.3. Overview of Topic Modeling

Topic modeling is actually an automated process of classifying text documents into different topics, which is a part of natural language processing. A common approach for topic modeling is the latent Dirichlet allocation (LDA) [6],

and the algorithm outputs a topic assignment vector for each document, as well as the "top" words for each topic. The number of topics is a preset constant, while the contents of each topic are to be determined by the text corpus. LDA seems to be the standard algorithm of topic modeling because many researchers extend LDA to more advanced topic models [35, 38, 27].

Nevertheless, to the best of my knowledge, no one has questioned the fundamental criteria of how LDA selects the "top" words in each topic – the selection is based on the posterior probability $P(\text{ word } j|\text{ topic } i,\text{ data })$. This answers the question "Given topic i and the data, which words would the model generate?" But more often than not, we are given a document with words, and would like to know which topic(s) the words belong to. Hence a better selection criteria is $P(\text{ topic } i|\text{ word } j,\text{ data })$, which is the word distinctivity before we take the maximum probability across each topic.

Since topic models assign each word in the corpus to one or more topics, we use the word distinctivity to measure the signal strength generated by each word. If a word (or a retained phrase) has a high probability to be assigned to a particular topic, the word is considered highly distinctive. Upon seeing this word, we know that it is highly likely that the word came from the particular topic. On the contrary, if a word is equally likely to be assigned to all topics, the word has low distinctivity.

To compare and quantify the topic modeling results, we created two versions of the same text dataset – before and after n-gramming and implemented the LDA algorithm with the same number of preset topics. To show the improvement from n-gramming, we look at the word distinctivity of a retained phrase and the word distinctivity of each word in that phrase. The former is expected to be much higher than any of the latter.

2. DATA DESCRIPTION

Our text dataset originates from the collection of 109,055 blog posts from the top 467 US political blogs, as ranked in 2012 by Technorati (now Synacor) [24]. The blog posts were written in English. The blog post collection was obtained from MaxPoint Interactive (now Valassis Digital), where the computer scientists web-scraped the text and stemmed the words using a modified version of Snowball [30] developed in-house. Therefore, the words in the corpus are actually tokens, but we use the terms "word" and "token" interchangeably.

"Stemming" a word removes its suffix and keeps only its root, and the output is a "token". In this way, words of the same root are consolidated into the same token. For example, according to the `wordStem` function in the R package `SnowballC` [7], "worry" and its present principle form "worrying" are both assigned to the same token "worri". When we summarize the word counts in the

corpus, we may see that "worry" appears 50 times and "worrying" appears 30 times. After stemming the corpus, we would see that "worri" appears a combination of 80 times. The stemming process not only reduces the size of vocabulary, but also increases the readability of the results.

We focus on the articles relevant to the fatal shooting of Trayvon Martin by George Zimmerman on February 26, 2012[1], which triggered a heated debate on the media and many political blogs. Among the 109,055 blog posts, 450 contains the keyword "Trayvon", and the 450 blog posts form the actual corpus for analysis. We call the text corpus the "Trayvon Martin dataset", and we generated two versions of this dataset.

2.1. First Version: Stop Words Removed (Before N-Gramming)

In the first version, we removed predefined stop words (approximately 300) with little semantic meaning (e.g. "to", "for") from the Trayvon Martin Dataset. Note that negation terms, such as "no", "not", and "don't", are excluded from the stop word list, because they can reverse the meaning of the next word. For example, "not a good idea" means "a bad idea".

Since many topic models are bag-of-words models and do not preserve word order [42], one solution [11] is to replace words and a preceding negation term with its corresponding antonym, e.g. "not good" becomes "bad". However, this is outside the paper's scope because we would like to focus on the improvement of topic modeling results from n-gramming, instead of adding another variation to the data.

2.2. Second Version: Special Phrases Retained (After N-Gramming)

In the second version of the dataset, we identified and preserved the special phrases. The process is called n-gramming, whose goal is to keep sets of words with high probability of co-occurrence. If a special phrase contains n words, it is called an n-gram. In particular, a word can be called a uni-gram; a two-word phrase retained this way is a bi-gram, and a three-word phrase of this kind is a tri-gram. Section 3.1 explains the n-gramming methodology in detail.

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

3. METHODS

The methods section describes the n-gramming process, the latent Dirichlet allocation (LDA) algorithm, and the word distinctivity measure. The n-gramming process is used to keep certain words together, so their order would not be affected by the bag-of-words models, which assume an orderless document representation. The LDA is used for topic modeling, and it determines which document contains which topic(s) in a probabilistic way. The word distinctivity measure determines which word(s) have a strong signal in topic identification, and this measure can be computed from the LDA topic assignment vectors.

3.1. N-Gramming

The objective of n-gramming is to retain phrases with high probability of occurrence. One obvious solution is to select phrases that appears many times in the corpus, but this is likely to include many common expressions with little semantic meaning.

A major question we also try to answer is, "Given a particular word, how likely is this word going to follow it?" The Turbo Topics [5] software demonstrated an example: Given the word "new" in their corpus, the word "york" follows it 60% of the time. Therefore, we can infer that "new york" is a bi-gram.

To identify the n-grams, we start by searching for all n-word phrases (a.k.a. n-gram candidates) and filter them in terms of raw frequency and conditional probability. Next, we set certain thresholds to determine which are the actual n-grams, and finally provide the implementation results.

3.1.1. Search for N-Gram Candidates

We retrieve all n-gram candidates in the corpus by "shingling" at the word level [45, 8], a standard approach of slicing down a long sentence into phrases with n words each [19, 9]. In comparison, "shingling" at the character level creates each n-gram candidate as a string of n characters, which is not of interest here [49, 10].

