Prototype Synthesis for Model Laws

Matthew Burgess
University of Michigan
mattburg@umich.edu

Eugenia Giraudy
UC Berkeley & YouGov
eugenia.giraudy@gmail.com

Eytan Adar
University of Michigan
eadar@umich.edu

Abstract

State legislatures often rely on existing text when drafting new bills. Resource and expertise constraints, which often drive this copying behavior, can be taken advantage of by lobbyists and special interest groups. These groups provide *model bills*, which encode policy agendas, with the intent that the models become actual law. Unfortunately, model legislation is often opaque to the public–both in source and content. In this paper we present LOBBYBACK, a system that reverse engineers model legislation from observed text. LOBBYBACK identifies clusters of bills which have text reuse and generates "prototypes" that represent a canonical version of the text shared between the documents. We demonstrate that LOBBYBACK accurately reconstructs model legislation and apply it to a dataset of over 550k bills.

1 Introduction

Beginning in 2005, a number of states began passing "Stand Your Ground" laws–legal protections for the use of deadly force in self-defense. Within a few years, at least two dozen states implemented a version of the this legislation (Garrett and Jansa, 2015). Though each state passed its own variant, there is striking similarity in the text of the legislation. While seemingly "viral" the expedient adoption of these laws was not the result of an organic diffusion process, but rather more centralized efforts. An interest group, the American Legislative Exchange Council (ALEC), drafted model legislation (in this case modeled on Florida's law) and lobbied to have the model law enacted in other states. While the influence of the lobbyists through model laws grows, the hidden nature of their original text (and source) creates a troubling opacity.

Reconstructing such hidden text through analysis of observed, potentially highly mutated, copies poses an interesting and challenging NLP problem. We refer to this as the *Dark Corpora* problem. Since legislatures are not required to cite the source of the text that goes into a drafted bill, the bills that share text are unknown beforehand. The first problem therein lies in identifying clusters of bills with reused text. Once a cluster is identified, a second challenge is the reconstruction of the original or prototype bill that corresponds to the observed text. The usual circumstances under which a model law is adopted by individual states involves "mutation." This may be as simple as modifying parameters to the existing policy (*e.g.*, changing the legal limit allowed of medical marijuana possession to 3.0 ounces from 2.5) or can be more substantial, with significant additions or deletions of different conditions of a policy. Interestingly, the need to maintain the objectives of the law creates a pressure to retain a legally meaningful structure and precise language–thus changes need to satisfy the existing laws of the state but carry out the intent of the model. Both subtle changes of this type, and more dramatic ones, are of great interest to political scientists. A specific application, for example, may be predicting likely spots for future modifications as additional states adopt the law. Our challenge is to identify and represent "prototype" sentences that capture the *similarity* of observed sentences while also capturing the *variation*.

In this paper we propose LOBBYBACK, a system that automatically identifies clusters of documents that exhibit text reuse, and generates "prototypes" that represent a canonical version of text shared between the documents. In order to syn-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1579–1588,
Berlin, Germany, August 7-12, 2016.

thesize the prototypes, LOBBYBACK first extracts clusters of sentences, where each sentence pertains to the same policy but can exhibit variation. LOBBYBACK then uses a greedy multi-sequence alignment algorithm to identify an approximation of the optimal alignment between the sentences. Prototype sentences are synthesized by computing a consensus sentence from the multi-sentence alignment. As sentence variants are critical in understanding the effect of the model legislation, we can not simply generate a single "common" summary sentence as a prototype. Rather, LOBBYBACK creates a data structure that captures this variability in text for display to the end-user.

With LOBBYBACK, end-users can quickly identify clusters of text reuse to better understand what type of policies are diffused across states. In other applications, sentence prototypes can be used by journalists and researchers to discover previously unknown model legislation and the involvement of lobbying organizations. For example, prototype text can be compared to the language or policy content of interest groups documents and accompanied with qualitative research it can help discover which lobbyists have drafted this legislation.

We evaluated LOBBYBACK on the task of reconstructing 122 known model legislation documents. Our system was able to achieve an average of 0.6 F1 score based on the number of prototype sentences that had high similarity with sentences from the model legislation. We have also run LOBBYBACK on the entire corpus of state legislation (571,000 documents) from openstates.org as an open task. The system identified 4,446 clusters for which we generated prototype documents. We have released the resulting data set and code at http://github.com/mattburg/LobbyBack. LOBBYBACK is novel in fully automating and scaling the pipeline of model-legislation reconstruction. The output of this pipeline captures both the likely "source sentences" but also the variations of those sentences.

2 Related Work

While no specific system or technique has focused on the problem of legislative document reconstruction, we find related work in a number of domains. Multi-document summarization (MDS), for example, can be used to partially model the underlying problem–generating a representative document from multiple sources. Extractive MDS, in particular, is promising in that representative sentences are identified.

