Department of Statistics
University of California, Los Angeles

The Semantic Bootstrap: Application of the Bootstrap for Small Text Classification

Ryan Robert Rosario

March 14, 2017

Outline

1. Introduction
2. Existing Methods
3. Semantic Bootstrap Proposal
4. Experiments
5. Data
6. Models
7. Results
8. Conclusion

1 Introduction

2 Existing Methods

3 Semantic Bootstrap Proposal

4 Experiments

5 Data

6 Models

7 Results

8 Conclusion

Short Texts

Short texts are ubiquitous across the World Wide Web.

Short texts allow the Web to be more accessible to the world as users can communicate thoughts and desires and ingest new information in a very quick manner.

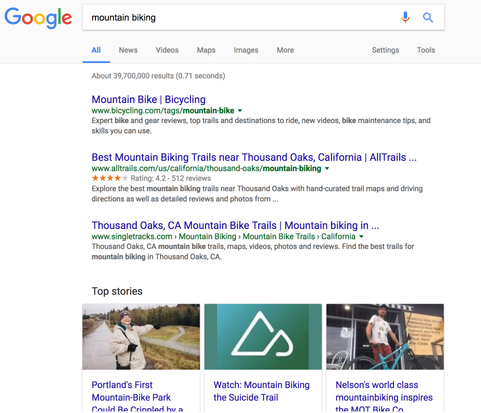
Example: Google Suggested Search

Suggested Queries for query "ucla."

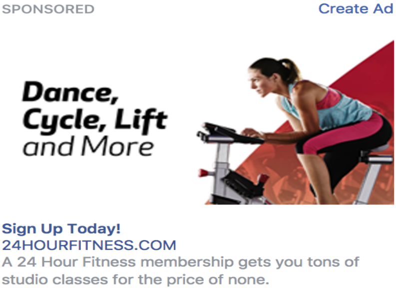
A classifier takes what a user types, as well as their location, user profile and other information and is able to suggest neighboring queries based on some cluster or classification. We extract meaning using external data.

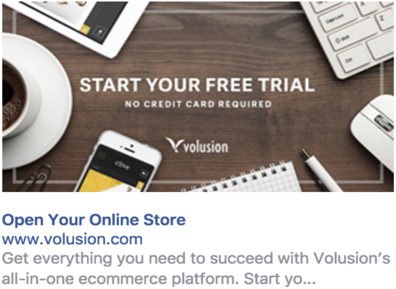
Example: Google Search Results

Customized Search Results: Another example of using external data to customize search results.



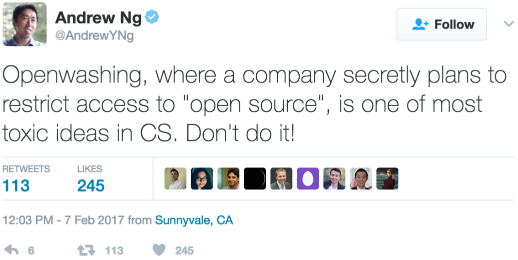
Example: Facebook Ad Creative





Example: Tweet

Twitter still rules the short text world.



But there a whole series of issues regarding their use in machine learning.

Example: Title of a Blog Post

Problems with Short Text Classification

In various works including [7][14][9][6], the main problems associated with *classifying* short texts are as follows

1. severe data/feature sparsity;
2. words have less discriminative power since less relational information is available;
3. limited context;

Our goal is to create a method that improves short text classification despite these limitations via text augmentation.

1 Introduction

2 Existing Methods

3 Semantic Bootstrap Proposal

4 Experiments

5 Data

6 Models

7 Results

8 Conclusion

Existing Methods

The difficulties of working with short texts date back several decades, gaining mass interest in the 1980s for database search. Text augmentation methods (also called query expansion) can be classified into three categories

1. Relevance Feedback
2. Domain-Specific
3. Lexical Modeling

Throughout the decades, the definition of "short" has also changed.

Existing Methods: Relevance Feedback

A relevance feedback method works as follows

1. user issues a short query (short text) $\mathbf{q}$;
2. system returns an initial set of (similar) results;
3. user marks results as relevant (similar) or not;
4. system computes ranking or retrieval criteria for a new query $\mathbf{q}'$ (augmented query);
5. system displays a revised set of retrieval results.

Existing Methods: Relevance Feedback, Xu and Croft Example

One key contribution is provided in Xu and Croft[13], where the human step is removed.

1. the retrieval engine uses *global*[1] context to return what it considers relevant results;
2. the results are scored and ranked as *local* context;
3. *concepts* from the top *n* results are included in augmented text/query.

Their method improved precision by about 10% on collections of long texts, but did not improve precision on smaller texts.

[1] The researchers defined global context as using word cooccurrence or other statistics.

Existing Methods: Relevance Feedback

The relevance feedback class of algorithms has the following drawbacks:

1. assumes that a retrieval system actually provides relevant results in the first place, and
2. **Implementation Detail:** the wording strongly suggests there is in fact a separate system, called the retrieval system;
3. based on a strict ranking system that may be too arbitrary to be used deterministically;
4. the type of data used in augmentation is arbitrary (concepts? phrases?);
5. authors of more modern methods (e.g. [10]) have cited these methods as performing poorly.

Existing Methods: Domain Based

In a domain-based approach, **external datasets** are used to augment or annotate texts for classification. The most common being Wikipedia[2] and WordNet[3].

In such domain-specific methods, there are some other features in common

1. they use some measure of term cooccurrence rather than retrieval and ranking;
2. they generate other features from the corpus as a way of transferring knowledge, such as part-of-speech.

[2] `http://www.wikipedia.org`
[3] `https://wordnet.princeton.edu/`

Existing Methods: Domain Based, Mandala et. al. Example

In [8], researchers used three different datasets (thesauri) to serve as the body where sampling occurs:

1. WordNet to measure distance between groups of terms (external data);
2. term cooccurrence metrics to identity synonyms;
3. "head-modifier-based" thesaurus which considered language structure such as subject-verb, adjective-noun etc. (feature generation).

Existing Methods: Domain Based, Mandala et. al. Example

Then, all combinations of thesauri are considered as an ensemble to expand queries to an *arbitrary* length of 20 terms for evaluation.

Topic Type	Base	Expanded with						
		WordNet only	Head-Mod only	Cooccur only	WordNet+ Head-Mod	WordNet+ Cooccur	Head-Mod+ Cooccur	Combined method
Title	0.1175	0.1299 (+10.6%)	0.1505 (+28.1%)	0.1637 (+39.3%)	0.1611 (+37.1%)	0.1698 (+44.5%	0.1859 (+58.2%)	0.2337 (+98.9%)
Description	0.1428	0.1525 (+6.8%)	0.1705 (+19.4%)	0.1950 (+33.4%)	0.1832 (+28.3%)	0.1973 (+38.2%)	0.2315 (+62.1%)	0.2689 (+88.3%)
All	0.1976	0.2018 (+2.1%)	0.2249 (+13.8%)	0.2395 (+21.2%)	0.2276 (+15.2%)	0.2423 (+22.6%)	0.2565 (+29.8%)	0.2751 (+39.2%)

Titles are the shortest text available in this corpus, and we see that from a base of ≈ 0.12, the combined weighted ensemble of thesauri accomplish a 98.9% improvement.

Existing Methods: Domain Based

The domain-based class of algorithms has the following drawbacks:

1. **the corpus likely does not match the experimental dataset (e.g. WordNet, Wikipedia) in style or sophistication**;
2. due to potential mismatches in the data, distance metric selection becomes too much of an art and/or arbitrary;
3. again, sampling is done greedily rather than probabilistically ("always pick the best term");
4. requires feature generation likely to be costly (e.g. part-of-speech).

Existing Methods: Lexical Modeling

Methods in the lexical modeling category focus more on tackling the short text issue head-on within a *model* rather than within the data or the features.

Most methods reviewed in this category use either Bayesian topic models, and neural networks are the next frontier.

Existing Methods: Lexical Modeling, Yan et. al. Example

One such model by [14] modifies Latent Dirichlet Allocation[1]:

- by modeling *biterms* rather than individual words. Biterms are collections of 2 words that may appear anywhere in a sentence (i.e. "`i visit apple store`" $\rightarrow$ {`"visit apple", "apple store", "visit store"`})
- researchers believed that such a method models cooccurrence *patterns* and not just cooccurences.
- moreover, the model considers biterms, not words in documents, to be generated by a topic.

