Bag of What? Simple Noun Phrase Extraction for Text Analysis

Abram Handler
UMass Amherst
ahandler@cs.umass.edu

Matthew J. Denny
Penn State
matthewjdenny@gmail.com

Hanna Wallach
Microsoft Research
hanna@dirichlet.net

Brendan O'Connor
UMass Amherst
brenocon@cs.umass.edu

Abstract

Social scientists who do not have specialized natural language processing training often use a unigram bag-of-words (BOW) representation when analyzing text corpora. We offer a new phrase-based method, **NPFST**, for enriching a unigram BOW. NPFST uses a part-of-speech tagger and a finite state transducer to extract multiword phrases to be added to a unigram BOW. We compare NPFST to both n-gram and parsing methods in terms of yield, recall, and efficiency. We then demonstrate how to use NPFST for exploratory analyses; it performs well, without configuration, on many different kinds of English text. Finally, we present a case study using NPFST to analyze a new corpus of U.S. congressional bills.

For our open-source implementation, see http://slanglab.cs.umass.edu/phrases/.

1 Introduction

Social scientists typically use a unigram representation when analyzing text corpora; each document is represented as a unigram bag-of-words (BOW), while the corpus itself is represented as a document–term matrix of counts. For example, Quinn et al. (2010) and Grimmer (2010) used a unigram BOW as input to a topic model, while Monroe et al. (2008) used a unigram BOW to report the most partisan terms from political speeches. Although the simplicity of a unigram BOW is appealing, unigram analyses do not preserve meaningful multiword phrases, such as "health care" or "social security," and cannot distinguish between politically significant phrases that share a word, such as "illegal immigrant" and "undocumented immigrant." To address these limitations, we introduce **NPFST**, which extracts multiword phrases to enrich a unigram BOW as additional columns in the document–term matrix. NPFST is suitable for many different kinds of English text; it uses modest computational resources and does not require any specialized configuration or annotations.

2 Background

We compare NPFST to several other methods in terms of yield, recall, efficiency, and interpretability. Yield refers to the number of extracted phrases—a lower yield requires fewer computational and human resources to process the phrases. Recall refers to a method's ability to recover the most relevant or important phrases, as determined by a human. A good method should have a low yield, but high recall.

2.1 n-grams

Our simplest baseline is **AllNGrams(K)**. This method extracts all n-grams, up to length K, from tokenized, sentence-segmented text, excluding n-grams that cross sentence boundaries. This method is commonly used to extract features for text classification (e.g., Yogatama et al. (2015)), but has several disadvantages in a social scientific context. First, social scientists often want to substantively interpret individual phrases, but fragmentary phrases that cross sentence constituents may not be meaningful. For example, the Affordable Care Act includes the hard-to-interpret 4-gram, "the Internet website of." Second, although AllNGrams(K) has high recall (provided that K is sufficiently large), it suffers from a higher yield and can therefore require substantial resources to process the extracted phrases.

2.2 Parsing

An alternative approach[1] is to use syntax to restrict the extracted phrases to constituents, such as noun phrases (NPs). Unlike verb, prepositional,

[1] Statistical collocation methods provide another approach (e.g., Dunning (1993), Hannah and Wallach (2014)). These methods focus on within-n-gram statistical dependence. In informal analyses, we found their recall unsatisfying for low-frequency phrases, but defer a full comparison for future work.

Proceedings of 2016 EMNLP Workshop on Natural Language Processing and Computational Social Science, pages 114–124,
Austin, TX, November 5, 2016.

or adjectival phrases, NPs often make sense even when stripped from their surrounding context—e.g., *[Barack Obama]*$_{NP}$ vs. *[was inaugurated in 2008]*$_{VP}$. There are many methods for extracting NPs. Given the long history of constituent parsing research in NLP, one obvious approach is to run an off-the-shelf constituent parser and then retrieve all NP non-terminals from the trees.[2] We refer to this method as **ConstitParse**. Unfortunately, the major sources of English training data, such as the Penn Treebank (Marcus et al., 1993), include determiners within the NP and non-nested flat NP annotations,[3] leading to low recall in our context (see §4). Since modern parsers rely on these sources of training data, it is very difficult to change this behavior.

2.3 Part-of-Speech Grammars

Another approach, proposed by Justeson and Katz (1995), is to use part-of-speech (POS) patterns to find and extract NPs—a form of shallow partial parsing (Abney, 1997). Researchers have used this approach in a variety of different contexts (Benoit and Nulty, 2015; Frantzi et al., 2000; Kim et al., 2010; Chuang et al., 2012; Bamman and Smith, 2014). A pattern-based method can be specified in terms of a triple of parameters: (G, K, M), where G is a grammar, K is a maximum length, and M is a matching strategy. The grammar G is a non-recursive regular expression that defines an infinite set of POS tag sequences (i.e., a regular language); the maximum length K limits the length of the extracted n-grams to $n \leq K$; while the matching strategy M specifies how to extract text spans that match the grammar.