Early work in extractive summarization include greedy approaches such as that proposed by Carbonell and Goldstein (1998). The algorithm uses an objective function which trades off between relevance and redundancy. Global optimization techniques attempt to generate "summaries" (selected sets of sentences or utterances) that maximize an objective based on informativeness, redundancy and/or length of summary. These have shown superior performance to greedy algorithms (Yih et al., 2007; Gillick et al., 2009). Approaches based on neural networks have recently been proposed for ranking candidate sentences (Cao et al., 2015). Graph based methods, such as LexRank (Erkan and Radev, 2004), have also proven effective for MDS. Extensions to this approach combine sentence ranking with clustering in order to minimize redundancy (Qazvinian and Radev, 2008; Wan and Yang, 2008; Cai and Li, 2013). The C-LexRank algorithm (Qazvinian and Radev, 2008), in particular, uses this combination and inspired our high level design. Similar to our approach, Wang and Cardie (2013) propose a method for generating templates of meeting summaries using multi-sequence alignment.

Though related, it is important to note that the objectives of summarization (informativeness, reduced redundancy, etc.) are not entirely consistent with our task. For example, using the n-gram co-occurrence based ROUGE score would not be sufficient at evaluating LOBBYBACK. Our goal is to accurately reconstruct entire sentences of a hidden document given observed mutations of that document. Additionally, our goal is not simply to find a representative sentence that reflects the original document, but to capture the similarity *and* variability of the text within a given "sentence cluster."

Within the political science and legal studies communities research has focused on *manual* approaches to both understanding how model legislation impacts law and how policy ideas diffuse between bill text. As these studies are time consuming, there is no large-scale or broad analysis of legislative materials. Rather, researchers have limited their workload by focusing on a single topic (*e.g.*, abortion (Patton, 2003) and crime (Kent and Carmichael, 2015)) or a single lobbying group (*e.g.*, ALEC (Jackman, 2013). Similarly, those

studying policy diffusion across US states have also limited their analysis to a few topics (*e.g.*, same-sex marriage (Haider-Markel, 2001)).

Recent attempts to automate the analysis of model legislation has had similar problems, as most researchers have limited their analysis to one interest group or a few relevant topics (Hertel-Fernandez and Kashin, 2015; Jansa et al., 2015). Hertel-Fernandez and Kashin proposed a supervised model in which they train on hand labeled examples of state bills that borrow text and/or concepts from ALEC bills. The problem they focus on is different from ours. The motivation behind LOBBYBACK is that there exists many model legislation which we don't have access to and the goal is to try and reconstruct these documents without labeled training data. Jansa *et al.* propose a technique for inferring a network of policy diffusion for manually labeled clusters of bills. Both Jansa *et al.* and Hertel-Fernandaz and Kashin propose techniques that only look at the problem of inferring whether two bills exhibit text reuse but unlike LOBBYBACK they do not attempt to infer whether specific policies (sentences) in the documents are similar/different.

A related "dark corpora" problem, though at a far smaller scale, is the biblical "Q source" where hidden oral sources are reconstructed through textual analysis (Mournet, 2005).

3 Problem Definition

Policy diffusion is a common phenomenon in state bills (Gray, 1973; Shipan and Volden, 2008; Berry and Berry, 1990; Haider-Markel, 2001). Unlike members of the (Federal) Congress, few state legislators have the expertise, time, and staff to draft legislation. It is far easier for a legislator to adapt existing legislative text than to write a bill from scratch. As a consequence, state legislatures have an increased willingness to adopt legislation drafted by interest groups or by legislators in other states (Jansa et al., 2015). In addition to states borrowing text from other legislators and lobbyists, another reason why bills can exhibit text reuse is when a new federal law passes and each state needs to modify its existing policy to conform with the new federal law.

The result of legislative copying, whether caused by diffusion between states, influence from a lobby or the passing of a new federal law, is similar: a cluster of bills will share very similar text, often varying only by implementation details of a given policy. The goal in constructing a prototype document–a representation of the "original" text–is to synthesize this document from the modified copies. In the case when the bill cluster was influenced by one external source, such as lobby or passage of a federal bill, the ideal prototype document would capture the language that each bill borrowed from the source document. In the case when their is not one single document that influenced a cluster of bills, the prototype will still give a summary of a concise description of the diffused text between bills, providing fast insight into what text was shared and changed within a bill cluster.