If a sentence contains n words, then there are n uni-grams, $n-1$ bi-gram candidates, and $n-2$ tri-gram candidates. For example, "heard gun shot outside apartment yesterday" contains 6 words, 5 two-word phrases, and 4 three-word phrases, listed as below:

- Text: “heard gun shot outside apartment yesterday”
- Two-word phrases: “heard gun”, “gun shot”, “shot outside”, “outside apartment”, and “apartment yesterday”
- Three-word phrases: “heard gun shot”, “gun shot outside”, “shot outside apartment”, and “outside apartment yesterday”

Note that “n-gram candidates” are different from “n-grams”. The former term refers to the n-word phrases which can potentially be n-grams. The latter term “n-grams” refers to only the selected ones n-word phrases, typically with both practical and statistical significance. How we select actual n-grams from the candidates is described next.

3.1.2. Raw Frequency: Practical Significance

For an n-word phrase to “qualify” as an n-gram candidate, the phrase must occur at least a certain number of times in the data, so setting a minimum cut-off frequency is essential [51, 16]. The raw frequency threshold corresponds to the practical significance of n-grams and removes rare words, which have little semantic meaning in the corpus. One example is the name of the police officer who arrested George Zimmerman.

A default minimum frequency of 5 is recommended by the Microsoft Azure Machine Learning Studio [31], but this is too low for the Trayvon Martin dataset. The results include lots of unmeaningful phrases, such as “countri better” and “claim obama”.

Instead, the cutoff for phrase counts should be determined by corpus size, and we empirically set it to 100 for the bi-grams in the Trayvon Martin dataset. That is, to be considered a bi-gram candidate, any two-word phrase has to appear in the corpus at least 100 times.

3.1.3. Conditional Probability: Statistical Significance

The conditional probability is also a widely used approach to filter out n-gram candidates [9, 13], and this measures the statistical significance of each n-word phrase. The conditional probability in n-gramming is denoted as

$$P(\text{word } n | \text{ words } 1, \cdots, n-1),$$

i.e., the probability of getting the nth word given the first $n-1$ words.

In mathematical terms, the marginal probability $P(\text{word})$ is the frequency of the word, divided by the total number of words in the corpus. Similarly, $P(\text{ words } 1, \cdots, m)$ is the frequency of the m-word phrase, divided by the total number of words in the corpus.

Hence the conditional probability of n-grams is written as

$$\begin{aligned} P(\text{word } n | \text{ words } 1, \cdots, n-1) &= \frac{P(\text{words } 1, \cdots, n)}{P(\text{ words } 1, \cdots, n-1)} \\ &= \frac{\text{Frequency of words } 1, \cdots, n}{\text{Frequency of words } 1, \cdots, n-1}. \end{aligned}$$

Particularly, the conditional probability for bi-grams is $P(\text{word 2} \mid \text{word 1})$. If the answer is "yes" to the question "Is Word 2 more likely to follow Word 1?", then the two words should form a bi-gram.

The hypotheses are:

- $H_0 : P(\text{word 2} \mid \text{word 1}) \leq P(\text{word 2})$
- $H_1 : P(\text{word 2} \mid \text{word 1}) > P(\text{word 2})$

The p-value cutoff is set to 0.05 by default, and this removes most words which just happened to appear together. For example, "said obama" is a common phrase but not a meaningful bi-gram, and the high frequency is due to the high marginal probability of "said".

Note that the raw frequency cutoff is also crucial, since rare phrases can distort the conditional probability and produce undesirable results. As an extreme example, if the first word appears only once in the data, the second word following the first word has conditional probability of 100%.

3.2. Latent Dirichlet Allocation (LDA)

LDA is a Bayesian data generative process that performs topic modeling, i.e., classifies documents and words into topics. The LDA algorithm first draws each topic from a Dirichlet distribution as the prior, then updates the probabilities by using the words in the documents. Finally, the algorithm outputs the `top.topic.words` for each topic, based on the posterior probability $P(\text{ word } j | \text{ topic } i, \text{ data })$ for each combination of topic i and word j.

For each document, LDA outputs the topic proportions – the probabilistic topic assignment vector. The number of components of this vector is equal to the preset number of topics. The probabilities for topics are defined using word counts, i.e., the number of relevant words in the document. For example, (0.5, 0.3, 0.2) means the document has topic proportions 50% in Topic 1, 30% in Topic

2, and 20% in Topic 3. In this document, 50% of the words belong to Topic 1, 30% of the words belong to Topic 2, and 20% of the words belong to Topic 3.

LDA also produces topic assignments at the word level, which is the main usage in this paper. For example, a word has a probabilistic topic assignment vector (0.5, 0.3, 0.2), and we implemented 100 simulations. This means the word is assigned to Topic 1 for 50 times, assigned to Topic 2 for 30 times, and assigned to Topic 3 for 20 times.

3.2.1. Setup

The LDA algorithm assumes the corpus $\mathbf{D}$ to be a fixed set of M documents and the words from a finite vocaublary set $\mathbf{W}$. The LDA also requires a predefined number of topics K, and the setup is specified as below.

- Fixed set of M documents: $\mathbf{D} = \{D_1, \cdots, D_M\}$
- Words within a document D_d: $\mathbf{W}_d = \{w_{d,1}, \cdots, w_{d,N_d}\}$
  - The document D_d contains N_d words.
- Finite vocabulary set: $\mathbf{W} = \mathbf{W}_1 \bigcup \cdots \bigcup \mathbf{W}_M$, with size N
  - $\mathbf{W}$ is the union of all sets $\mathbf{W}_d$, where $d = 1, \cdots, M$.
  - The vocabulary set of $\mathbf{D}$ contains N words in total.
- Predefined number of topics K
- Fixed vectors $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_K)$ and $\boldsymbol{\beta} = (\beta_1, \cdots, \beta_K)$

For the parameters, we set the number of topics to $K = 5$ and $\alpha_i = 0.1, \beta_i = 0.1$ for all $i = 1, \cdots, K$, same as previous researchers did on the Trayvon Martin dataset [42, 1].