This model is called BTM, for Biterm Topic Model.

Existing Methods: Lexical Modeling, Yan et. al. Example

Researchers found that based on *coherence*, BTM performed better than LDA even on longer texts.

But...

Table 6: Average coherence score on the top T words (ordered by $P(w|z)$) in topics discovered by LDA, LDA-U, mixture of unigrams, and BTM. A larger coherence score means the topics are more coherent. It suggests that BTM outperforms others significantly (P-value < 0.01 by t-test).

T	5	10	20
LDA	-55.0 ± 0.4	-236.4 ± 2.0	-1015.7 ± 5.9
LDA-U	-54.2 ± 0.8	-234.8 ± 1.1	-1009.4 ± 4.4
Mix	-53.8 ± 0.1	-233.0 ± 1.4	-1007.6 ± 6.7
BTM	$\mathbf{-52.4 \pm 0.1}$	$\mathbf{-227.8 \pm 0.3}$	$\mathbf{-990.2 \pm 3.8}$

TABLE 8
Time cost (seconds) per iteration of BTM and LDA on Tweets2011 collection.

K	50	100	150	200	250
LDA	38.07	74.38	108.13	143.47	178.66
BTM	128.64	250.07	362.27	476.19	591.24

TABLE 9
Memory cost (megabytes) per iteration of BTM and LDA on Tweets2011 collection.

K	50	100	150	200	250
LDA	3177	5524	7890	10218	12561
BTM	927	946	964	984	1002

Existing Methods: Lexical Modeling

The lexical modeling class of algorithms has the following drawbacks:

1. parameter estimation for such models can be **very slow**;
2. it is the researcher's opinion that building a full-blown model is overkill for this problem;
3. biterms seem arbitrary – why not triterms? [4]
4. native neural networks tend to suffer from overfitting problems[12].
5. Bayesian topic models have mixed success in scalability.

[4] many methods in this category use *biterms*.

Motivation: So, Why Yet Another Method?

The previous work raises the following questions:

1. Can't we just use the existing terms in the text rather than using other data or feature generation?
    - this is faster and more computationally efficient;
    - we do not have to worry about concept drift, transfer or lexicon mismatch since we use the same data;
    - we can make fewer major arbitrary decisions (i.e. distance metrics);
2. Can we avoid building a special model and treat text augmentation as a preprocessing step?
3. Can we propose something more general purpose?

The researcher believes that we already have the statistical tools to do so...

1 Introduction

2 Existing Methods

3 Semantic Bootstrap Proposal

4 Experiments

5 Data

6 Models

7 Results

8 Conclusion

The Classical Bootstrap

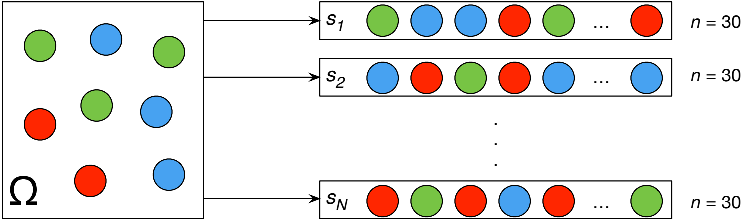
The classical bootstrap[4] is a technique for constructing the sampling distribution of a sample statistic to calculate an estimate or its confidence interval. It consists of

1. Draw a sample of size n from some population where the observations are
   1. with replacement;
   2. independent;
   3. of a sufficiently large size.
2. Compute some function of the data f on such sample.
3. Repeat steps 1 and 2 for a total of N times, where N is large.

By the Central Limit Theorem, the metrics calculated by f follow a normal distribution.

The Bootstrap for Small Samples

While the bootstrap is most often used to estimate a sampling distribution of a parameter, it has also been used when a sample is **small**, with mixed results[11][3].



The Bootstrap for Small Samples

But how small is too small? One author [3] has stated **8** to be an approximate lower bound.

But, many texts are even *shorter* than 8 words. In my experimental data, the average is 4-5.

Research Question: Can we modify the Bootstrap to work with text?

Doing so requires modifications to the Bootstrap procedure.

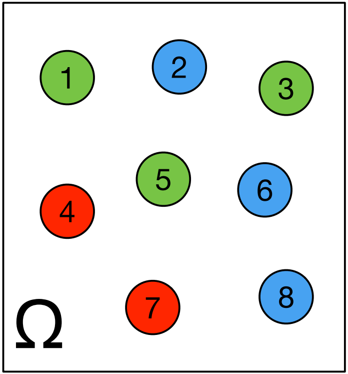
Motivation: Why Use the Bootstrap?

Several advantages to the Bootstrap over existing methods:

1. very easy to implement;
    - improved scalability and embarrassingly parallel;
2. very accessible to practitioners even outside machine learning;
3. uses the existing data as a population rather than external data or queries;
4. makes no transformations to individual observations (only in their dependence and collection);
5. the Bootstrap is one of the most popular simulation methods for working with sample sizes.
    - Regularization is another common method;
    - A general purpose method that does not require special data or special models.

How Would a Bootstrap for Text Look?

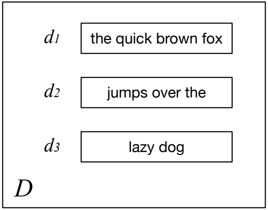
First we need to define, what is an observation, and what is the population?

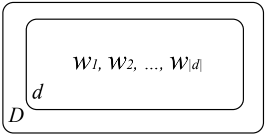


With a standard dataset, the dataset itself is the population, and each ball is an observation to be sampled. There is a simple nesting of observations within data.

How Would a Bootstrap for Text Look?

The situation is not so simple for text. We have an inherent nesting of *words* within *documents* within a *corpus*.

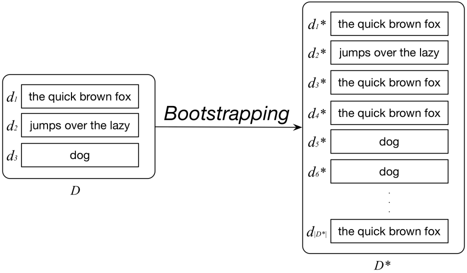




We have a few choices of how to define an *observation* and *population*.

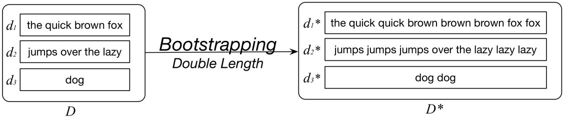
How Would a Bootstrap for Text Look?

Option 1 If we *directly* apply the resampling from the bootstrap on the corpus level, treating the corpus as the population, and the documents as the observations, we end up just duplicating existing documents.



How Would a Bootstrap for Text Look?

Option 2 If we *directly* apply the resampling from the bootstrap on the document level, treating the document as the population, and the words as the observations, we end up just duplicating existing words.

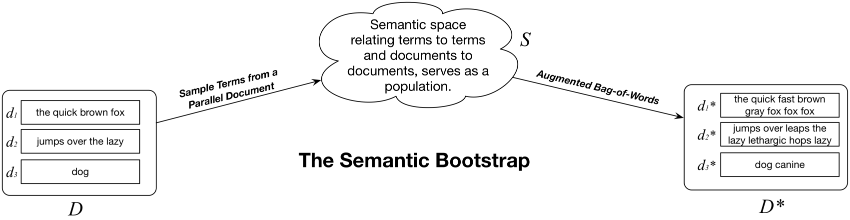


While duplicating words increases sample size, it does nothing to address sparsity.

How Would a Bootstrap for Text Look?

Proposed Variation of Option 2: The Semantic Bootstrap

Suppose instead we sample *indirectly* from the document by choosing words that are semantically similar to the existing words in each document.



The Semantic Bootstrap

Semantic Bootstrap as a Variation of Classical Bootstrap

What we just described can be summarized as the classical bootstrap with three variations:

1. only the resampling step is of interest;
2. treat each term in a document as an observation;
3. treat a semantic space $S|d$ as the population;
4. independence is lost as certain words are more discriminant than others

But what is the semantic space S?

The Semantic Space S

We must define the population – the semantic space S.

Belief: any matrix factorization that maintains distance between terms, documents and term/document pairs.

For this research, we use *Latent Semantic Analysis (LSA)*, a very commonly used matrix factorization in text mining and natural language processing.