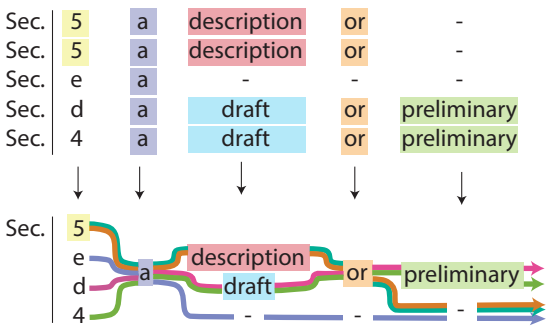


Figure 1: Visualization of a multi-sentence alignment (fragment) and resulting prototype sentence.

3.1 State Legislation Corpus

We obtained the entire openstates.org corpus of state legislation, which includes 550,000 bills and 200,000 resolutions for all 50 states. While for some states this corpus includes data since 2007, for the majority of states we have data from 2010 on. We do not include data from Puerto Rico, where the text is in Spanish, and from Washington DC, which includes many idiosyncrasies (*e.g.*, correspondence from city commissions introduced as bills). On average, each state introduced 10,524 bills, with an average length of 1205 words.

4 LOBBYBACK Architecture

LOBBYBACK consists of 3 major components. The first component identifies clusters of bills that have text reuse. Then for each of these bill clusters, LOBBYBACK extracts and clusters the sentences from all documents. For each of the sentence clusters, LOBBYBACK synthesizes prototype sentences in order to capture the similarity and variability of the sentences in the cluster.

4.1 Clustering Bills

Groups of bills with significant text reuse represent candidates for analysis as they have may have all copied from the same model legislation. There are a number of ways one could identify such clusters through text mining. In our implementation, we have opted to generate a network representation of the bills and then use a network clustering (i.e., "community-finding") algorithm to generate the bill clusters. In our network representation each node represents a state bill and weighted edges represent the degree to which two bills exhibit substantive text reuse. Since most pairs of bills do not have text reuse, we chose to use a network model because community finding algorithms work well on sparse data and do not require any parameter choice for the number of clusters. In the context of this paper, text reuse occurs when two state bills share:

1. Long passages of text, *e.g.* (sections of bills) that *can* differ in details.
2. These passages contain text of substantive nature to the topic of the bill (*i.e.*, text that is *not* boilerplate).

In addition to text that describes policy, state bills also contain boilerplate text that is common to all bills from a particular state or to a particular topic. Examples of legislative boilerplate include: "Read first time 01/29/16. Referred to Committee on Higher Education" (meta-data describing where the bill is in the legislative process); and "Safety clause. The general assembly hereby finds, determines, and declares ..." (a standard clause included in nearly all legislation from Colorado, stating the eligibility of a bill to be petitioned with a referendum).

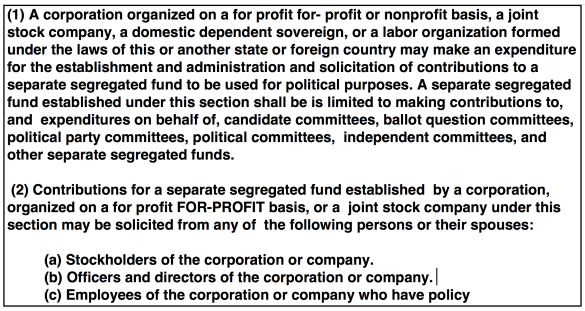
In order to identify pairs of bills that exhibit text reuse, we created an inverted index that contained n-grams ranging from size 4-8. We use ElasticSearch to implement the inverted index and computed the similarity between bills using the "More like This" (MLT) query (Elastic, 2016). The MLT query first selects the 100 highest scoring TF*IDF n-grams from a given document and uses those to form a search query. The MLT query is able to quickly compute the similarity between documents and since it ranks the query terms by using TF*IDF the query text is more likely to be substantive rather then boilerplate. The MLT query we used was configured to only return documents that matched at least 50% of the query's shingles and returned at most 100 documents per query. By implementing the similarity search using a TF*IDF cutoff we were able to scale the similarity computation while still maintaining our desire to identify reuse of substantive text.

The edges of the bill similarity network are computed by calculating pairwise similarity. Each bill is submitted as input for an MLT query and scored matches are returned by the search engine. Since the MLT query extracts n-grams only for the query document, the similarity function between two documents d_i and d_j is not symmetric. We construct a symmetric bill similarity network by taking the average score of each (d_i, d_j) and its reciprocal (d_j, d_i). A non-existent edge is represented as an edge with score 0. We further reduce the occurrence of false-positive edges by removing all edges with a score lower than 0.1. The resulting network is very sparse, consisting of 35,346 bills that have 1 or more neighbors, 125,401 edges, and 3534 connected components that contain an average of 10 bills.