3.2.2. Algorithm Description

The data generative process of LDA is defined as below, and the plate diagram is illustrated in Figure 1. The word proportion vector ϕ_k determines the relative "weights" of each word in topic k, and the topic proportion vector θ_d determines how the document D_d is composed from each of the K topics.

- For each topic k
  - Draw a word proportion vector: $\phi_k|\boldsymbol{\alpha} \sim \text{Dirichlet}_N(\boldsymbol{\alpha})$
- For each document D_d
  - Draw a topic proportion vector: $\theta_d|\boldsymbol{\beta} \sim \text{Dirichlet}_K(\boldsymbol{\beta})$
  - For each word $w_{d,i}$ in document D_d
    - Draw a topic assignment $z_{d,i}|\theta_d \sim \text{Multinomial}(\theta_d)$
    - Draw a word from the topic $w_{d,i}|\phi_{z_{d,i}} \sim \text{Multinomial}(\phi_{z_{d,i}})$

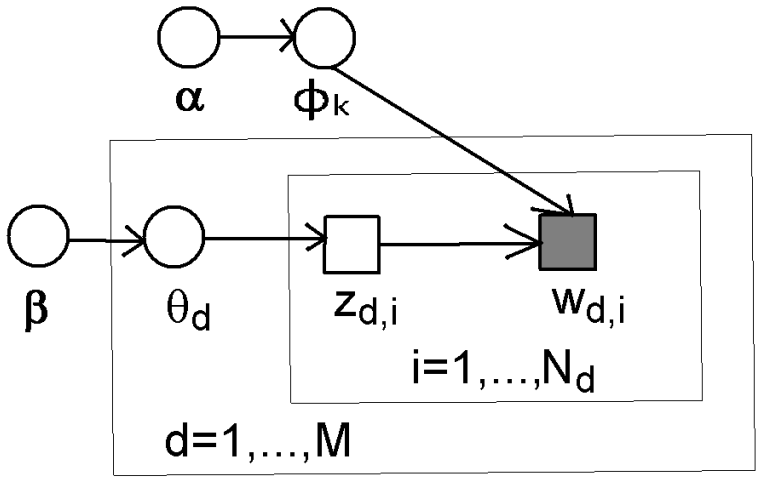

Figure 1: Plate diagram for the LDA process

The full posterior model specification of LDA is:

$$p(\theta_{1:M}, z_{1:M,1:N}, \phi_{1:K}|w_{1:M,1:N}, \boldsymbol{\alpha}, \boldsymbol{\beta}, K)$$
$$= \frac{p(\theta_{1:M}, z_{1:M,1:N}, \phi_{1:K}|\boldsymbol{\alpha}, \boldsymbol{\beta}, K) \times p(w_{1:M,1:N}|\theta_{1:M}, z_{1:M,1:N}, \phi_{1:K}, \boldsymbol{\alpha}, \boldsymbol{\beta}, K)}{\int_{\phi_{1:K}} \int_{\theta_{1:M}} \sum_{z_{1:M,1:N}} p(\theta_{1:M}, z_{1:M,1:N}, \phi_{1:K}, w_{1:M,1:N}|\boldsymbol{\alpha}, \boldsymbol{\beta}, K)}.$$

Since a document D_d has only N_d words, $w_{d,n} = 0$ for all $n > N_d$, i.e., a non-existent word. Similarly, $z_{d,n} = 0$ for all $n > N_d$, i.e., a non-existent topic assignment.

The prior $p(\boldsymbol{\theta}, \mathbf{z}, \boldsymbol{\phi}|\boldsymbol{\alpha}, \boldsymbol{\beta}, K)$ can also be written as

$$p(\boldsymbol{\phi}|\boldsymbol{\alpha}, K)p(\boldsymbol{\theta}|\boldsymbol{\beta}, K)p(\mathbf{z}|\boldsymbol{\theta}),$$

which is derived from the data generative process.

On the other hand, the likelihood in the denominator is intractable, and this requires Markov Chain Monte Carlo or variational inference methods to compute. Existing solutions include a variational Bayes approximation approach [6] and a collapsed Gibbs sampler that integrates out both $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ [20].

3.3. Word Distinctivity

Word distinctivity measures how "distinctive" a word is in terms of topic classification, and this can be regarded as an add-on to the LDA model. Since LDA returns words of the highest (posterior) probability given each topic and the data, the output compares many words and decides which ones should be assigned to the particular topic. In contrast, word distinctivity compares the assignment probabilities across topics for a particular word, so when we see the word, we know how likely it is from a certain topic.

Word distinctivity is defined as the highest posterior probability of a word to be assigned to a particular topic. For example, if a word w_1 has a topic assignment vector of (0.1, 0.6, 0.3), the word distinctivity of w_1 is 0.6. To put it differently, word distinctivity is the maximum signal level observed from the word.

For another example, assume the words w_2, w_3 have topic assignment vectors (0.33, 0.34, 0.33) and (0.80, 0.10, 0.10), respectively. Then w_2 has low distinctivity because its topic assignment vector nearly corresponds to a discrete uniform distribution, which has the largest entropy. On the other hand, w_3 has high distinctivity, because given w_3 and the data, we are 80% sure that the word w_3 came from the first topic.

Using the Bayes' theorem, we can convert the direct output of LDA $P(\text{ word } j|\text{ topic } i,\text{ data })$ into the word distinctivity candidates for word j:

$$P(\text{ topic } i|\text{ word } j,\text{ data }) = \frac{P(\text{ word } j|\text{ topic } i,\text{ data })P(\text{ topic } i)}{\sum_{\text{topic } k} P(\text{ word } j|\text{ topic } k,\text{ data })P(\text{ topic } k)}.$$

Then we take the maximum probability across all topics, and the value is the word distinctivity of word j:

$$\max_i P(\text{ topic } i|\text{ word } j,\text{ data }).$$

4. IMPLEMENTATION RESULTS

We demonstrated the statistical methods on the Trayvon Martin dataset (described in Section 2). First, we identified and created 22 bi-grams from n-gramming. Next, we implemented LDA and compared the topic modeling results before and after n-gramming. Then we converted the LDA posterior probabilities into word distinctivity, and we compared the new results again to show the information gain from bi-grams – the increase in word distinctivity. Finally, we compared the selected words for each topic under different versions of LDA implementations, showing that word distinctivity also improves the quality of topic modeling results.