The Semantic Space S: Latent Semantic Analysis

We start by representing a *training* corpus as a term-document matrix $\mathbf{X}$ with $|D|$ rows, one for each document d_i, and $|V|$ columns, one for each term t_j.

$$\mathbf{X} = \begin{bmatrix} t_{0,0} & t_{1,0} & t_{2,0} & \cdots & t_{|V|,0} \\ t_{0,1} & t_{1,1} & t_{2,1} & \cdots & t_{|V|,1} \\ \ldots & \ldots & \ldots & \vdots & \vdots \\ t_{0,|D|} & t_{1,|D|} & t_{2,|D|} & \cdots & t_{|V|,|D|} \end{bmatrix}$$

Each entry in the matrix X_{ij} represents some relationship between each term in each document: presence/absence (0/1), word count, or...

The Semantic Space S: Latent Semantic Analysis

A common metric relating terms and documents is TF-IDF[5] and is the score used for X_{ij}.

$$X_{ij} = \text{TF-IDF}_{ij} = \text{TF}_{ij} \times \text{IDF}_i = c_{ij} \times \left[\log\left(\frac{|D|}{|\{d : t_i \in d\}|}\right) + 1\right]$$

Properties of TF-IDF

- high TF-IDF is associated with highly discriminative or influential words within a document;
- low TF-IDF is associated with stopwords and other words that do not impart much meaning in the document.

[5] where c_{ij} is the number of times t_i appears in document d_j, $|D|$ is the number of documents, and $\{d : t_i \in d\}$ is the set of documents containing t.

The Semantic Space S: Latent Semantic Analysis

Then we can decompose $\mathbf{X}$ as follows:

$$\mathbf{X} \approx \mathbf{D_k \Sigma_k W_k}^{\mathbf{T}}$$

We can then construct term and document similarities as follows

$$\begin{aligned}
\mathbf{S_t} &= \mathbf{X X^T} \\
&= \left(\mathbf{D_k \Sigma_k W_k}^{\mathbf{T}}\right)\left(\mathbf{D_k \Sigma_k W_k}^{\mathbf{T}}\right)^{T} \\
&= \mathbf{D_k \Sigma_k W_k}^{\mathbf{T}} \mathbf{W_k \Sigma_k^T D_k^T} \\
&= \mathbf{D_k \Sigma_k^2 D_k^T}
\end{aligned}$$

and similarly,

$$\mathbf{S_d} = \mathbf{X^T X} = \mathbf{W_k \Sigma_k^2 W_k^T}$$

The Semantic Space S: Cosine Similarity

Once we have computed $\mathbf{S_t}$ and $\mathbf{S_d}$, we can make comparisons across terms and documents using a distance metric. Cosine similarity is perhaps the most common metric used in text mining and NLP.

$$\cos\theta = \frac{u \cdot v}{||u||||v||}$$

For terms,

$$\delta_{t_i,t_j} = \mathrm{sim}(t_i, t_j) = \frac{t_i \cdot t_j}{||t_i||||t_j||}$$

For documents,

$$\delta_{d_i,d_j} = \mathrm{sim}(d_i, d_j) = \frac{d_i \cdot d_j}{||d_i||||d_j||}$$

A Sampling Scheme

Up to this point, we have decomposed $\mathbf{X}$ into a semantic space S using SVD and computed pairwise similarities using cosine similarity. We also have information about the relationship between a document and its terms in X_{ij}, the TF-IDF score.

We can now describe a sampling scheme:

1. we have a probability distribution over term similarities δ_t,
2. we have a probability distribution over document similarities δ_d,
3. we can also have a probability distribution over word discrimination $X_{i\cdot}$.

A Conditional Probability Distribution over S

We can now propose a sampling from S:

$$P\left(t_j|t_i\right) = \frac{\delta_{t_i,t_j} + |\min \delta_{t_i,\cdot}|}{\sum_m \left(\delta_{t_i,t_m} + |\min \delta_{t_i,\cdot}|\right)}$$

$$P\left(d_j|d_i\right) = \frac{\delta_{d_i,d_j} + |\min \delta_{d_i,\cdot}|}{\sum_m \left(\delta_{d_i,d_m} + |\min \delta_{d_i,\cdot}|\right)}$$

A Probability Distribution over d

Within each document d, each term has a different level of discriminative power quantified by TF-IDF. So if we want to select a random term based on discriminative power, we use X_{ij} as follows

$$P(t_i|d_j) = \frac{x_i}{\sum_i x_i}$$

The Semantic Bootstrap Algorithm

Now that we have a population S to sample new words from, and a probability distribution induced from it, and we also have a probability distribution over the terms in a document d, we can propose a sampling scheme as follows...

Let ε be the *augmentation rate* – a value greater than 1, specifying how much longer the new document d^* should be relative to the original document d:

$$d^* = \varepsilon|d|$$

The Semantic Bootstrap Algorithm

Proposal: We sample terms from S until the new augmented document reaches the desired length. That is, while $|d^*| < \varepsilon |d|$:

1. Pick a target term t in d according to $P(t_i|d_j)$. This yields a term with high discriminating value with highest probability. We want to find a word similar to it *given* the context of the current document.
2. Pick a candidate document d' according to $P(d_j|d_i)$. This yields a document d' that is most similar to d according to S, with highest probability.
3. Pick a candidate term t' from d' according to $P(t_j|t_i)$. This yields the most semantically related term from the most semantically related document, with highest probability.

Using probabilities rather than simply picking the most relevant terms and documents reduces the possibility of bias from a bad selection. It also eliminates the need for arbitrary ranking cutoffs.

The Semantic Bootstrap Algorithm: A Final Sanity Check

In [13], the researchers did one final check when augmenting terms to protect against *concept drift*, where selecting a bad term causes the results to be irrelevant.

- In this research, all samplings are based on d, and not on iterations of d^*, so concept drift is not possible, **BUT**
- A poor sampling will still yield bad results.

We want to make sure that any sampled terms contribute as much semantic cohesion as possible. So...

The Semantic Bootstrap Algorithm: Mutual Information

We use Mutual Information (MI) as a final check to try to ensure the augmented bag-of-words d^* is as semantically related to d as possible. The higher the change in MI, the more candidate term t' adds to semantic cohesiveness.

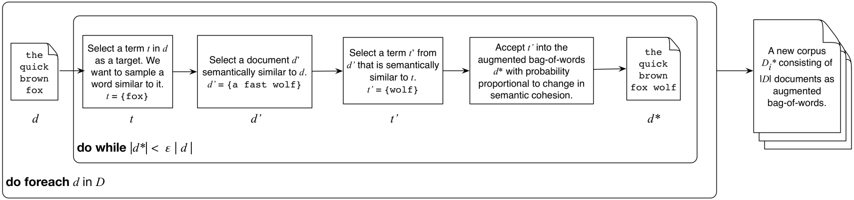
$$I(d) = \sum_{(t_i, t_j, i \neq j) \in d} P(t_i, t_j) \log \left(\frac{P(t_i, t_j)}{P(t_i,)P(t_j)} \right)$$

We accept the term t' into d^* according to the transition probability

$$P(d \to d^*) \propto \min \left\{ 1, \frac{1}{Z} \exp \left(I(d^*) - I(d) \right) \right\}$$

The Semantic Bootstrap Algorithm

The Semantic Bootstrap algorithm can be illustrated as the following flowchart



ε is referred to in this research as the *augmentation extension* and measures $|d^*|$ relative to $|d|$ (e.g. $\varepsilon = 2$ means doubling the length).

The Semantic Bootstrap Algorithm: A Quick Example

Suppose we have the following text we want to augment.

```
Yom Kippur, Tel Aviv style
```

The text is typically short and vague that is also not a complete sentence. It could be about the Jewish *religion*, it could be about *politics* in the region, or it could be more *geographical/cultural* in nature. Maybe we can augment it with more terms to assist in classification.

The Semantic Bootstrap Algorithm: A Quick Example

First we preprocess the text and turn it into a bag of words representation:

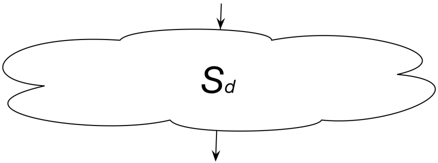
`yom kippur tel aviv style`

Next, we *probabilistically* select a document d' from semantic space S according to how similar it is to d.

The Semantic Bootstrap Algorithm: A Quick Example

The d' selected here had a high cosine similarity to d and was chosen with probability 0.878.