The simplest grammar that we consider is

$$(A \mid N) * N(PD * (A \mid N) * N)*$$

defined over a coarse tag set of adjectives, nouns (both common and proper), prepositions, and determiners. We refer to this grammar as SimpleNP. The constituents that match this grammar are bare NPs (with optional PP attachments), N-bars, and names. We do not include any determiners at the root NP.

[2]Another type of syntactic structure prediction is NP chunking. This produces a shallower, non-nested representation.

[3]The English Web Treebank (LDC2012T13) has some more nesting structure and OntoNotes (version 5, LDC2013T19) includes a variant of the Penn Treebank with Vadas and Curran (2011)'s nested NP annotations. We look forward to the availability of constituent parsers trained on these data sources.

We also consider three baseline matching strategies, each of which can (in theory) be used with any G and K. The first, FilterEnum, enumerates all possible strings in the regular language, up to length K, as a preprocessing step. Then, at runtime, it checks whether each n-gram in the corpus is present in this enumeration. This matching strategy is simple to implement and extracts all matches up to length K, but it is computationally infeasible if K is large. The second, FilterFSA, compiles G into a finite-state automaton (FSA) as a preprocessing step. Then, at runtime, it checks whether each n-gram matches this FSA. Like FilterEnum, this matching strategy extracts all matches up to length K; however, it can be inefficient if K is large. The third, GreedyFSA, also compiles G into an FSA, but uses a standard greedy matching approach at runtime to extract n-grams that match G. Unlike the other two matching strategies, it cannot extract overlapping or nested matches, but it can extract very long matches.[4]

In their original presentation, Justeson and Katz (1995) defined a grammar that is very similar to SimpleNP and suggested using 2- and 3-grams (i.e., $K = 3$). With this restriction, their grammar comprises seven unique patterns. They also proposed using FilterEnum to extract text spans that match these patterns. We refer to this method as **JK** = (SimpleNP, $K = 3$, FilterEnum). Many researchers have used this method, perhaps because it is described in the NLP textbook by Manning and Schütze (1999).

3 NPFST

Our contribution is a new pattern-based extraction method: **NPFST** = (FullNP, $K = \infty$, RewriteFST). In §3.1, we define the FullNP grammar, and in §3.2, we define the RewriteFST matching strategy.

3.1 FullNP Grammar

FullNP extends SimpleNP by adding coordination of pairs of words with the same tag (e.g., *(VB CC VB)* in *(cease and desist) order*); coordination of noun phrases; parenthetical post-modifiers (e.g., *401(k)*, which is a 4-gram because of common NLP tokenization conventions); numeric modifiers and nominals; and support for the Penn Treebank tag set,

[4]We implemented both FilterFSA and GreedyFSA using standard Python libraries—specifically, *re.match* and *re.finditer*.

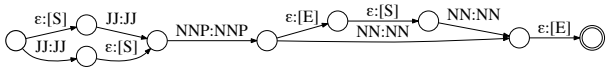


Figure 1: Composed rewrite lattice $L = I \circ P$ for input $I =$ (JJ NNP NN). Five spans are retrieved during lattice traversal.

the coarse universal tag set (Petrov et al., 2011), and Gimpel et al. (2011)'s Twitter-specific coarse tag set. We provide the complete definition in the appendix.

3.2 RewriteFST Matching Strategy

RewriteFST uses a finite-state transducer (FST) to rapidly extract text spans that match G—including overlapping and nested spans. This matching strategy is a form of finite-state NLP (Roche and Schabes, 1997), and therefore builds on an extensive body of previous work on FST algorithms and tools.

The input to RewriteFST is a POS-tagged[5] sequence of tokens I, represented as an FSA. For a simple tag sequence, this FSA is a linear chain, but, if there is uncertainty in the output of the tagger, it can be a lattice with multiple tags for each position.

The grammar G is first compiled into a phrase transducer P,[6] which takes an input sequence I and outputs the same sequence, but with pairs of start and end symbols—[S] and [E], respectively—inserted to indicate possible NPs (see figure 1). At runtime, RewriteFST computes an output lattice $L = I \circ P$ using FST composition;[7] since it is nondeterministic, L includes all overlapping and nested spans, rather than just the longest match. Finally, FilterFST traverses L to find all edges with a [S] symbol. From each one, it performs a depth-first search to find all paths to an edge with an [E] symbol, accumulating all [S]- and [E]-delimited spans.[8]

In table 1, we provide a comparison of FilterFST and the three matching strategies described in §2.3.

[5]We used the ARK POS tagger for tweets (Gimpel et al., 2011; Owoputi et al., 2013) and used Stanford CoreNLP for all other corpora (Toutanova et al., 2003; Manning et al., 2014).

[6]We used `foma` (Hulden, 2009; Beesley and Karttunen, 2003) to compile G into P. `foma` was designed for building morphological analyzers; it allows a developer to write a grammar in terms of readable production rules with intermediate categories. The rules are then compiled into a single, compact FST.