4.1. N-Gramming Output

In the Trayvon Martin dataset, n-gramming was implemented in R using code from an existing GitHub repository [23]. The function `textToBigrams` generates bi-grams from the corpus; the cutoff frequency is set to 100; the level of statistical significance is set to 0.05.

The 22 bi-grams generated from the Trayvon Martin dataset are:

```
barack_obama, black_panther, civil_right, comment_dave
dave_surl, dont_know, dont_think, fox_news
georg_zimmerman, gregori_william, look_like, mitt_romney
neighborhood_watch, new_york, presid_obama, right_wing
self_defens, stand_ground, trayvon_martin, unit_state
white_hous, year_old
```

The bi-grams fall into three categories: people names, special phrases, and common expressions. For most bi-grams, the original form can be clearly determined, e.g. "unit_state" is originally "United States". The special phrases, such as "neighborhood_watch" and "self_defens" are the most interesting because they may not be easily identified when they were two separate words. We expect an increase in semantic information when the special phrases are regarded as single tokens.

There are few tri-grams of interest because more than 93% of the three-word phrases appear only once in the Trayvon Martin dataset. On the other hand, we found an extremely long n-gram – one blogger includes the Second Amendment to the United States Constitution[2] in every post, just like a signature.

4.2. Topic Modeling Results

We implemented LDA for topic modeling using the R package `lda` [12], with the main function `lda.collapsed.gibbs.sampler` [20, 28]. Then we list the top 10 words for each LDA-generated topic from the function `top.topic.words`. Table 1 shows the results before n-gramming. Table 2 shows the results after n-gramming, and four bi-grams are present – "fox_news", "trayvon_martin", "georg_zimmerman", and "dave_surl".

The five topic names are manually assigned from the vocabulary, and the topics are also aligned for easy cross-table comparison. The first four topics are

[2] "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."

Table 1: **Before** n-gramming: Top 10 words for each LDA-generated topic

Topic 1 General	Topic 2 Election	Topic 3 Incident	Topic 4 News Coverage	Topic 5 Gun Laws
like	obama	zimmerman	anonym	law
peopl	presid	martin	fox	gun
dont	year	said	news	ground
comment	romney	trayvon	liber	stand
get	american	polic	tommi	forc
know	democrat	georg	malkin	alec
think	said	call	msnbc	mar
right	nation	black	show	defend
make	women	prosecutor	conserv	reason
white	govern	charg	gregori	shoot

Table 2: **After** n-gramming: Top 10 words for each LDA-generated topic

Topic 1 General	Topic 2 Election	Topic 3 Incident	Topic 4 News Coverage	Topic 5 Zimmerman Trial
like	obama	zimmerman	anonym	comment
peopl	presid	martin	liber	spokesmancom
get	year	case	fox	perjuri
make	american	polic	tommi	**dave_surl**
know	romney	law	show	bond
think	republican	**trayvon_martin**	conserv	free
right	govern	said	news	access
say	law	**georg_zimmerman**	**fox_news**	view
dont	women	trayvon	msnbc	surl
see	countri	shoot	hanniti	account

the same for both tables and are explained as below:

- Topic 1 is "General" because the words are used in everyday language, such as "like", "get", "know", and "think" – obviously not a distinctive topic.
- Topic 2 is "Election", mainly due to the words "obama", "presid", and "romney", which normally appear in the 2012 US presidential election.
- Topic 3 is "Incident" due to the key words "martin" (or "trayvon_martin") and "zimmerman" (or "georg_zimmerman") for the fatal shooting incident of Trayvon Martin.
- Topic 4 is "News Coverage", because of the words "anonym" (anonymous), "fox" (Fox News), "malkin" (Michelle Malkin, an American political commentator), and "msnbc" (an American television network).

The last topic differs in the results before and after n-gramming. In Table 1, Topic 5 is "Gun Laws" because of the words "law", "gun", "ground", and

"stand" (stand your ground law), although it is a little difficult to tell. In contrast, Topic 5 in Table 2 is "Zimmerman Trial" due to the words "perjuri", "bond", and "free". Both "Gun Laws" and "Zimmerman Trial" are meaningful topics, but we can reveal one, but not both, from a single iteration of the LDA topic model.

The function `top.topic.words` in the R package `lda` [12] selects words for each topic based on the posterior probability $P(\text{ word } j | \text{ topic } i, \text{ data })$, i.e. the probability of getting word j given topic i and the data. This seems reasonable because the function returns words that are most likely to appear in each topic.

However, some words (or bi-grams) in the topics are not "distinctive" enough – these words are common across the corpus and are (unfortunately) not stop words. For example, the bi-grams "trayvon_martin" and "georg_zimmerman" belong to Topic 3 (Incident) in Table 2 because the incident is about George Zimmerman shooting Trayvon Martin. But the opposite does not hold: Upon seeing the bi-gram "georg_zimmerman", we do not know whether it came from Topic 3 (Incident), Topic 4 (News Coverage), or Topic 5 (Zimmerman Trial).

Our explanation is that the two bi-grams have low distinctivity in terms of topic selection, i.e., upon seeing them, we do not know which topic they belong to. Another example is the words in Topic 1 (General) – the words in this particular category also often appear in other topics. This presents the need of "word distinctivity", that is, given the word, how likely it is going to be in a certain topic.

Remarks

We had to manually align the topics in Tables 1 and 2 from the original LDA output, because which topic is labeled as "Topic 1" is arbitrary in the LDA output. This is a common labeling issue in finite mixture models, where the label's index has no meaning to the model itself [37]. LDA does not "know" which topic the words actually belong to; instead, LDA simply determines which words belong to the same topic.