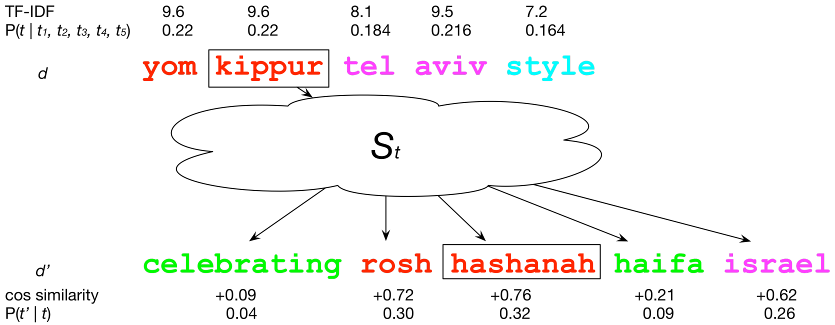
d	yom	kippur	tel	aviv	style
TF-IDF	9.6	9.6	8.1	9.5	7.2
$P(t \mid t_1, t_2, t_3, t_4, t_5)$	0.22	0.22	0.184	0.216	0.164



d'	celebrating	rosh	hashanah	haifa	israel
TF-IDF	5.3	9.1	9.4	8.9	7.8
$P(t \mid t_1, t_2, t_3, t_4, t_5)$	0.13	0.225	0.232	0.22	0.19

The Semantic Bootstrap Algorithm: A Quick Example

Next, from d, we sample a target word t. We will then sample a word t' from d' according to how similar they are in the semantic space S.



The Semantic Bootstrap Algorithm: A Quick Example

Now we have a candidate document d^*. We accept the new term t' and thus d^* according to the change in semantic cohesiveness, $P(d \to d^*)$:

d^* **`yom kippur tel aviv style`** $\boxed{\texttt{hashanah}}$

- If $P(d \to d^*)$ is relatively "large", then accept t' into d^*.
- Otherwise, reject t' and keep d.

The Semantic Bootstrap Algorithm: A Quick Example

The sampling process continues until the desired number of terms has been added.

Some minutiae:

- It is possible to select the same document d' across iterations;
- It is possible to select the same t and/or t' across iterations. The final term acceptance step had some effect in preventing these words from overwhelming d^*;
- It is possible to get into a situation where d^* is repeatedly rejected. It was found that as many as 5 retries for each iteration were required for as few as 0.1% of cases.

1 Introduction

2 Existing Methods

3 Semantic Bootstrap Proposal

4 Experiments

5 Data

6 Models

7 Results

8 Conclusion

Experimental Questions

The most important question we will review is obviously

Does the semantic bootstrap even work?

And more specifically, the following

1. How many terms should be sampled to yield the best resuilts? In other words, what are appropriate values for augmentation rate ε?
2. When should the Semantic Bootstrap algorithm be applied? In the training data? In the testing data? Or both, as is typical?
3. Is the final probabilistic term acceptance step truly necessary?
4. Which of the experimental classification models works best?
5. How can we control for sampling variation?

A Note on Unmatched Training/Testing Sets

It is standard practice to apply the same transformations to both the training and testing sets.

There are cases where it may be more convenient from a computational standpoint to use a unmatched training or testing set.

Some research suggests that unmatched datasets can sometimes outperform matched datasets[5].

A Note on Unmatched Training/Testing Sets: Example I

Augmenting only the Training Data

Suppose I stand in a noisy room and there is a microphone at the far end of the room that is to record only my voice.

One change we can make is to the *microphone* and its logic, making it more sensitive to the frequencies in my voice. This is akin to training a classifier on an augmented corpus and evaluating on a the standard dataset.

A Note on Unmatched Training/Testing Sets: Example II

Augmenting only the Unseen Testing Data

Suppose I stand in a noisy room and there is a microphone at the far end of the room that is to record only my voice.

Another change we can make is to leave the microphone as it is, and for me to yell loudly over the noise in the room. This is akin to augmenting the unseen signal being classified while leaving the training set the same.

A Note on Unmatched Training/Testing Sets

Throughout this research, these configurations are called variations:

Variation	Description
SB1	Augment only the training data, use original testing data.
SB2	Use original training data and classifier, augment only testing data.
SB12	Matched case; augment both sets.
SB0	Raw data; neither is augmented.

Controlling for Sampling Variation

The Semantic Bootstrap, and the Classical Bootstrap are both probabilistic. Each iteration yields a different sampling. Each iteration of the Semantic Bootstrap yields a new corpus containing augmented documents.

Because of this, the Semantic Bootstrap is applied for 100 iterations and performance metrics averaged over all iterations.

The iterations are for this research only, and not intended for actual use.[6]

[6] Picking an appropriate number of iterations should be future work.

1. Introduction
2. Existing Methods
3. Semantic Bootstrap Proposal
4. Experiments
5. Data
6. Models
7. Results
8. Conclusion

Technorati Data

The experimental data from a 2014 widespread crawl of the Technorati blog aggregator and search engine. **The titles of each blog post were used as the dataset for this research.**

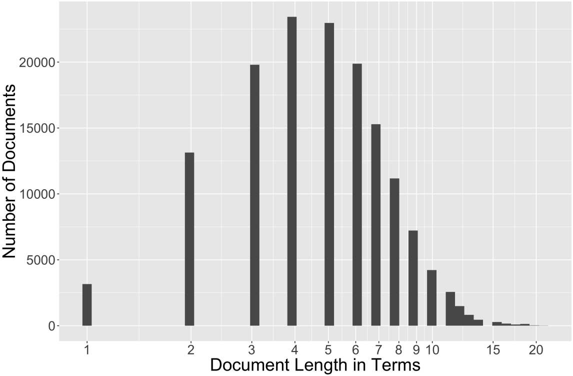
Each blog belongs to one[7] of ten categories.

Category	% of Titles	Category	% of Titles
autos	2.1	politics	7.5
business	7.4	science	3.0
entertainment	20.8	sports	14.3
green	2.1	technology	18.8
living	21.6	overall	2.4

[7] Blogs belonging to more than one category were removed from consideration.

Technorati Data

On average, blog post titles contain between 4 and 5 words, and the length is log-normal distributed.



Technorati Data

The text was cleaned using typical text mining procedures

- discard titles not written in English;
- converting foreign characters to ASCII where possible;
- removing known stopwords (i.e. `a`, `the` etc.);
- removing words that appear too often or too seldom to be useful;
- removing duplicates and empty documents caused by the above.

The dataset consisted of

- Corpus consisted of **131,519** documents.
- Lexicon/vocabulary size consisted of **10,889** words.
- Dimension of term-document matrix reduced to $K = 500$ from scree analysis.

1 Introduction

2 Existing Methods

3 Semantic Bootstrap Proposal

4 Experiments

5 Data

6 Models

7 Results

8 Conclusion

Experimental Models

To test the efficacy of the Semantic Bootstrap as a preprocessing step for classification, the researcher used two models:

1. Linear Support Vector Machine (SVM) [Classical Approach]
   - via `libshorttext`, a variation designed specifically for short texts, and one state-of-the-art for this research
   - linear SVM with L2 penalty and word count as features
2. Latent Dirichlet Allocation Variation (LDA) [Topic Model Approach]
   - Supervised Latent Dirichlet Allocation (sLDA), $K = 50, \alpha = 1, \eta = 1$

Individual classifiers were constructed for each category and evaluated. The resolution of such a one vs. rest classifier is saved as future work.

Supervised Latent Dirichlet Allocation (sLDA)

A supervised variant of LDA exists that can remedy the problem of inconsistent labeling by using category labels as a dependent variable.

We can represent a Generalized Linear Model (GLM) as a linear combination of coefficients β_i and K topics X_i, where each topic consists of a set of terms. In theory, many different link functions can be used. For this research, we used the implemented **logit** link.

`libshorttext`: Linear SVM for Short Texts

`libshorttext` is a package and variation of SVM developed specifically for use with small texts. It supports

- stemming
- stopword removal
- construction of *n*-grams
- **feature normalization/standardization**
- L1 and L2 regularization
- several feature types: binary, word count, term frequency (TF), TF-IDF.