[7]We implemented the FST composition using *OpenNLP* (Allauzen et al., 2007) and *pyfst* (http://pyfst.github.io/).

[8]There are alternatives to this FST approach, such as a backtracking algorithm applied directly to the original grammar's FSA to retrieve all spans starting at each position in the input.

Matching Strategy	All Matches?	Large K?
FilterEnum	yes	infeasible
FilterFSA	yes	can be inefficient
GreedyFSA	no	yes
RewriteFST	yes	yes

Table 1: RewriteFST versus the matching strategies described in §2.3. Like FilterEnum and FilterFSA, RewriteFST extracts all matches up to length K; in contrast, GreedyFSA cannot extract overlapping or nested matches. Like GreedyFSA, RewriteFST can extract long matches; in contrast, FilterEnum and is infeasible and FilterFSA can be inefficient if K is large.

4 Experimental Results

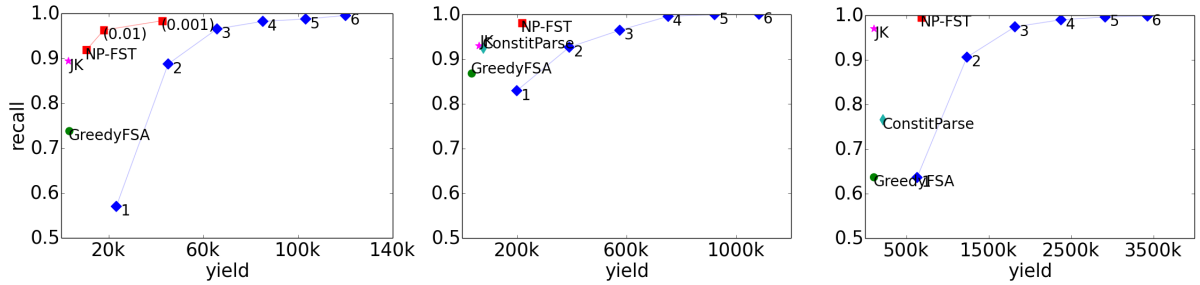
In this section, we provide experimental results comparing NPFST to the baselines described in §2 in terms of yield, recall, efficiency, and interpretability. As desired, NPFST has a low yield and high recall, and efficiently extracts highly interpretable phrases.

4.1 Yield and Recall

Yield refers to the number of phrases extracted by a method, while recall refers to a method's ability to recover the most relevant or important phrases, as determined by a human. Because relevance and importance are domain-specific concepts that are not easy to define, we compared the methods using three named-entity recognition (NER) data sets: mentions of ten types of entities on Twitter from the WNUT 2015 shared task (Baldwin et al., 2015); mentions of proteins in biomedical articles from the BioNLP shared task 2011 (Kim et al., 2011); and a synthetic data set of named entities in New York Times articles (Sandhaus, 2008), identified using Stanford NER (Manning et al., 2014). Named entities are undoubtedly relevant and important phrases in all three of these different domains.[9] For each data set, we defined a method's yield to be the total number of spans that it extracted and a method's recall to be the percentage of the (labeled) named entity spans that were present in its list of extracted spans.[10]

[9]Although we use NER data sets to compare the methods' yield and recall, social scientists are obviously interested in analyzing other phrases, such as "heath care reform," which have a less psycholinguistically concrete status (Brysbaert et al., 2014). We focus on these kinds of phrases in §4.3 and §5.

[10]We assumed that all methods extracted all unigram spans.



(a) WNUT (b) BioNLP (c) NYT

Figure 2: Recall versus yield for AllNGrams(K) with $K = 1, \ldots, 6$, ConstitParse, JK, (SimpleNP, $K = \infty$, GreedyFSA), and NPFST. A good method should have a low yield, but high recall—i.e., the best methods are in the top-left corner of each plot. For visual clarity, the y-axis starts at 0.5. We omit yield and recall values for AllNGrams(K) with $K > 6$ because recall approaches an asymptote. For the WNUT data set, we omit yield and recall values for ConstitParse because there is no reliable constituent parser for tweets. As described in §4.1, we also show yield and recall values for NPFST run on input lattices (denoted by 0.01 and 0.001).

Figure 2 depicts recall versus yield[11] for NPFST and the following baseline methods: AllNGrams(K) with different values of K, ConstitParse,[12] JK, and (SimpleNP, $K = \infty$, GreedyFSA). Because the yield and recall values for (SimpleNP, $K = 3$, FilterFSA) are the same as those of JK, we omit these values from figure 2. We also omit yield and recall values for (FullNP, $K = \infty$, FilterEnum) and (FullNP, $K = \infty$, FilterFSA) because they are identical to those of NPFST. Finally, we omit yield and recall values for (FullNP, $K = \infty$, GreedyFSA) because our implementation of GreedyFSA (using standard Python libraries) is too slow to use with FullNP.