A specific connected component may contain more than one bill cluster. To isolate these clusters in the bill network we use the InfoMap community detection algorithm (Rosvall and Bergstrom, 2008). We use the InfoMap algorithm because it has been to shown to be one of the best community detection algorithms and it is able to detect clusters at a finer granularity than other methods. Our corpus contains both bills that have passed and those that have not. Bills can often be reintroduced in their entirety after failing the previous year. As we do not want to bias the clusters towards bills that are re-introduced more than others, we filter the clusters such that they only include the earliest bill from each state.

4.2 Prototype Synthesis

Once we have identified a cluster of bills that have likely emerged from a single "source" we would like to construct a plausible representation of that source. The prototype synthesizer achieves this by constructing a canonical document that captures the similarity and variability of the content in a given bill cluster. The two main steps in prototype synthesis consists of clustering bill sentences and generating prototype sentences from the clusters.


(1) A corporation organized on a for profit for- profit or nonprofit basis, a joint stock company, a domestic dependent sovereign, or a labor organization formed under the laws of this or another state or foreign country may make an expenditure for the establishment and administration and solicitation of contributions to a separate segregated fund to be used for political purposes. A separate segregated fund established under this section shall be is limited to making contributions to, and expenditures on behalf of, candidate committees, ballot question committees, political party committees, political committees, independent committees, and other separate segregated funds.

(2) Contributions for a separate segregated fund established by a corporation, organized on a for profit FOR-PROFIT basis, or a joint stock company under this section may be solicited from any of the following persons or their spouses:

(a) Stockholders of the corporation or company.
(b) Officers and directors of the corporation or company.
(c) Employees of the corporation or company who have policy

Figure 2: An sample state bill segment from Michigan Senate Bill 571

4.2.1 Sentence Clustering

Most state bills have a common structure, consisting of an introduction that describes the intent of the bill followed by sections that contain the law to be implemented. Each section of a bill is comprised of self-contained policy, usually consisting of a long sentence that describes the policy and the implementation details of that policy. Each document is segmented into these policy sentences using the standard Python NLTK sentence extractor. Sentences are cleaned by removing spurious white space characters, surrounding punctuation and lower-casing each word. Once we have extracted all of the sentences for a given bill cluster, we compute the cosine similarity between all pairs of sentences which are represented using a unigram bag-of-words model. We used a simple unweighted bag of words model because in legal text stop words can be import[1] In this case, we are generating a similarity "matrix" capturing sentence–sentence similarity.

Given the similarity matrix, our next goal is to isolate clusters of variant sentences that likely came from the same source sentence. We elected to use the DBSCAN (Ester et al., 1996) algorithm to generate these clusters. The DBSCAN algorithm provides us with tunable parameters that can isolate better clusters. Specifically, the parameter ϵ controls the maximum distance between any two points in the same neighborhood. By varying ϵ we are able to control both the number of clusters and the amount of sentence variation within a cluster. A second reason for selecting DBSCAN is that the algorithm automatically deals with noisy data points, placing all points that are not close enough to other points in a separate cluster labeled "noise." Since many sentences in a given bill cluster do not contribute to the reused text between bills, the noise cluster is useful for grouping those sentences together rather than having them be outsiders in "good" clusters.

[1]The difference between the words "shall" and "may" for instance is important, the former requires that a specific action be put on a states budget while the later does not

4.2.2 Multi-Sequence Alignment

Once we have sentence clusters we then synthesize a "prototype" sentence from all of the sentences in a given cluster. An ideal prototype "sentence" is one that simultaneously captures the similarity between each sentence in the cluster (the common sentence structures) and the variation between the sentences in a cluster. For a simple pair of (partial) sentences, "The Department of Motor Vehicles retains the right to . . . " and "The Department of Transportation retains the right to . . . ", a prototype might be of the form, "The Department of { Motor Vehicles, Transportation } retains the rights to . . . " Our "sentence" is not strictly a single linear piece of text. Rather, we have a data structure that describes alternative sub-strings and captures variant text.

To generate this structure we propose an algorithm that computes an approximation of the optimal multi-sentence alignment (MSA) in the cluster and then generates a prototype sentence representing a 'consensus' for sentences in the MSA.

We generate an MSA using a modified version of the iterative pairwise alignment algorithm described in (Gusfield, 1997). The greedy algorithm builds a multi-alignment by iteratively applying the Needleman-Wunsch pairwise global alignment algorithm. Needleman-Wunsch computes the optimal pairwise alignment by maximizing the alignment score between two sentences. An alignment score is calculated based on three parameters: word matches, word mismatches (when a word appears in one sentence but not the other), and gaps (when the algorithm inserts a space in one of the sentences).

An MSA is generated by the following steps:

1. Construct an edit-distance matrix for all pairs of sentences
2. Construct an initial alignment between the two sentences with the smallest edit distance
3. Repeat this step k times:
   (a) Select the sentence with the smallest average edit distance to the current MSA.
   (b) Add the chosen sentence to the MSA by aligning it to the existing MSA.