And supports related models using similar optimization problems

- logistic regression
- multiclass SVM

libshorttext: Linear SVM for Short Texts

While there are several different models implemented in libshorttext, the **Linear SVM with L2 penalty with Word Count** features performed near the best on precision/recall/F1 score[8] and *also uses the same feature type as the sLDA model.*

Model / Metric	Accuracy Macro	Accuracy Micro	Precision Macro	Recall Macro	F1 Score Macro
Linear SVM L2, TF-IDF	0.911	0.696	0.720	0.624	0.668
Linear SVM L2, Binary	0.889	0.645	0.697	0.628	0.644
Linear SVM L2, Word Count	0.832	0.627	0.725	0.638	0.679
Linear SVM L2, Term Frequency	0.899	0.587	0.701	0.628	0.663
Linear SVM L1, TF-IDF	0.894	0.628	0.703	0.619	0.638
Linear SVM L1, Binary	0.933	0.614	0.691	0.625	0.640
Linear SVM L1, Word Count	0.885	0.622	0.691	0.625	0.639
Linear SVM L1, Term Frequency	0.900	0.601	0.691	0.626	0.641
Logistic Regression, TF-IDF	0.931	0.623	0.700	0.624	0.642
Logistic Regression, Binary	0.840	0.675	0.700	0.619	0.639
Logistic Regression, Word Count	0.898	0.622	0.700	0.620	0.639
Logistic Regression, Term Frequency	0.849	0.567	0.700	0.620	0.639

[8] the researcher used precision/recall and F1 score to assist in making the decision because the dataset is imbalanced.

1 Introduction

2 Existing Methods

3 Semantic Bootstrap Proposal

4 Experiments

5 Data

6 Models

7 Results
- Effect of ε
- Effect of Time of Sampling
- Effect of the Probabilistic Term Acceptance Step
- Effect of Model Choice

8 Conclusion

Results

In this section, we present the results from applying the Semantic Bootstrap to the Technorai data. As a reminder, we studied the following questions

1. **How much should each document be augmented by?** What are appropriate values for ε?
2. **When should the Semantic Bootstrap be applied**: to both the training and testing sets? Only the training set? Or only in the unseen set?
3. **Is the final probabilistic term acceptance step necessary?**
4. **Which experimental model works better under the Semantic Bootstrap? SVM or sLDA?**

Results

As a result, we also implicitly answer the following important questions via the above questions

1. **Does the Semantic Bootstrap even work?**
2. **Is a matrix factorization (LSA) an appropriate semantic space S?**

A Note on Performance Metrics

- **sLDA** *natively* supports posterior probabilities, so **AUC** is used as the performance metric for comparing sLDA models.
- **SVM** does not, so **F1** score is used instead for comparing SVM models.
- F1 score is used when comparing **SVM** to **sLDA**

This is fine because we are mostly interested in comparing performance with and without the Semantic Bootstrap.

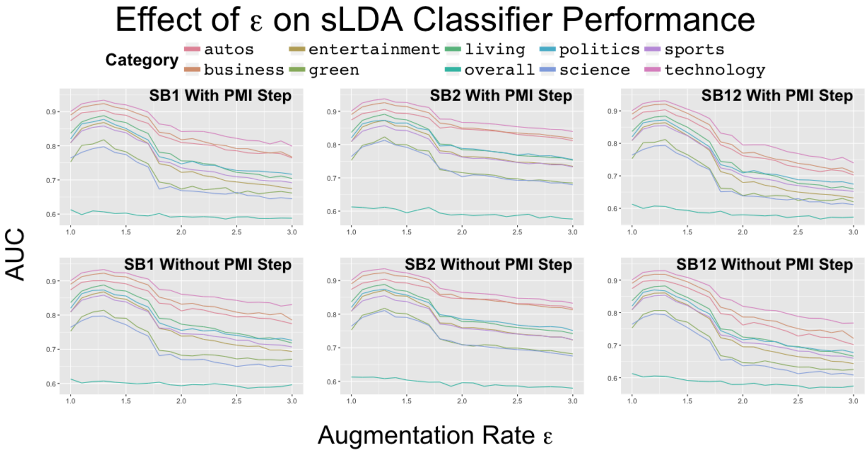
Experiment 1: Effect of ε

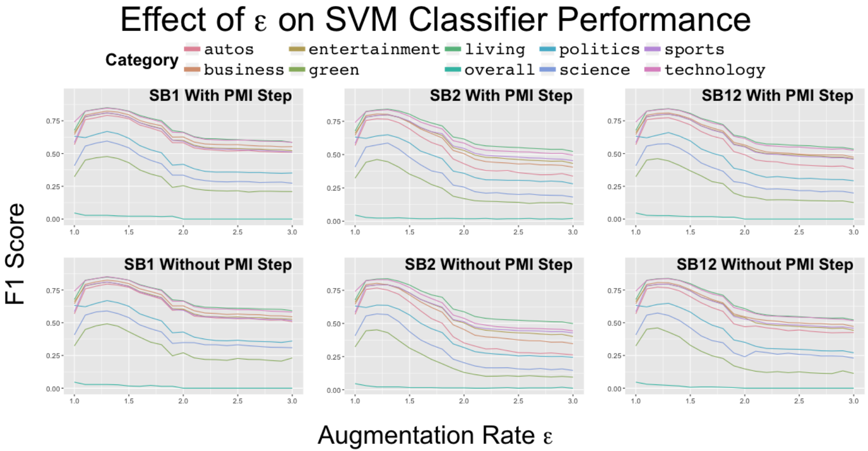
Hypothesis:

- a value such as 1.5 should ensure that a minimum of 1 term is chosen as a candidate for sampling.
- a value higher than 2.0 adds more words than there existed in original d, introducing noise and bias.
- $1.5 \leq \varepsilon^* < 2$

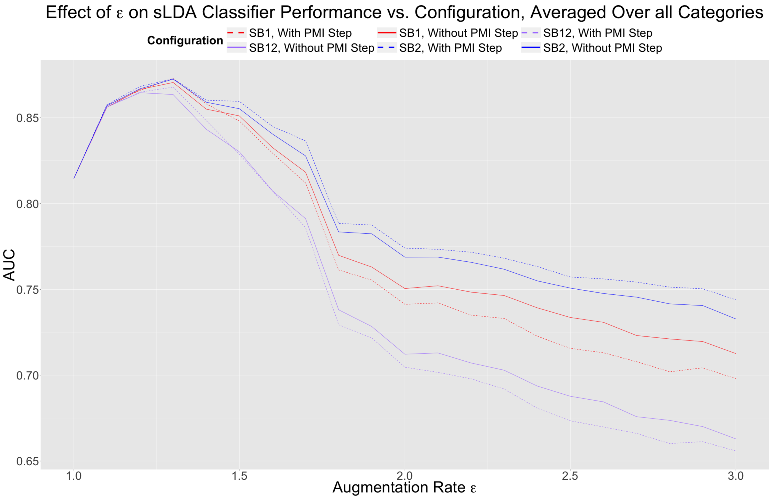
In the graphs that follow, $\varepsilon = 1.0$ represents the raw, un-augmented data (`SB0`).

Effect of ε

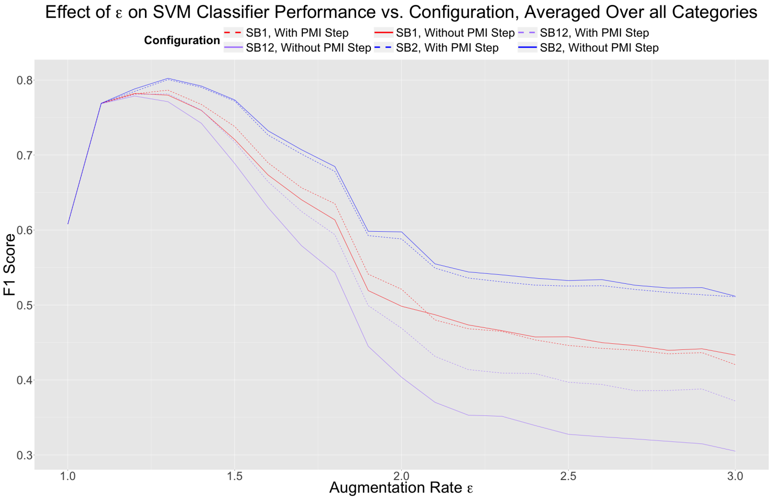



Effect of ε on SVM Classifier Performance
Category
autos
business
entertainment
green
living
overall
politics
science
sports
technology
SB1 With PMI Step
SB2 With PMI Step
SB12 With PMI Step
SB1 Without PMI Step
SB2 Without PMI Step
SB12 Without PMI Step
F1 Score
0.00
0.25
0.50
0.75
1.0
1.5
2.0
2.5
3.0
Augmentation Rate ε