A good method should have a low yield, but high recall—i.e., the best methods are in the top-left corner of each plot. The pattern-based methods all achieved high recall, with a considerably lower yield than AllNGrams(K). ConstitParse achieved a lower yield than NPFST, but also achieved lower recall. JK performed worse than NPFST, in part because it can only extract 2- and 3-grams, and, for example, the BioNLP data set contains mentions of proteins that are as long as eleven tokens (e.g., "Ca2+/calmodulin-dependent protein kinase (CaMK) type IV/Gr"). Finally, (SimpleNP, $K = \infty$, GreedyFSA) performed much worse than JK because it cannot extract overlapping or nested spans.

[11]The WNUT data set is already tokenized; however, we accidentally re-tokenized it in our experiments. Figure 2 therefore only depicts yield and recall for the 1,278 (out of 1,795) tweets for which our re-tokenization matched the original tokenization.

[12]We used the Stanford CoreNLP shift–reduce parser.

Method	Time
AllNGrams(∞)	44.4 ms
ConstitParse	825.3 ms
JK	45.3 ms
(SimpleNP, $K = 3$, FilterFSA)	46.43 ms
(SimpleNP, $K = \infty$, GreedyFSA)	39.34 ms
NPFST	82.2 ms

Table 2: Timing results for AllNGrams(∞), ConstitParse, JK, (SimpleNP, $K = 3$, FilterFSA), (SimpleNP, $K = \infty$, GreedyFSA), and NPFST on ten articles from the BioNLP data set; AllNGrams(∞) is equivalent to AllNGrams(56) in this context. The pattern-based methods' times include POS tagging (37.1 ms), while ConstitParse's time includes parsing (748 ms).

For the WNUT data set, NPFST's recall was relatively low (91.8%). To test whether some of its false negatives were due to POS-tagging errors, we used NPFST's ability to operate on an input lattice with multiple tags for each position. Specifically, we constructed an input lattice I using the tags for each position whose posterior probability was at least t. We experimented with $t = 0.01$ and $t = 0.001$. These values increased recall to 96.2% and 98.3%, respectively, in exchange for only a slightly higher yield (lower than that of AllNGrams(2)). We suspect that we did not see a greater increase in yield, even for $t = 0.001$, because of posterior calibration (Nguyen and O'Connor, 2015; Kuleshov and Liang, 2015).

4.2 Efficiency

We used ten articles from the BioNLP data set to compare the methods' preprocessing and runtime costs. Table 2 contains timing results[13] for AllNGrams(∞), ConstitParse, JK, (SimpleNP, $K = 3$, FilterFSA), and (SimpleNP, $K = \infty$, GreedyFSA), and NPFST. We omit results for (FullNP, $K = \infty$, FilterEnum), (FullNP, $K = \infty$, FilterFSA), and (FullNP, $K = \infty$, GreedyFSA) because they are too slow to compete with the other methods.

POS tagging is about twenty times faster than parsing, which is helpful for social scientists who may not have fast servers. NPFST is slightly slower than the simpler pattern-based methods; however, 80% of its time is spent constructing the input I and traversing the output lattice L, both of which are implemented in Python and could be made faster.

4.3 Interpretability

When analyzing text corpora, social scientists often examine ranked lists of terms, where each term is ranked according to some score. We argue that multiword phrases are more interpretable than unigrams when stripped from their surrounding context and presented as a list. In §4.3.1 we explain how to merge related terms, and in §4.3.2, we provide ranked lists that demonstrate that NPFST extracts more interpretable phrases than other methods.

4.3.1 Merging Related Terms

As described in §3.2, NPFST extracts overlapping and nested spans. For example, when run on a data set of congressional bills about crime, NPFST extracted "omnibus crime control and safe streets act," as well as the nested phrases "crime control" and "safe streets act." Although this behavior is generally desirable, it can also lead to repetition in ranked lists. We therefore outline an high-level algorithm for merging the highest-ranked terms in a ranked list.

The input to our algorithm is a list of terms L. The algorithm iterates through the list, starting with the highest-ranked term, aggregating similar terms according to some user-defined criterion (e.g., whether the terms share a substring) until it has generated C distinct term clusters. The algorithm then selects a single term to represent each cluster. Finally, the algorithm orders the clusters' representative terms to form a ranked list of length C. By starting with the highest-ranked term and terminating after C clusters have been formed, this algorithm avoids the inefficiency of examining all possible pairs of terms.

4.3.2 Ranked Lists

To assess the interpretability of the phrases extracted by NPFST, we used three data sets: tweets about climate change, written by (manually identified) climate deniers;[14] transcripts from criminal trials at the Old Bailey in London during the 18th century;[15] and New York Times articles from September, 1993. For each data set, we extracted phrases using ConstitParse, JK, and NPFST and produced a list of terms for each method, ranked by count. We excluded domain-specific stopwords and any phrases that contained them.[16] Finally, we merged related terms using our term-merging algorithm, aggregating terms only if one term was a substring of another, to produce ranked lists of five representative terms. Table 4.3 contains these lists, demonstrating that NPFST produces highly interpretable phrases.