The algorithm stops after the alignment has reached a size that is determined as a free parameter k (user configured, but can be chosen to be the number of sentences in the cluster). As the algorithm execution is ordered on the edit distance between the current MSA and the next sentence to be added, the larger the MSA, the more variation we are allowing in the prototype sentence.

4.2.3 Synthesizing Prototype Sentences

We synthesize a prototype sentence by finding a *consensus* sentence from all of the aligned sentences in the MSA for a given cluster. We achieve this by going through each "column" of the MSA and using the following rules to decide which token will be used in the prototype. A token can be either a word in one of the sentences or a "space" that was inserted during the alignment process.

1. If there is a token that occurs in the majority of alignments ($> 50\%$) then that token is chosen.
2. If no token appears in a majority, then a special variable token is constructed that displays all of the possible tokens in each sentence. For example the 1^{st} and 3^{rd} columns in Figure 1.
3. If a space is the majority token chosen then it is shown as a variable token with the second most common token.

5 Evaluation

We provide experiments that evaluate all three components of LOBBYBACK.

5.1 Model Legislation Corpus

To test the effectiveness of LOBBYBACK in recreating a model bill, we first identified a set of known model bills. Our model legislation corpus consists of 1846 model bills that we found by searching on Google using the keywords "model law", "model policy" and "model legislation." Most of the corpus is comprised of bills from the conservative lobby group, American Legislative Exchange Council (ALEC, 708 documents), its liberal counterpart, the State Innovative Exchange (SIX, 269 documents) and the non-partisan Council of State Governments (CSG, 470 documents), and the remainder (399) from smaller interest groups that focus on specific issues.

Using the clusters we previously described (Section 4.1), we found the most similar cluster to each model bill. This was done by first computing the set of neighbors (ego-network) for a model bill using the same procedure used in creating the bill similarity network. We then matched a bill cluster to the model legislation by finding the bill cluster that had the highest Jaccard similarity (based on the overlapping bills in each cluster) with the neighbor set of a model bill. Each test example in our evaluation data set consists of model bill and its corresponding bill cluster. The total number of model legislation documents that had matches in the state bill corpus was 360 documents.

Once we have an evaluation data set comprised of model bill/cluster pairs our goal is to compare the prototype sentences we infer for a cluster to the model bill that matches that cluster. Since we do not have ground truth on which sentences from the model bill match sentences in the documents that comprise the cluster we need to infer such labels. In order to identify which sentences from the model legislation actually get re-used in the bill cluster, we take the following steps:

1. Extract all sentences from each of the bills in a cluster and the sentences in the corresponding model legislation.
2. Compute the pairwise cosine similarity between bill sentences and each of the model bill sentences using the same unigram bag-of-words model described in Section 4.2.1
3. Compute the "oracle" matching M^* using the Munkres algorithm (Munkres, 1957)

The Munkres algorithm gives the best possible one-to-one matching between the sentences in the model legislation and the sentences in the bill clusters. There are some sentences in the model bill that are never used in actual state legislation (e.g., sentences that describe the intent of a law or instructions of how to implement a model policy). Therefore we label model bill sentences in M^* that match a bill sentence with a score greater than 0.85 as true matches (S^*)[2]. The final set of 122 evaluation examples consists of all model legislation/bill cluster pairs where more than 50% of model bill sentences have true matches.

[2]A threshold of 0.85 was found effective in prior work (Garrett and Jansa, 2015) and we observed good separation between matching sentences and non-matches in our data-set.

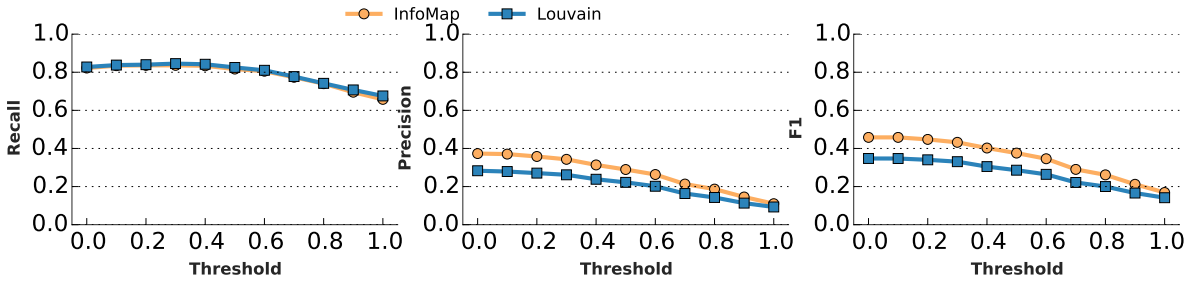


Figure 3: Precision, Recall, and F1 scores for the bill clustering component of LOBBYBACK configured with both the Louvain and InfoMap clustering algorithms.