Effect of ε



Effect of ε



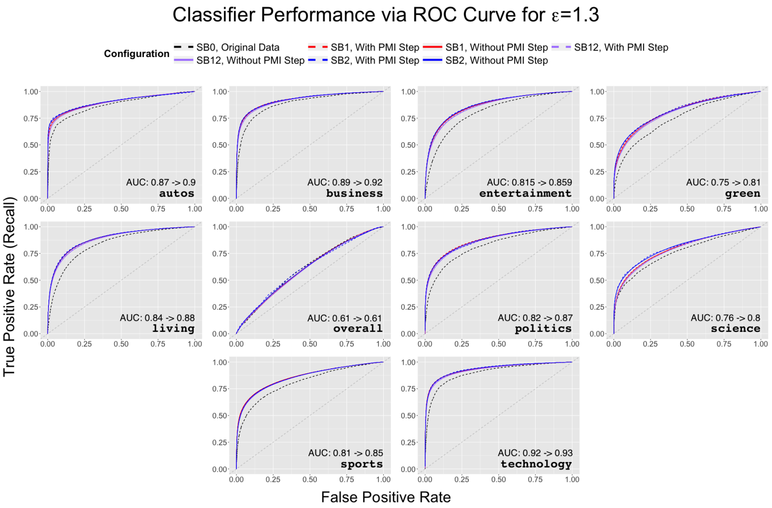
Experiment 1 Conclusion

The hypothesis of $1.5 \leq \varepsilon^* < 2$ was disproven:

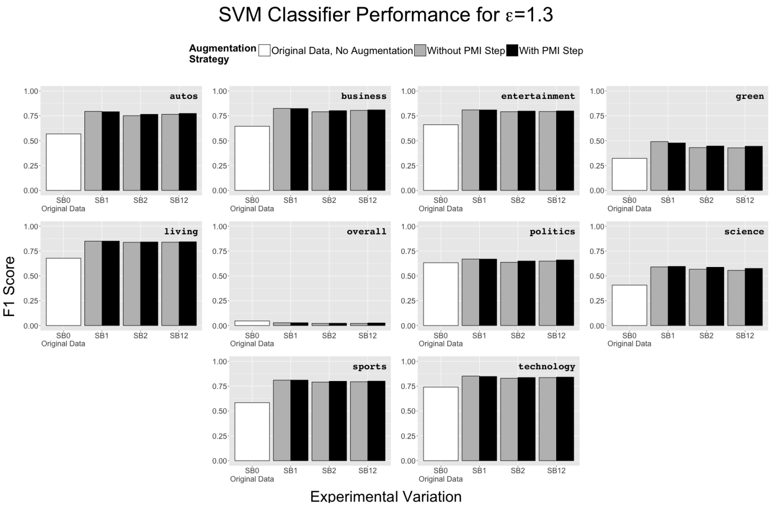
1. The optimal value of ε is approximately 1.3.
2. $1.5 \leq \varepsilon \leq 1.7$ performance is approximately equal to baseline.
3. $\varepsilon > 1.7$ performance begins to drop substantially.

Since the average document in the corpus had 4 to 5 terms, $\epsilon = 1.3$ corresponds to sampling approximately **one** term. This does validate that sampling from S yields highly discriminative terms that help performance.

Effect of $\varepsilon^* = 1.3$ on Classifier Performance



Effect of $\varepsilon^* = 1.3$ on Classifier Performance

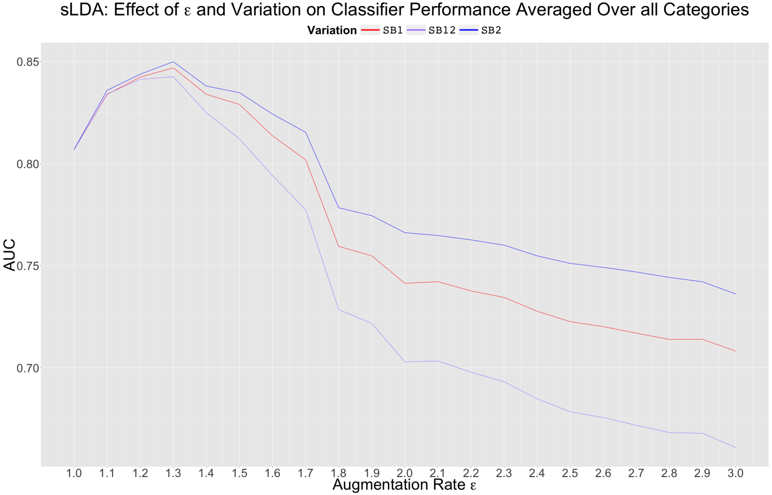


Experiment 2: When Should the Semantic Bootstrap be Applied?

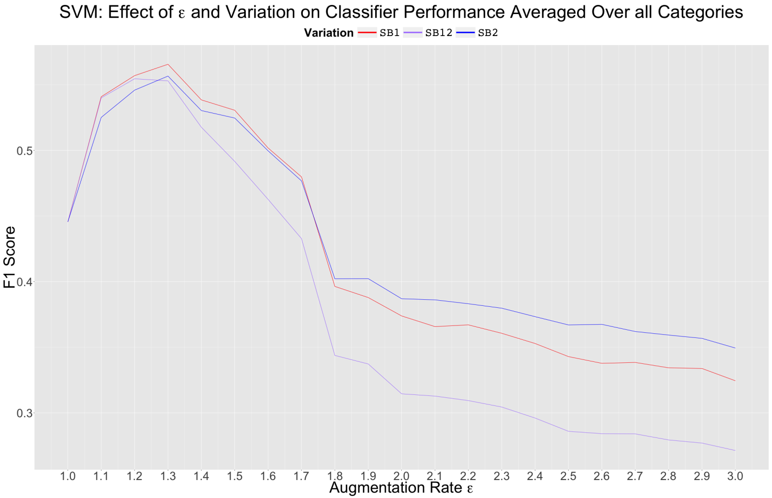
Hypothesis

- Matched training and testing sets are the norm.
- There is some research suggesting unmatched sets may be better.
- In industry, we have used unmatched sets with success.
- No strong hypothesis; but interesting to study.