5 Case Study: Finding Partisan Terms in U.S. Congressional Legislation

Many political scientists have studied the relationship between language usage and party affiliation (Laver et al., 2003; Monroe et al., 2008; Slapin and Proksch, 2008; Quinn et al., 2010; Grimmer and Stewart, 2013). We present a case study, in which we use NPFST to explore partisan differences in U.S. congressional legislation about law and crime. In §5.1, we describe our data set, and in §5.2, we explain our methodology and present our results.

5.1 The Congressional Bills Corpus

We used a new data set of 97,221 U.S. congressional bills, introduced in the House and Senate between

[13] We used Python's *timeit* module.

[14] https://www.crowdflower.com/data/sentiment-analysis-global-warmingclimate-change-2/

[15] http://www.oldbaileyonline.org/

[16] For example, for the tweets, we excluded phrases that contained "climate" and "warming." For the Old Bailey transcripts, we excluded phrases that contained "st." or "mr." (e.g., "st. john" or "mr. white"). We also used a regular expression to exclude apparent abbreviated names (e.g., "b. smith") and used a stopword list to exclude dates like "5 of february." For the New York Times articles, we excluded phrases that contained "said."

Data Set	Method	Ranked List
Twitter	unigrams	snow, #tcot, al, dc, gore
	JK	al gore's, snake oil science, snow in dc, mine safety
	NPFST	al gore's, snake oil science, 15 months, snow in dc, *bunch of snake oil science
Old Bailey	unigrams	jacques, goodridge, rust, prisoner, sawtell
	ConsitParse	the prisoner, the warden, the draught, the fleet, the house
	JK	middlesex jury, public house, warrant of attorney, baron perryn, justice grose
	NPFST	middlesex jury, public house, warrant of attorney, baron perryn, *middlesex jury before lord loughborough
NYT	unigrams	will, united, one, government, new
	ConstitParse	he united states, the government, the agreement, the president, the white house
	JK	united states, united nations, white house, health care, prime minister
	NPFST	united states, united nations, white house, health care, *secretary of state warren christopher

Table 3: Ranked lists of representative terms for unigrams, ConstitParse, JK, and NPFST. For NPSFT, we include the highest-ranked phrase of length four or more (on its own line, denoted by *) in order to highlight the kinds of longer phrases that JK is unable to extract. For the Twitter data set, we omit results for ConstitParse because there is no reliable constituent parser for tweets.

1993 and 2014. We created this data set by scraping the Library of Congress website.[17] We used Stanford CoreNLP to tokenize and POS tag the bills. We removed numbers and punctuation, and discarded all terms that occurred in fewer than five bills. We also augmented each bill with its author, its final outcome (e.g., did it survive committee deliberations, did it pass a floor vote in the Senate) from the Congressional Bills Project (Adler and Wilkerson, 2014), and its major topic area (Purpura and Hillard, 2006).

For our case study, we focused on a subset of 488 bills, introduced between 2013 and 2014, that are primarily about law and crime. We chose this subset because we anticipated that it would clearly highlight partisan policy differences. For example, the bills include legislation about immigration enforcement and about incarceration of low-level offenders—two areas where Democrats and Republicans tend to have very different policy preferences.

5.2 Partisan Terms

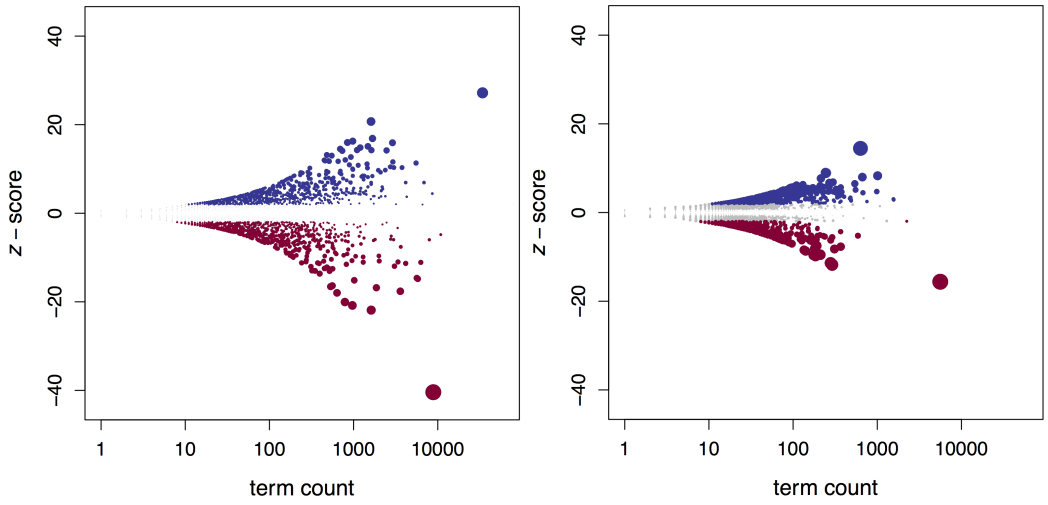
We used NPFST to extract phrases from the bills, and then created ranked lists of terms for each party using the informative Dirichlet[18] feature selection method of Monroe et al. (2008). This method computes a z-score for each term that reflects how strongly that term is associated with Democrats over Republicans—a positive z-score indicates that Democrats are more likely to use the term, while a negative z-score indicates that Republications are more likely to use the term. We merged the highest-ranked terms for each party, aggregating terms only if one term was a substring of another and if the terms were very likely to co-occur in a single bill,[19] to form ranked lists of representative terms. Finally, for comparison, we also used the same approach to create ranked lists of unigrams, one for each party.