5.2 Baselines

While no specific baseline exists for our problem, we implemented two alternatives to test against. The first, Random-Baseline, was implemented simply to show the performance of randomly constructing prototype documents. The second, LexRank-Baseline, implements a popular extractive summarization method.

Random-Baseline – The *random-baseline* randomly samples sentences from a given bill cluster. The number of sentences it samples is equal to the number of sentences in the optimal matching $|M^*|$

LexRank-Baseline – The LexRank baseline uses the exact same clustering algorithm as LOBBYBACK except instead of synthesizing prototype sentences, it uses the LexRank algorithm (Erkan and Radev, 2004) to pick the most salient sentence from each of the sentence clusters.

5.3 Evaluating Bill Clusters

As described in Section 5.1, each model bill is associated with a set of bills that comprise its neighbors in the bill network. In order to evaluate how LOBBYBACK clusters bills we compare the inferred clusters to the corresponding neighbor set for each model bill.

The inferred cluster for a given model bill can exhibit false positive and false negative errors. False negatives, in which a bill that exists in the neighbor set of a model bill but is not contained in the inferred cluster, are easy to identify (allowing us to calculate recall). Precision, on the other hand, is more difficult as any "extra" bills in the inferred cluster are not necessarily incorrect. It is possible that there are bills which do exhibit text re-use with the model legislation but did not match via the ElasticSearch query used to construct the network. We have found that in practice the second component of LOBBYBACK, which clusters the sentences extracted from a bill cluster, is robust to false-positives due to DBSCAN's treatment of "noisy" data points. Because of this, we are more concerned with recall in the bill clustering module as any data lost in this step propagates through the rest of the system.

Figure 3 shows the precision, recall, and F1 scores for LOBBYBACK coupled with both the Louvain (Blondel et al., 2008) and InfoMap (Rosvall and Bergstrom, 2008) network clustering algorithms. Because we do not know with absolute certainty which state bills are derived from model legislation, we would like to test our approach at different levels of confidence. To do this we vary the threshold on the similarity scores (edge weights) of the ego-network determined for each model legislation. A normalized weight is computed by taking the score provided by ElasticSearch and dividing that edge weight weight by the maximum edge weight in the neighbor set (to control for the unbounded scores provided by ElasticSearch). We vary the threshold for this weight (ranging from 0 to 1) and calculate the precision, recall, and F1 on a neighborhood set that is comprised of all bills that have a weight greater than the threshold. Higher threshold values means fewer state bills are included in the set, but they are increasingly similar to the model bill.

As shown in Figure 3, recall stays high for the majority of threshold values. This indicates that both clustering algorithms do a good job at recovering the bills that are the most similar to the model legislation. However, the two clustering algorithms differ somewhat in precision. Louvain, in particular performs worse, as it suffer from "resolution" problems. While effective for networks with large clusters, Louvain can not isolate small groups, of which we have many. For this reason,

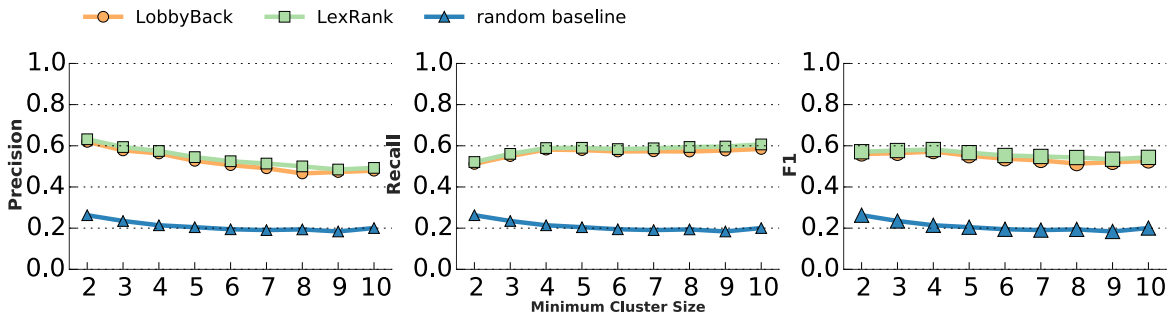


Figure 4: Precision, Recall, and F1 scores of LOBBYBACK and baselines.

we chose InfoMap as the method to use for the final implementation of LOBBYBACK.