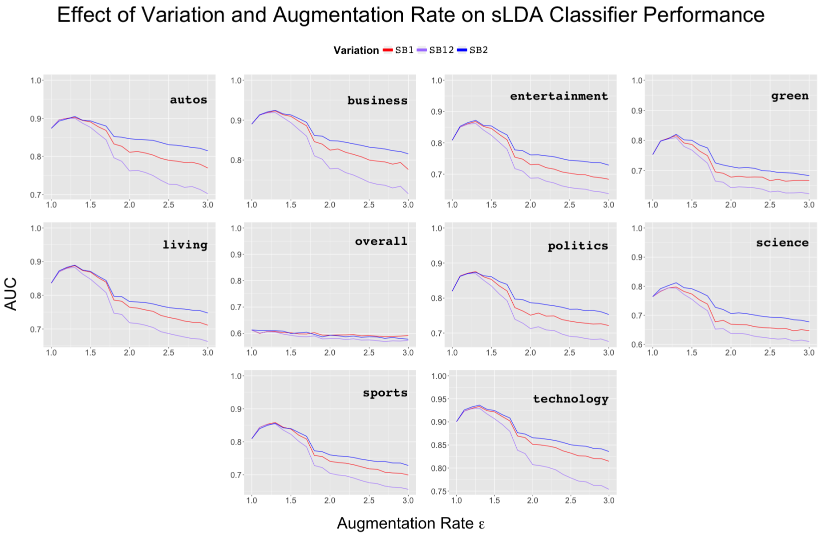
Effect of Time of Sampling

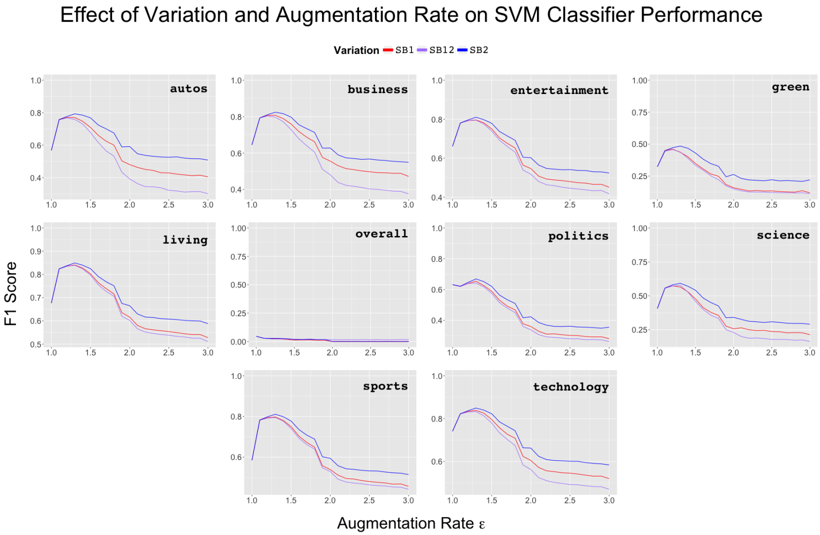


Effect of Time of Sampling

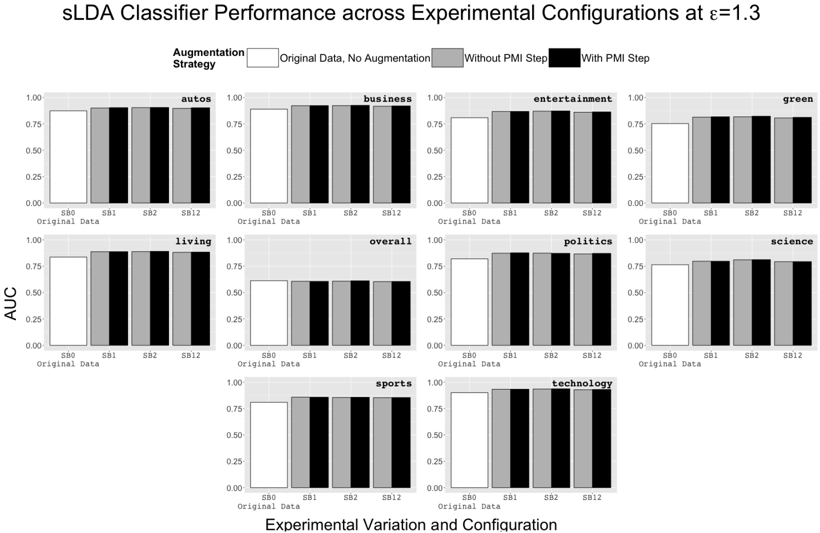


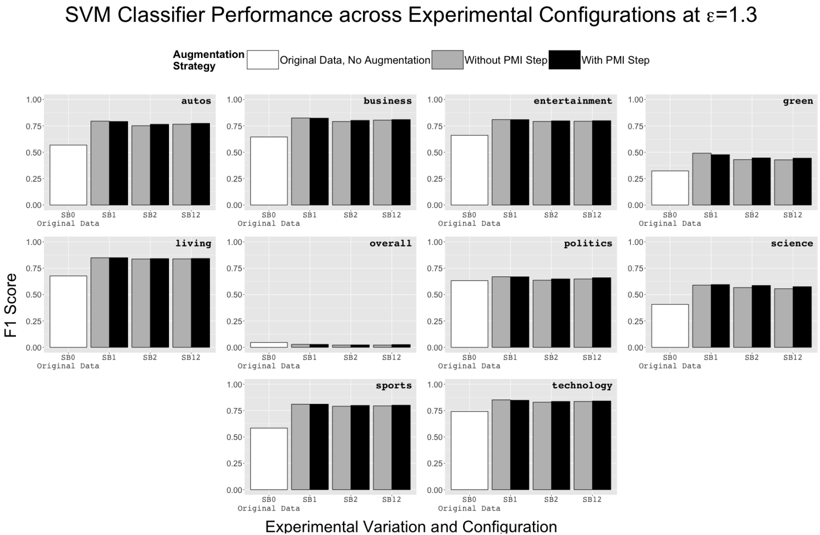
Effect of Time of Sampling




Effect of Variation and Augmentation Rate on SVM Classifier Performance
Variation SB1 SB12 SB2
autos
business
entertainment
green
living
overall
politics
science
sports
technology
F1 Score
Augmentation Rate ε

Effect of Time of Sampling



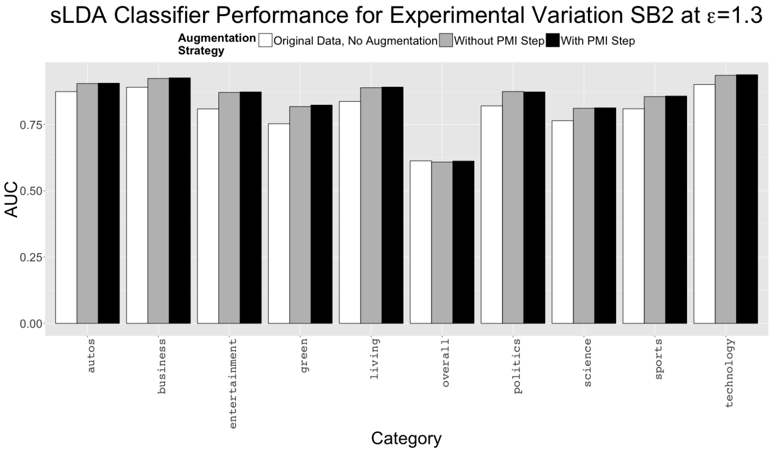

SVM Classifier Performance across Experimental Configurations at ε=1.3
Augmentation Strategy
Original Data, No Augmentation
Without PMI Step
With PMI Step
autos
business
entertainment
green
living
overall
politics
science
sports
technology
1.00
0.75
0.50
0.25
0.00
SB0 Original Data
SB1
SB2
SB12
F1 Score
Experimental Variation and Configuration

Experiment 2 Conclusion

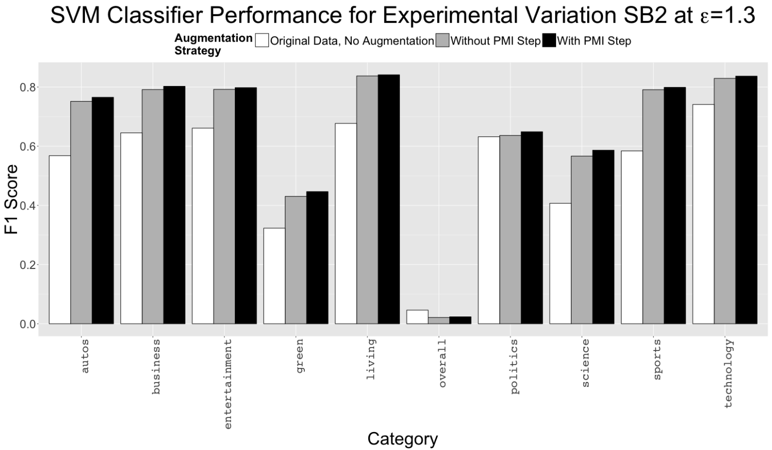
Results among the `SBx` variations were mixed:

1. At small ε and ε^*, differences were negligible.
2. For sLDA, `SB2` consistently performed better as ε increased.
3. For SVM, `SB2` performed best at large $\varepsilon \geq 1.8$.
4. `SB2` performed the best at higher values for ε.
5. **Bottom Line:** Significant augmentation performs best when performed only on the testing set.
6. **Remarkably:** The matched training/testing set scenario, the status quo, consistently performed the worst.

Effect of Time of Sampling



Effect of Time of Sampling

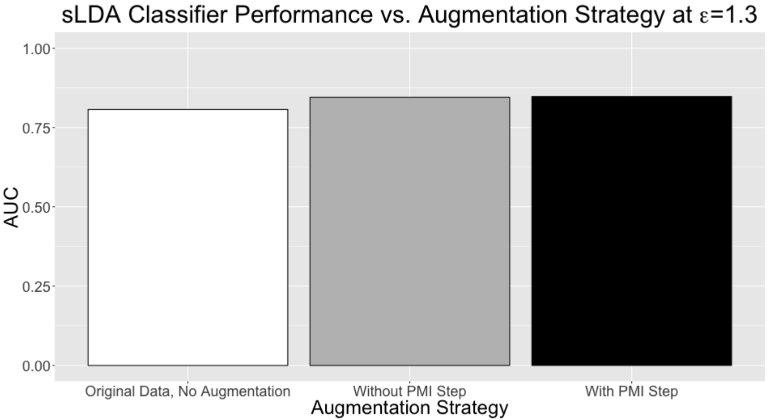


Question 3: Does the Final Term Acceptance Step Help?