In addition, the meaning of the bi-gram "dave_surl" is difficult to determine because we do not have the original, non-stemmed version of the Trayvon Martin corpus. We used the R package `SnowballC` [7] and found the stemmed token of the word "surveillance" to be "surveil", not "surl".

4.3. Conversion to Word Distinctivity

For word distinctivity, we used the R package `topicmodels` [21] because it is much easier to obtain the posterior probability values than from the R package

`lda`. In the R package `topicmodels`, the `posterior` function returns the posterior probabilities $P(\text{ word } j|\text{ topic } i,\text{ data })$ and $P(\text{ topic } i|\text{ document } d,\text{ data })$ for each combination of topic i, word j, and document d.

In Section 4.2, we demonstrated using the R package `lda` on purpose because the function `top.topic.words` returns the words with the highest posterior probability for each topic. This is an easy and straightforward way to obtain the LDA topic modeling results. Actually, the results from R package `topicmodels` based on posterior probability are not ideal – the same word or bi-gram can appear in more than one topic.

Tables 3 and 4 list the top 10 **distinctive** words for each topic from LDA. The former shows the results before n-gramming, while the latter shows the results after n-gramming and contains nine bi-grams. The two tables share the same five topics (in alphabetical order):

Table 3: **Before** n-gramming: Top 10 **distinctive** words for each topic from LDA

Topic 1 Election	Topic 2 Gun Laws	Topic 3 News Coverage	Topic 4 Racism	Topic 5 Zimmerman Trial
barack	alec	tommi	hoodi	spokesmancom
mitt	gun	anonym	dispatch	surl
presid	moral	tue	mar	perjuri
obama	legisl	malkin	sharpton	corey
administr	group	idiot	minut	dave
candid	individu	stupid	polic	bond
tax	weapon	gregori	martin	expert
health	ground	palin	trayvon	access
congress	violenc	rich	walk	prosecutor
romney	violat	liber	unarm	comment

Table 4: **After** n-gramming: Top 10 **distinctive** words for each topic from LDA

Topic 1 Election	Topic 2 Gun Laws	Topic 3 News Coverage	Topic 4 Racism	Topic 5 Zimmerman Trial
mitt_romney	spokesmancom	malkin	**black_panther**	**comment_dave**
barack_obama	york	palin	sharpton	**dave_surl**
alec	retreat	tue	panther	perjuri
congress	**stand_ground**	fox	young	surl
administr	access	hanniti	white	dave
econom	**neighborhood_watch**	tommi	race	bond
tax	hoodi	msnbc	black	expert
economi	sanford	dog	trayvon	comment
senat	**self_defens**	anonym	racism	voic
unit_state	florida	mitt	drug	corey

- Topic 1 is "Election" – the 2012 US presidential campaign between Barack Obama and Mitt Romney.
- Topic 2 is "Gun Laws" due to the words "gun", "moral", "legisl", "individu",

"weapon", and "violenc". This topic is about whether people are allowed to have their own guns.

- Topic 3 is "News Coverage". The words "malkin" and "gregori" refers to the American political commentators Michelle Malkin and Dick Gregory, respectively.
- Topic 4 is "Racism" mainly because of the word "sharpton". Al Sharpton is known for his engagement in civil right cases involving racism. Moreover, the word "hoodi" refers to a hoodie because Trayvon Martin was wearing a hooded sweatshirt at the time of the shooting incident.
- Topic 5 is "Zimmerman Trial" due to the words "perjuri" and "prosecutor".

In Table 4, some bi-grams make it easier to identify the topics. For instance, the bi-grams "stand_ground" (stand your ground law) and "self_defens" make it clear that the Topic 2 is about gun laws. In Topic 4, the bi-gram "black_panther" (Black Panther Party) shows discussion about racism.

The key bi-grams to the Trayvon Martin dataset, "trayvon_martin" and "georg_zimmerman", are not present in the top 10 distinctive words. The whole corpus is related to the two terms, but given these bi-grams, it is difficult to know which sub-topic they come from. As a result, they are not "distinctive" enough within the Trayvon Martin dataset.

4.4. Quantitative Comparison: Increase in Word Distinctivity

This section compares the results in Tables 3 and 4. Since "People don't ask how; they ask how much," we need to quantify the improvement of topic modeling results from n-gramming. Hence we define the "change" of word distinctivity for words as the difference between the word distinctivity before and after n-gramming. The definition of the "change" is slightly different for words (uni-grams) and for bi-grams.

4.4.1. Words (Uni-Grams)

In mathematical terms, the **change** of word distinctivity is written as

$$\max_i[P(\text{ topic } i|\text{ word } j,\text{ data after })] - \max_i[P(\text{ topic } i|\text{ word } j,\text{ data before })],$$

where "data before" refers to the data before n-gramming, and "data after" refers to the data after n-gramming.

For example, if the topic assignment vector for a word w changes from (0.8, 0.1, 0.1) to (0.05, 0.9, 0.05), the change of word distinctivity for w is 0.9 − 0.8 = 0.1. Since n-gramming increases the word distinctivity of w, this is evidence of n-gramming improving the topic modeling results.

Note that word distinctivity is defined as the maximum value of the components in the vector, so the order of the components does not matter. The word distinctivity values 0.8 and 0.9 do not have to be in the same position in the topic assignment vector.

4.4.2. Bi-Grams

Similarly, the **change** of word distinctivity for a bi-gram is written as

$$\max_i[P(\text{ topic } i|\text{ bi-gram } b,\ \text{data after})] - \max_i[P(\text{ topic } i|\text{ bi-gram } b,\ \text{data before})],$$

where the first component refers to the word distinctivity of the bi-gram b **after** n-gramming.