5.4 Evaluating Sentence Clusters

We first evaluate the quality of the sentence clusters using the optimal matching M^* described above. For each test example in the evaluation set we generate a prototype document using LOBBYBACK and each of the baselines described above. We then compute a matching M between the prototype sentences and the model bill sentences (using the same procedure described in 5.1), where S is the set of sentences in the prototype and $S_{0.85}$ is the set of sentences that match with a score greater than 0.85. We compute precision, P, as $P = |S_{0.85}|/|S|$, and recall, R, as $R = |S|/|S^*|$.

Figure 4 shows precision, recall and F1 scores for both baselines and LOBBYBACK. Each curve is generated by averaging the precision/recall/F1 scores computed for each of the examples in the test set. The x-axis represents the minimum bill cluster size of the test examples for which the score is computed. For example, a minimum cluster size would average over all test examples with at least 2 bills in a cluster. LOBBYBACK relies on the fact that if text was borrowed from a model bill, then it would have been borrowed by many of the bills in the cluster. By analyzing how LOBBYBACK performs with respect to the minimum cluster size, we can determine how much evidence LOBBYBACK needs in order to construct clusters that correspond to sentences in the model bills. While the performance of LOBBYBACK and the LexRank baseline substantially improves over the random baseline, the different between LOBBYBACK and LexRank for this task is negligible. Since our cut-off similarity is 0.85, all sentences above the threshold are treated as true positives, making the distinction between the LexRank baseline and system small. LOBBYBACK performs a little worse than LexRank for large cluster sizes because it is penalized for having space and variable tokens which don't occur in model bills. Space and variable tokens occur more frequently in prototype sentences in larger clusters because there is more variation in the sentence clusters.

5.5 Evaluating Sentence Synthesis

The experiment in the previous section evaluated the quality of the sentence clusters by treating all matching sentences with a similarity greater than 0.85 as true positives. Here we provide an evaluation of the synthesized sentences that LOBBYBACK generates and compare them to the LexRank baseline, which chooses the most salient sentence from each cluster. We evaluate the quality of the synthesized prototype sentences by computing the word-based edit-distance between the prototype sentence with its corresponding model bill sentence in S for each test example.

Since the prototypes contain variable and space tokens which do not occur in the model bill sentences we modify the standard edit distance algorithm by *not* penalizing space tokens and allowing for any of the tokens that comprise a variable to be positive matches. In addition, we remove punctuation and lowercase all words in all of the sentences, regardless of method. We generate the results in Table 1 by averaging the edit distance for a configuration of LOBBYBACK or LexRank over sentence clusters produced for each test example. LOBBYBACK was configured to run with the number of iterations set to the size of the sentence cluster.

We compared both the performance of LOBBYBACK and LexRank for DBSCAN ϵ values of 0.1 and 0.15 as well as computing the average edit distance for different minimum sizes of cluster values. As the table shows, LOBBYBACK obtains a lower edit distance than LexRank in every config-

uration and as the size of the clusters increase the gap between the two increases. The goal of LOBBYBACK is not to be a better summarization algorithm than LexRank. By comparing to LexRank and showing that the edit distances are smaller on average, we can conclude that the prototype sentences created by LOBBYBACK are capturing the text that is "central" or similar within a given cluster. In addition, the prototype sentences produced by LOBBYBACK are superior because they also capture and describe in a succinct way, the variability of the sentences within a cluster.

6 Discussion

One assumption that we made about the nature of state adoption of model legislation is that the legislatures make modifications that largely preserve the model language in an effort to preserve policy. However, we currently do not consider cases in which a legislature has intentionally obscured the text while still retaining the same meaning. While not as frequent as common text reuse, Hertel-Fernandez and Kashin (2015) observed that some legislatures almost completely changed the text while reusing the concepts. One area of future work would be to try and extend LOBBYBACK to be more robust to these cases. One strategy would be to allow for a more flexible representation of text, such as word vector embeddings. The embeddings might even be used to extend the multi-sentence alignment to include a penalty based on word distance in embedding space.

LOBBYBACK performs well on reconstructing model legislation from automatically generated bill clusters. However, there are a number of improvements that can refine part of the pipeline. A potential change, but one that is more computationally costly, would be to use a deeper parsing of the sentences that we extract from the documents. We used a simple unigram model when computing sentence similarities because we wanted to ensure that stop words were included–due to their importance in legal text. We suspect that by using a parser we could, for example, weight the similarity of noun-phrases, yielding a better similarity matrix and potentially higher precision/recall. Currently LOBBYBACK does not consider the order of the sentences when attempting to construct a prototype document. We envision a future version of LOBBYBACK that tries to reconstruct the original ordering of the prototype sentences.

		Min Cluster Size			
Method	ϵ	2	4	6	8
LOBBYBACK	0.1	**24.4**	**20.4**	**18.2**	**17.2**
LexRank	0.1	25.4	22.5	20.3	19.4
LOBBYBACK	0.15	**25.5**	**21.6**	**19.4**	**17.9**
LexRank	0.15	27.3	25.6	25.0	24.1

Table 1: Mean edit distance scores for LOBBYBACK and LexRank.