Figure 3 depicts z-score versus term count, while table 4 lists the twenty highest-ranked terms. The unigram lists suggest that Democratic lawmakers focus more on legislation related to mental health, juvenile offenders, and possibly domestic violence, while Republican lawmakers focus more on illegal immigration. However, many of the highest-ranked unigrams are highly ambiguous when stripped from their surrounding context. For example, we do not know whether "domestic" refers to "domestic violence," "domestic terrorism," or "domestic programs" without manually reviewing the origi-

[17] http://congress.gov/

[18] In order to lower the z-scores of uninformative, high-frequency terms, we set the Dirichlet hyperparameters to be proportional to the term counts from our full data set of bills.

[19] We used the correlation between the terms' tf-idf vectors determine how likely the terms were to co-occur in a single bill.



(a) unigrams (b) NPFST

Figure 3: z-score versus term count. Each dot represents a single term and is sized according to that term's z-score. Terms that are more likely to be used by Democrats are shown in blue; terms that are more likely to be used by Republicans are shown in dark red.

nal bills (e.g., using a keyword-in-context interface (O'Connor, 2014)). Moreover, many of the highest-ranked Republican unigrams, such as "communication," are not unique to law and crime.

In contrast, the phrase-based lists are less ambiguous and much more interpretable. They include names of bills (which are often long) and important concepts, such as "mental health," "victims of domestic violence," "interstate or foreign commerce," and "explosive materials." These lists suggest that Democratic lawmakers have a very strong focus on programs to prevent child abuse and domestic violence, as well as issues related to mental health and gang violence. Republican lawmakers appear to focus on immigration and incarceration. This focus on immigration is not surprising given the media coverage between 2013 and 2014; however, there was much less media coverage of a Democratic focus on crime-related legislation during that time period.

These results suggest that social scientists will be less likely to draw incorrect conclusions from ranked lists of terms if they include multiword phrases. Because phrases are less ambiguous than unigrams, social scientists can more quickly discover meaningful term-based associations for further exploration, without undertaking a lengthy process to validate their interpretation of the terms.

6 Conclusions and Future Work

Social scientists typically use a unigram BOW representation when analyzing text corpora, even though unigram analyses do not preserve meaningful multiword phrases. To address this limitation, we presented a new phrase-based method, NPFST, for enriching a unigram BOW. NPFST is suitable for many different kinds of English text; it does not require any specialized configuration or annotations.

We compared NPFST to several other methods for extracting phrases, focusing on yield, recall, efficiency, and interpretability. As desired, NPFST has a low yield and high recall, and efficiently extracts highly interpretable phrases. Finally, to demonstrate the usefulness of NPFST for social scientists, we used NPFST to explore partisan differences in U.S. congressional legislation about law and crime. We found that the phrases extracted by NPFST were less ambiguous and more interpretable than unigrams.

In the future, we plan to use NPFST in combination with other text analysis methods, such as topic modeling; we have already obtained encouraging preliminary results. We have also experimented with modifying the FullNP grammar to select broader classes of phrases, such as subject–verb and verb–object constructions (though we anticipate that more structured syntactic parsing approaches will eventually be useful for these kinds of constructions).

Method	Party	Ranked List
unigrams	Democrat	and, deleted, health, mental, domestic, inserting, grant, programs, prevention, violence, program, striking, education, forensic, standards, juvenile, grants, partner, science, research
	Republican	any, offense, property, imprisoned, whoever, person, more, alien, knowingly, officer, not, united, intent, commerce, communication, forfeiture, immigration, official, interstate, subchapter
NPFST	Democrat	mental health, juvenile justice and delinquency prevention act, victims of domestic violence, child support enforcement act of u.s.c., fiscal year, child abuse prevention and treatment act, omnibus crime control and safe streets act of u.s.c., date of enactment of this act, violence prevention, director of the national institute, former spouse, section of the foreign intelligence surveillance act of u.s.c., justice system, substance abuse criminal street gang, such youth, forensic science, authorization of appropriations, grant program
	Republican	special maritime and territorial jurisdiction of the united states, interstate or foreign commerce, federal prison, section of the immigration and nationality act, electronic communication service provider, motor vehicles, such persons, serious bodily injury, controlled substances act, department or agency, one year, political subdivision of a state, civil action, section of the immigration and nationality act u.s.c., offense under this section, five years, bureau of prisons, foreign government, explosive materials, other person

Table 4: Ranked lists of unigrams and representative phrases of length two or more for Democrats and Republicans.