We need to explicitly define the second component, since the bi-gram was not formed before the n-gramming step. A bi-gram contains two words, so the word distinctivity of the bi-gram b **before** n-gramming should be the higher of the two words' distinctivity values. That is, the "baseline" of a bi-gram's word distinctivity is the highest distinctivity of the two words.

In mathematical terms, the word distinctivity of a bi-gram before n-gramming is defined as

$$\begin{aligned}&\max_i[P(\text{ topic } i|\text{ bi-gram } b,\ \text{data before })]\\&= \max\{\max_i[P(\text{ topic } i|\text{ word } 1,\ \text{data before})], \max_i[P(\text{ topic } i|\text{ word } 2,\ \text{data before})]\}.\end{aligned}$$

The bi-grams of interest would have this feature: The two words have low distinctivity, but the formed bi-gram has high distinctivity.

4.4.3. Increase in Word Distinctivity

Table 5 presents the eight bi-grams in the Trayvon Martin dataset whose word distinctivity increased at least 0.15 after n-gramming. They are listed in descending order of the increase in distinctivity. These bi-grams are of interest because they start with a low distinctivity of each word, but the bi-gram has a high distinctivity. In this way, the bi-gram almost always appears in a certain topic, so the uncertainty in topic identification decreases.

For example, the bi-gram "black_panther" is highly distinctive (90.1%) because it refers to the Black Panther Party. But if we look at the words "black" and "panther" separately, the meaning is not as clear. "Black" may refer to the color or the race, and "panther" may refer to the animal or the movie *Panther*[3].

For another example, the bi-gram "neighborhood_watch" is also highly distinctive (85.2%) because it refers to a group whose goal is to prevent crime within a neighborhood. If we break down the bi-gram, "neighborhood" and "watch" are common words and are often used in everyday English.

Table 5: The increase of word distinctivity in bi-grams

Bi-gram	Distinctivity of bi-gram	Distinctivity of word 1	Distinctivity of word 2	Increase in distinctivity
black_panther	0.901	0.484	0.525	0.376
unit_state	0.848	0.479	0.475	0.369
self_defens	0.798	0.370	0.476	0.322
neighborhood_watch	0.852	0.584	0.463	0.269
white_hous	0.793	0.354	0.528	0.265
stand_ground	0.910	0.632	0.687	0.222
year_old	0.574	0.385	0.391	0.183
new_york	0.491	0.339	0.314	0.152

4.5. Qualitative Comparison: Improvement of Topic Modeling Results

Last but not least, we also performed a qualitative comparison of the topic modeling results, and we examined the selected words for each topic under different versions of LDA implementations. The versions are determined by whether the input data were before or after n-gramming, and whether the selection criteria was the traditional posterior (Section 4.2) or the word distinctivity (Section 4.3).

The two LDA versions with word distinctivity both identified the topic "Racism" (Tables 3 and 4), while the versions using the traditional posterior did not (Tables 1 and 2). Next, we would like to present interesting results for the topics "Election" and "Gun Laws".

Table 6 compares the selected words for the topic "Election". The word "obama" appears in all four versions because Barack Obama ran for the 2012 US presidential election. Political party names such as "democrat" and "republican" appear only in the traditional posterior versions, and they are replaced with non-partisan government words in the word distinctivity versions, such as "congress", "tax", and "administr".

Although we performed n-gramming to the corpus, the bi-grams show up

[3] https://en.wikipedia.org/wiki/Panther_(film)

only when we applied the word distinctivity criteria to choose words (tokens, to be exact) for each topic. In the fourth column of Table 6, the two presidential candidate names "mitt_romney" and "barack_obama" are on top of the list. This is much more informative than the other three versions, showing that the combination of n-gramming and word distinctivity works better.

Table 6: Comparison of selected words for the topic "Election"

Before N-gramming Traditional Posterior	After N-gramming Traditional Posterior	Before N-gramming Word Distinctivity	After N-gramming Word Distinctivity
obama	obama	barack	**mitt_romney**
presid	presid	mitt	**barack_obama**
year	year	presid	alec
romney	american	obama	congress
american	romney	administr	administr
democrat	republican	candid	econom
said	govern	tax	tax
nation	law	health	economi
women	women	congress	senat
govern	countri	romney	**unit_state**

Table 7 also attempts to compare the selected words for the topic "Gun Laws", but this topic does not exist in the version of after n-gramming and traditional posterior. That is, when using the traditional posterior probability as the selection criteria, we could reveal the topic "Gun Laws" before n-gramming, but could not do so after n-gramming. Performing n-gramming is expected to result in information gain of the topic modeling results, but we did not see the desired outcome.

On the contrary, the combination of n-gramming and word distinctivity works well in Table 7. Three informative bi-grams appear: "stand_ground" (stand your ground law), "neighborhood_watch", and "self_defens". It is much easier to infer the context from the bi-gram "stand_ground" than from the two separate words "stand" and "ground".

Table 7: Comparison of selected words for the topic "Gun Laws"

Before N-gramming Traditional Posterior	After N-gramming Traditional Posterior	Before N-gramming Word Distinctivity	After N-gramming Word Distinctivity
law		alec	spokesmancom
gun		gun	york
ground		moral	retreat
stand		legisl	**stand_ground**
forc	N/A	group	access
alec		individu	**neighborhood_watch**
mar		weapon	hoodi
defend		ground	sanford
reason		violenc	**self_defens**
shoot		violet	florida

In short, n-gramming improves the topic modeling results, but it is difficult to show the improvement without using word distinctivity as the selection criteria

of `top.topic.words`.

4.6. Limitations

We can safely assume that topics with more highly distinctive tokens are better defined, but we are unable to prove or disprove whether n-gramming increases our ability to correctly guess the document's topic proportions, given that the Trayvon Martin dataset does not contain the ground truth. Existing literature [33] also shows that evaluating an unsupervised model is difficult. However, it is possible to create a synthetic dataset with pre-defined topic proportions from Wikipedia articles [11], then we can use the new dataset to test the hypothesis.