7 Conclusion

In this paper we present LOBBYBACK, a system to reconstruct the "dark corpora" that is comprised of model bills which are copied (and modified) by resource constrained state legislatures. LOBBYBACK first identifies clusters of text reuse in a large corpora of state legislation and then generates prototype sentences that summarizes the similarity and variation of the copied text in a bill cluster. We believe that by open-sourcing LOBBYBACK and releasing our data of prototype bills to the public, journalists and legal scholars can use our findings to better understand the origination of U.S state laws.

Acknowledgements

The authors would like to thank Mike Cafarella, Joe Walsh and Rayid Ghani for helpful discussions, the Sunlight Foundation for providing access to data and the anonymous reviewers for their helpful comments.

References

Frances Stokes Berry and William D. Berry. 1990. State lottery adoptions as policy innovations: An event history analysis. *American Political Science Review*, 84:395–415, 6.

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008.

Xiaoyan Cai and Wenjie Li. 2013. Ranking through clustering: An integrated approach to multi-document summarization. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(7):1424–1433.

Ziqiang Cao, Furu Wei, Li Dong, Sujian Li, and Ming Zhou. 2015. Ranking with recursive neural networks and its application to multi-document summarization. In *AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2153–2159. AAAI Press.

Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 335–336.

Elastic. 2016. Elasticsearch reference. *https://www.elastic.co/guide/en*.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22(1):457–479, December.

Martin Ester, Hans-Peter Kriegel, Jrg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press.

Kristin N. Garrett and Joshua M. Jansa. 2015. Interest group influence in policy diffusion networks. *State Politics & Policy Quarterly*, 15(3):387–417.

Dan Gillick, Korbinian Riedhammer, Benoit Favre, and Dilek Hakkani-Tur. 2009. A global optimization framework for meeting summarization. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, ICASSP '09, pages 4769–4772, Washington, DC, USA. IEEE Computer Society.

Virginia Gray. 1973. Innovation in the states: A diffusion study. *American Political Science Review*, 67(04):1174–1185.

Dan Gusfield. 1997. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, New York, NY, USA.

Donald P Haider-Markel. 2001. Policy diffusion as a geographical expansion of the scope of political conflict: Same-sex marriage bans in the 1990s. *State Politics & Policy Quarterly*, 1(1):5–26.

Alexander Hertel-Fernandez and Konstantin Kashin. 2015. Capturing business power across the states with text reuse. In *Annual Conference of the Midwest Political Science Association*, pages 16–19.

Molly Jackman. 2013. Alecs influence over lawmaking in state legislatures. *Brookings Institute Blog*, December 6.

Joshua M Jansa, Eric R Hansen, and Virginia H Gray. 2015. Copy and paste lawmaking: The diffusion of policy language across american state legislatures. *Working Paper*.

Stephanie L Kent and Jason T Carmichael. 2015. Legislative responses to wrongful conviction: Do partisan principals and advocacy efforts influence state-level criminal justice policy? *Social science research*, 52:147–160.

Terence C Mournet. 2005. *Oral tradition and literary dependency: Variability and stability in the synoptic tradition and Q*, volume 195. Mohr Siebeck.

J. Munkres. 1957. Algorithms for the assignment and transportation problems. *Journal of the Society of Industrial and Applied Mathematics*, 5(1):32–38, March.

Dana Jill Patton. 2003. *The Effect of U.S. Supreme Court Intervention on the Innovation and Diffusion of Post-Roe Abortion Policies in the American States, 1973-2000*. Ph.D. thesis, University of Kentucky.

Vahed Qazvinian and Dragomir R. Radev. 2008. Scientific paper summarization using citation summary networks. In *International Conference on Computational Linguistics*, COLING '08, pages 689–696, Stroudsburg, PA, USA. Association for Computational Linguistics.

Martin Rosvall and Carl T. Bergstrom. 2008. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123.

Charles R Shipan and Craig Volden. 2008. The mechanisms of policy diffusion. *American Journal of Political Science*, 52(4):840–857.

Xiaojun Wan and Jianwu Yang. 2008. Multi-document summarization using cluster-based link analysis. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 299–306.

Lu Wang and Claire Cardie. 2013. Domain-independent abstract generation for focused meeting summarization. In *Annual Meeting of the Association for Computational Linguistics*, ACL'13, pages 1395–1405.

Wen-tau Yih, Joshua Goodman, Lucy Vanderwende, and Hisami Suzuki. 2007. Multi-document summarization by maximizing informative content-words. In *International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 1776–1782.