Our open-source implementation of NPFST is available at http://slanglab.cs.umass.edu/phrases/.

Acknowledgments

We thank the anonymous reviewers for their comments (especially the suggestion of FSA backtracking) on earlier versions of this work. We also thank Ken Benoit, Brian Dillon, Chris Dyer, Michael Heilman, and Bryan Routledge for helpful discussions. MD was supported by NSF Grant DGE-1144860.

Appendix: FullNP Grammar

The following `foma` grammar defines the rewrite phrase transducer P:

```
# POS tag categories. "Coarse" refer to the Petrov Univeral tag set.
# We directly use PTB tags, but for Twitter, we assume they've been
# preprocessed to coarse tags.
# CD is intentionally under both Adj and Noun.
define Adj1     [JJ | JJR | JJS | CD | CoarseADJ];
define Det1     [DT | CoarseDET];
define Prep1    [IN | TO | CoarseADP];
define Adv1     [RB | RBR | RBS | CoarseADV];
# Note that Twitter and coarse tags subsume some of this under VERB.
define VerbMod1 [Adv1 | RP | MD | CoarsePRT];
# PTB FW goes to CoarseX, but we're excluding CoarseX since for Gimpel et al.'s
# Twitter tags, that's usually non-constituent-participating things like URLs.
define Noun     [NN | NNS | NNP | NNPS | FW | CD | CoarseNOUN | CoarseNUM];
define Verb     [VB | VBD | VBG | VBN | VBP | VBZ | CoarseVERB];
define AnyPOS   [O  |  Adj1|Det1|Prep1|Adv1|VerbMod1|Noun|Verb |
    CoarseDOT|CoarseADJ|CoarseADP|CoarseADV|CoarseCONJ|CoarseDET|
    CoarseNOUN|CoarseNUM|CoarsePRON|CoarsePRT|CoarseVERB|CoarseX
]
define Lparen   ["-LRB-" | "-LSB-" | "-LCB-"];  # Twitter doesnt have this.
define Rparen   ["-RRB-" | "-RSB-" | "-RCB-"];
# Ideally, auxiliary verbs would be VerbMod, but PTB gives them VB* tags.

# single-word coordinations
define Adj      Adj1 [CC Adj1]*;
define Det      Det1 [CC Det1]*;
define Adv      Adv1 [CC Adv1]*;
define Prep     Prep1 [CC Prep1]*;
define VerbMod  VerbMod1 [CC VerbMod1]*;

# NP (and thus BaseNP) have to be able to stand on their own.  They are not
# allowed to start with a determiner, since it's usually extraneous for our
# purposes. But when we want an NP right of something, we need to allow
# optional determiners since they're in between.
define BaseNP   [Adj|Noun]* Noun;
define PP       Prep+ [Det|Adj]* BaseNP;
define ParenP   Lparen AnyPOS^{1,50} Rparen;
define NP1      BaseNP [PP | ParenP]*;
define NP       NP1 [CC [Det|Adj]* NP1]*;

regex NP -> START ... END;
write att compiled_fsts/NP.attfoma
```

References

[Abney1997] Steven Abney. 1997. Part-of-speech tagging and partial parsing. In *Corpus-based methods in language and speech processing*, pages 118–136. Springer.

[Adler and Wilkerson2014] E. Scott Adler and John Wilkerson. 2014. Congressional Bills Project: (1980-2004).

[Allauzen et al.2007] Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFST: A general and efficient weighted finite-state transducer library. In *Implementation and Application of Automata*, pages 11–23. Springer.

[Baldwin et al.2015] Timothy Baldwin, Marie-Catherine de Marneffe, Bo Han, Young-Bum Kim, Alan Ritter, and Wei Xu. 2015. Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 126–135, Beijing, China, July. Association for Computational Linguistics.

[Bamman and Smith2014] David Bamman and Noah A. Smith. 2014. Unsupervised discovery of biographical structure from text. *Transactions of the Association for Computational Linguistics*, 2:363–376.

[Beesley and Karttunen2003] Kenneth R. Beesley and Lauri Karttunen. 2003. *Finite-state morphology: Xerox tools and techniques*. CSLI, Stanford.

[Benoit and Nulty2015] Kenneth Benoit and Paul Nulty. 2015. More than unigrams can say: Detecting meaningful multi-word expressions in political text. *MPSA Working Paper*, pages 1–19.