5. DISCUSSION

Adequate text data preprocessing helps in topic modeling, and the improvement of results from n-gramming can be quantified by word distinctivity. After we identify and combine the words with special meaning into bi-grams, more semantic information is retained, leading to a stronger signal in topic classification.

In this way, the text data cleaning quality can be measured in terms of topic modeling results at the word level. By retaining special phrases (i.e., bi-grams), n-gramming increases the word distinctivity, and word distinctivity improves the quality of the LDA-identified topics. Some bi-grams have a higher distinctivity than either of the two word components, so the signal of topic assignment is stronger after the bi-gram is formed.

On the other hand, the effect of n-gramming at the corpus level is still unclear, since bi-grams account for only a small part of a text database [2]. We attempted to measure the prediction power from the corpus after n-gramming, and we used "perplexity" as a single number to summarize how well the topic model predicts the remaining words, given a part of the document [4]. The perplexity is the effective number of equally likely words based on the model, so the perplexity is inversely proportional to the precision of the predictive model output. Nevertheless, the t-test results were inconclusive [11]. Therefore, more research is needed to evaluate the overall improvement of topic modeling results after n-gramming.

6. FUTURE DIRECTIONS

This research is a start of quantifying topic model performance, and we hope to further improve the text data cleaning process and statistically evaluate the results. We quantified the increase of word distinctivity from n-gramming in Section 4.4, and we are still looking for a metric to numerically measure the quality of selected topic words. Section 4.5 gives a preliminary and qualitative comparison to show evidence that the word distinctivity is a better selection criteria than the traditional posterior.

Moreover, a potential solution to the LDA labeling issue (Section 4.2) is seeded topic models [46]. The seeded topic model preassigns each topic with a word, then the model "grows" each topic from the preassigned word. An example is to start the first topic (Election) with "barack_obama", the second topic (Gun Law) with "self_defens", and the third topic (Racism) with "sharpton" (Al Sharpton). Other deterministic relabeling strategies are described in [37] and the R package `label.switching` [36].

A new possible direction is to compare various existing methods and determine which method is most appropriate for which type of text corpus. We used a list of predefined stop words to remove the words with little semantic meaning in the corpus, but there may exist better ways to perform text data cleaning.

Another possible extension is to combine the n-gram construction and the LDA into a single Bayesian hierarchical model. By introducing additional latent variables to determine the probability of a phrase to be an n-gram, we can eliminate the frequentist approach of testing for n-grams.

ACKNOWLEDGMENTS

The author would like to thank her PhD advisor at Duke University, Dr. David Banks, for his support on this research project. The author would also like to thank the people who also worked on this dataset, Derek Owens-Oas and Teague Rhine Henry, for the discussion. The author is also grateful for the comments from Andrew Raim (U.S. Census Bureau) and the anonymous reviewers.

REFERENCES

[1] Timothy C Au. *Topics in Computational Advertising*. PhD dissertation, Duke University, 2014. 10

[2] Ron Bekkerman and James Allan. Using bigrams in text categorization. Technical report, IR-408, Center of Intelligent Information Retrieval, University of Massachusetts Amherst, 2004. 21

[3] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: Analyzing text with the natural language toolkit*. O'Reilly Media, Inc., 2009. 3

[4] David Blei and John Lafferty. Correlated topic models. *Advances in Neural Information Processing Systems*, 18:147, 2006. 21

[5] David M Blei and John D Lafferty. Visualizing topics with multi-word expressions. *arXiv preprint arXiv:0907.1013*, 2009. 4, 7

[6] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003. 4, 11

[7] Milan Bouchet-Valat. *SnowballC: Snowball stemmers based on the C libstemmer UTF-8 library*, 2014. R package version 0.5.1. 5, 15

[8] Andrei Z Broder, Steven C Glassman, Mark S Manasse, and Geoffrey Zweig. Syntactic clustering of the web. *Computer Networks and ISDN Systems*, 29(8-13):1157–1166, 1997. ISDN stands for Integrated Services Digital Network. 7

[9] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992. 7, 8

[10] William B Cavnar and John M Trenkle. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, volume 161175. Citeseer, 1994. 7

[11] Christine P Chai. *Statistical Issues in Quantifying Text Mining Performance*. PhD dissertation, Duke University, 2017. 6, 21

[12] Jonathan Chang. *lda: Collapsed Gibbs sampling methods for topic models*, 2015. R package version 1.4.2. 13, 15

[13] Nick Chater and Christopher D Manning. Probabilistic models of language processing and acquisition. *Trends in Cognitive Sciences*, 10(7):335–344, 2006. 8

[14] CrowdFlower. 2016 data science report. http://visit.crowdflower.com/rs/416-ZBE-142/images/CrowdFlower_DataScienceReport_2016.pdf, 2016. 3

[15] Shyamala Doraisamy and Stefan Rüger. Robust polyphonic music retrieval with n-grams. *Journal of Intelligent Information Systems*, 21(1):53–70, 2003. 4

[16] Matthias Eck, Stephan Vogel, and Alex Waibel. Low cost portability for statistical machine translation based on n-gram frequency and tf-idf. In *International Workshop on Spoken Language Translation (IWSLT) 2005*, 2005. 8

[17] Ronen Feldman and James Sanger. *The text mining handbook: Advanced approaches in analyzing unstructured data*. Cambridge University Press, 2007. 3

[18] Louise Francis and Matt Flynn. *Text mining handbook*. Casualty Actuarial Society E-Forum, http://www.casact.org/pubs/forum/10spforum/completes10.pdf, 2010. 3

[19] Johannes Fürnkranz. A study using n-gram features for text categorization. *Austrian Research Institute for Artificial Intelligence*, 3(1998):1–10, 1998. 7

[20] Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101:5228–5235, 2004. 11, 13

[21] Bettina Grün and Kurt Hornik. topicmodels: An R package for fitting topic models. *Journal of Statistical Software*, 40(13):1–30, 2011. 15