[Brysbaert et al.2014] Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2014. Concreteness ratings for 40 thousand generally known English word lemmas. *Behavior Research Methods*, 46(3):904–911.

[Chuang et al.2012] Jason Chuang, Christopher D. Manning, and Jeffrey Heer. 2012. "Without the clutter of unimportant words": Descriptive keyphrases for text visualization. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 19(3):19.

[Dunning1993] Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19:61–74.

[Frantzi et al.2000] Katerina Frantzi, Sophia Ananiadou, and Hideki Mima. 2000. Automatic recognition of multi-word terms: The c-value/nc-value method. *International Journal on Digital Libraries*, 3(2):115–130.

[Gimpel et al.2011] Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for Twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 42–47. Association for Computational Linguistics.

[Grimmer and Stewart2013] Justin Grimmer and Brandon M. Stewart. 2013. Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political Analysis*, 21(3):267–297.

[Grimmer2010] Justin Grimmer. 2010. A Bayesian hierarchical topic model for political texts: Measuring expressed agendas in Senate press releases. *Political Analysis*, 18(1):1–35.

[Hannah and Wallach2014] Lauren Hannah and Hanna Wallach. 2014. Topic summarization: From word lists to phrases. In *Proceedings of the NIPS Workshop on"Modern Machine Learning and Natural Language Processing"*.

[Hulden2009] Mans Hulden. 2009. Foma: A finite-state compiler and library. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics: Demonstrations Session*, pages 29–32. Association for Computational Linguistics.

[Justeson and Katz1995] John S. Justeson and Slava M. Katz. 1995. Technical terminology: Some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, 1(01):9–27.

[Kim et al.2010] Su Nam Kim, Timothy Baldwin, and Min-Yen Kan. 2010. Evaluating n-gram based evaluation metrics for automatic keyphrase extraction. In *Proceedings of the 23rd international conference on computational linguistics*, pages 572–580. Association for Computational Linguistics.

[Kim et al.2011] Jin-Dong Kim, Sampo Pyysalo, Tomoko Ohta, Robert Bossy, Ngan Nguyen, and Jun'ichi Tsujii. 2011. Overview of BioNLP shared task 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 1–6. Association for Computational Linguistics.

[Kuleshov and Liang2015] Volodymyr Kuleshov and Percy S. Liang. 2015. Calibrated structured prediction. In *Advances in Neural Information Processing Systems*, pages 3456–3464.

[Laver et al.2003] Michael Laver, Kenneth Benoit, and John Garry. 2003. Extracting policy positions from political texts using words as data. *The American Political Science Review*, 97(2):311–331.

[Manning and Schütze1999] Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

[Manning et al.2014] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J.

Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

[Marcus et al.1993] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

[Monroe et al.2008] Burt L. Monroe, Michael P. Colaresi, and Kevin M. Quinn. 2008. Fightin' words: Lexical feature selection and evaluation for identifying the content of political conflict. *Political Analysis*, 16:372–403.

[Nguyen and O'Connor2015] Khanh Nguyen and Brendan O'Connor. 2015. Posterior calibration and exploratory analysis for natural language processing models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1587–1598. Association for Computational Linguistics.

[O'Connor2014] B O'Connor. 2014. MITEXTEXPLORER : Linked brushing and mutual information for exploratory text data analysis. In *ACL 2014 Workshop on Interactive Language Learning, Visualization, and Interfaces*.

[Owoputi et al.2013] Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of NAACL*.

[Petrov et al.2011] Slav Petrov, Dipanjan Das, and Ryan McDonald. 2011. A universal part-of-speech tagset. *arXiv preprint arXiv:1104.2086*.

[Purpura and Hillard2006] Stephen Purpura and Dustin Hillard. 2006. Automated classification of congressional legislation. *Proceedings of The 2006 International Conference on Digital Government Research*, pages 219–225.

[Quinn et al.2010] Kevin M. Quinn, Burt L. Monroe, Michael Colaresi, Michael H. Crespin, and Dragomir R. Radev. 2010. How to analyze political attention with minimal assumptions and costs. *American Journal of Political Science*, 54(1):209–228.

[Roche and Schabes1997] Emmanuel Roche and Yves Schabes. 1997. *Finite-State Language Processing*. MIT Press.

[Sandhaus2008] Evan Sandhaus. 2008. The New York Times Annotated Corpus. *Linguistic Data Consortium*, LDC2008T19.

[Slapin and Proksch2008] Jonathan B Slapin and Sven-Oliver Proksch. 2008. A scaling model for estimating time-serial positions from texts. *American Journal of Political Science*, 52(3):705–722.

[Toutanova et al.2003] K. Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180.

[Vadas and Curran2011] David Vadas and James R. Curran. 2011. Parsing noun phrases in the Penn Treebank. *Computational Linguistics*, 37(4):753–809.

[Yogatama et al.2015] Dani Yogatama, Lingpeng Kong, and Noah A. Smith. 2015. Bayesian optimization of text representations. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2100–2105. Association for Computational Linguistics.