[22] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: Concepts and techniques*. Elsevier, 2011. 3

[23] Teague Henry. Quick ngram processing script. https://github.com/trh3/NGramProcessing, 2016. 13

[24] Teague Henry, David Banks, Derek Owens-Oas, and Christine Chai. Modeling community structure and topics in dynamic text networks. *Journal of Classification, arXiv:1610.05756*, 2018. 5

[25] Vera Hollink, Jaap Kamps, Christof Monz, and Maarten De Rijke. Monolingual document retrieval for european languages. *Information retrieval*, 7(1-2):33–52, 2004. 4

[26] Jaap Kamps, Christof Monz, Maarten De Rijke, and Börkur Sigurbjörnsson. Language-dependent and language-independent approaches to cross-lingual text retrieval. In *Workshop of the Cross-Language Evaluation Forum for European Languages*, pages 152–165. Springer, 2003. 4

[27] Chenghua Lin and Yulan He. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 375–384. ACM, 2009. 5

[28] Jun S Liu. The collapsed Gibbs sampler in Bayesian computations with applications to a gene regulation problem. *Journal of the American Statistical Association*, 89(427):958–966, 1994. 13

[29] José B Marino, Rafael E Banchs, Josep M Crego, Adrià de Gispert, Patrik Lambert, José AR Fonollosa, and Marta R Costa-Jussà. N-gram-based machine translation. *Computational Linguistics*, 32(4):527–549, 2006. 4

[30] Paul McNamee and James Mayfield. Jhu/apl experiments in tokenization and non-word translation. In *Comparative Evaluation of Multilingual Information Access Systems*, pages 85–97. Springer, 2003. 5

[31] Microsoft. Extract n-gram features from text. https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/extract-n-gram-features-from-text, 2019. 8

[32] Thomas R Niesler and Philip C Woodland. A variable-length category-based n-gram language model. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, volume 1, pages 164–167. IEEE, 1996. 4

[33] Bruno M Nogueira, Maria F Moura, M da S Conrado, Rafael G Rossi, Ricardo M Marcacini, and Solange O Rezende. Winning some of the document preprocessing challenges in a text mining process. In *Embrapa Informática Agropecuária-Artigo em anais de congresso (ALICE)*. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, 23.; SIMPÓSIO BRASILEIRO DE . . . , 2008. 21

[34] Jason W Osborne. *Best practices in data cleaning: A complete guide to everything you need to do before and after collecting your data*. Sage Publications, 2012. 3

[35] Derek Owens-Oas. *Probabilistic Models for Text in Social Networks*. PhD dissertation, Duke University, 2018. 5

[36] Panagiotis Papastamoulis. label.switching: An R package for dealing with the label switching problem in mcmc outputs. *arXiv preprint arXiv:1503.02271*, 2015. 22

[37] Carlos E Rodriguez and Stephen G Walker. Label switching in Bayesian mixture models: Deterministic relabeling strategies. *Journal of Computational and Graphical Statistics*, 23(1):25–45, 2014. 15, 22

[38] Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. The author-topic model for authors and documents. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 487–494. AUAI Press, 2004. 5

[39] Howard Schuman and Stanley Presser. *Questions and answers in attitude surveys: Experiments on question form, wording, and context*. Sage, 1996. 3

[40] Julia Silge and David Robinson. *Text mining with* R*: A tidy approach*. O'Reilly Media, Inc., 2017. 3

[41] Simply Statistics. On the relative importance of mathematical abstraction in graduate statistical education. https://simplystatistics.org/2012/08/08/on-the-relative-importance-of-mathematical-abstraction/, 2012. 3

[42] Jacopo Soriano, Timothy Au, and David Banks. Text mining in computational advertising. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 6(4):273–285, 2013. 6, 10

[43] Megan Squire. *Clean Data*. Packt Publishing Ltd, 2015. 3

[44] Statistical Services Centre. Approaches to the analysis of survey data. Technical report, The University of Reading, United Kingdom, 2001. 3

[45] Rebecca C Steorts, Samuel L Ventura, Mauricio Sadinle, and Stephen E Fienberg. A comparison of blocking methods for record linkage. In *International Conference on Privacy in Statistical Databases*, pages 253–268. Springer, 2014. 7

[46] Matthew Stephens. Dealing with label switching in mixture models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(4):795–809, 2000. 22

[47] Ching Y Suen. N-gram statistics for natural language understanding and text processing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):164–172, 1979. 4

[48] SurveyMonkey. How to analyze survey data. https://www.surveymonkey.com/mp/how-to-analyze-survey-data/, n.d. Accessed in 2018. 3

[49] Tommi Vatanen, Jaakko J Väyrynen, and Sami Virpioja. Language identification of short text segments with n-gram models. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC)*, 2010. 7

[50] Hanna M Wallach. Topic modeling: Beyond bag-of-words. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 977–984. ACM, 2006. 3, 4

[51] Zhihua Wei, Duoqian Miao, Jean-Hugues Chauchat, Rui Zhao, and Wen Li. N-grams based feature selection and text representation for Chinese text classification. *International Journal of Computational Intelligence Systems*, 2(4):365–374, 2009. 8

[52] Wayne Xiong, Lingfeng Wu, Fil Alleva, Jasha Droppo, Xuedong Huang, and Andreas Stolcke. The Microsoft 2017 conversational speech recognition system. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5934–5938. IEEE, 2018. 4

[53] Richard Zens and Hermann Ney. N-gram posterior probabilities for statistical machine translation. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 72–77, 2006. 4

[54] Shichao Zhang, Chengqi Zhang, and Qiang Yang. Data preparation for data mining. *Applied Artificial Intelligence*, 17(5-6):375–381, 2003. 3

[55] Yin Zhang, Rong Jin, and Zhi-Hua Zhou. Understanding bag-of-words model: A statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1-4):43–52, 2010. 3