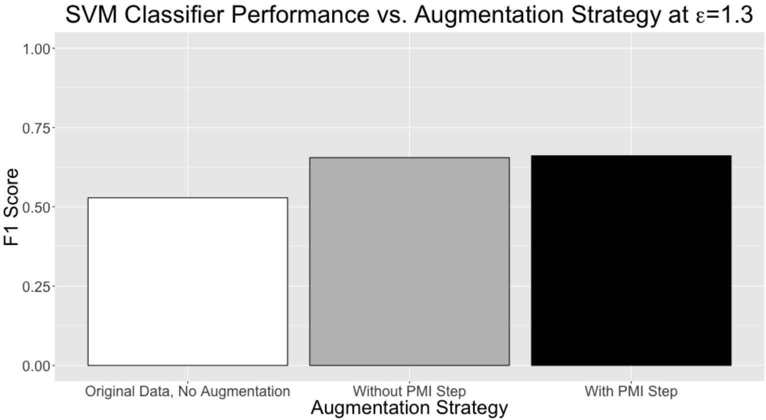
Although we sample words from S according to semantic similarity, we want to make sure that the terms we sample improve semantic cohesiveness.

Hypothesis
The hypothesis is that this final term acceptance step should improve performance because only words that improve cohesiveness are accepted whereas words that do not will be accepted with a lower probability.

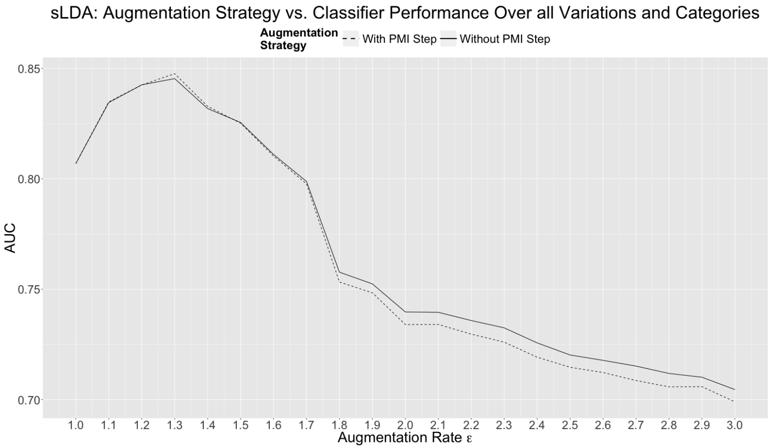
Effect of the Probabilistic Term Acceptance Step



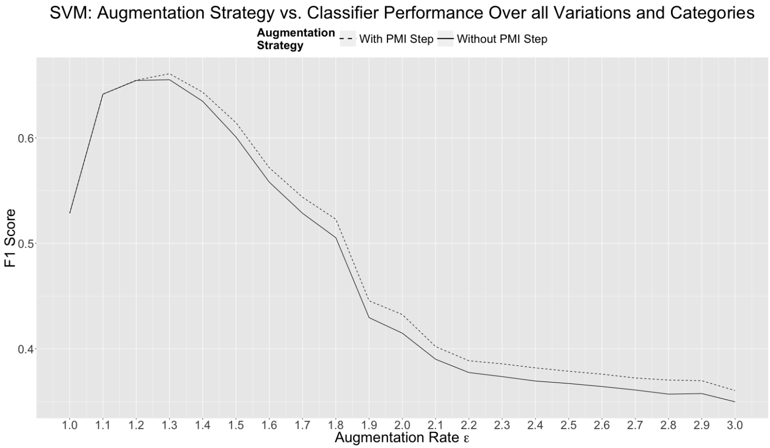
Effect of the Probabilistic Term Acceptance Step



Effect of the Probabilistic Term Acceptance Step



Effect of the Probabilistic Term Acceptance Step



Experiment 3 Conclusion

The final probabilistic term acceptance yielded no practical difference.

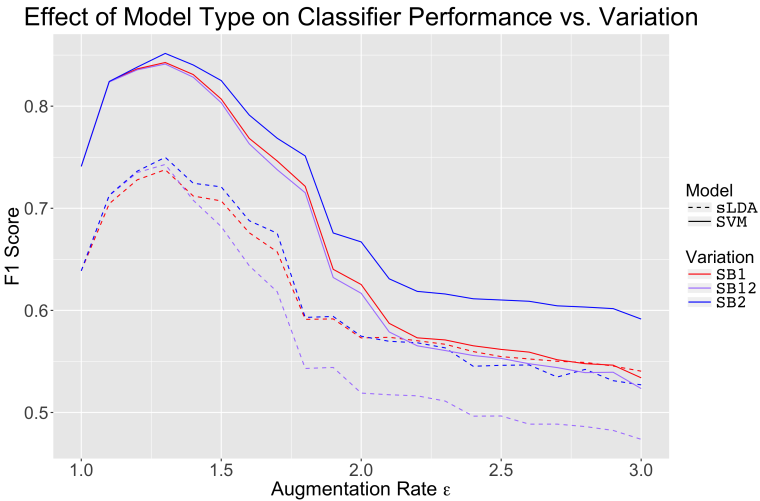
- For sLDA, straight-up sampling from S performed only 1% better on AUC than with the final check for $\varepsilon \geq 1.8$.
- For SVM, the final check again only yielded a 1% improvement in F1-score, but across all ε.

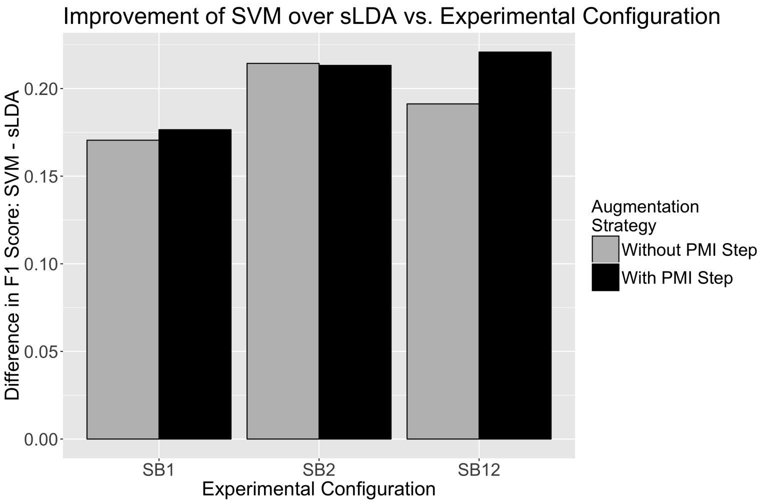
Likely Reason: The Semantic Bootstrap sampling step already does a good enough job ensuring semantic cohesiveness since both terms are sampled conditioned on a similar document.

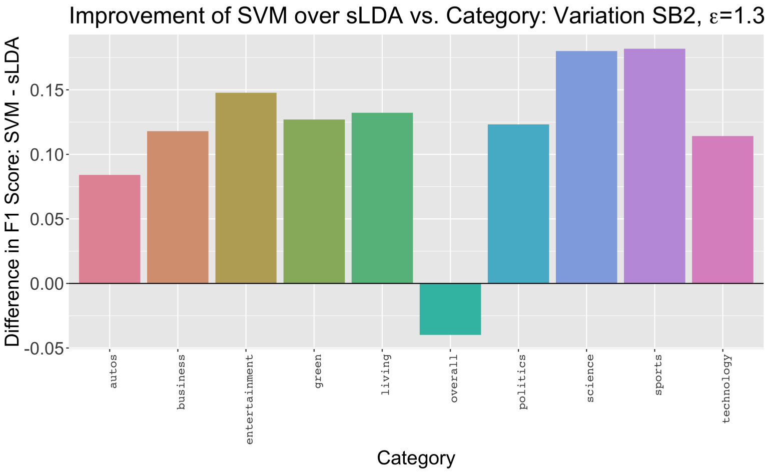
Experiment 4: Which Model Works Best Under the Semantic Bootstrap?

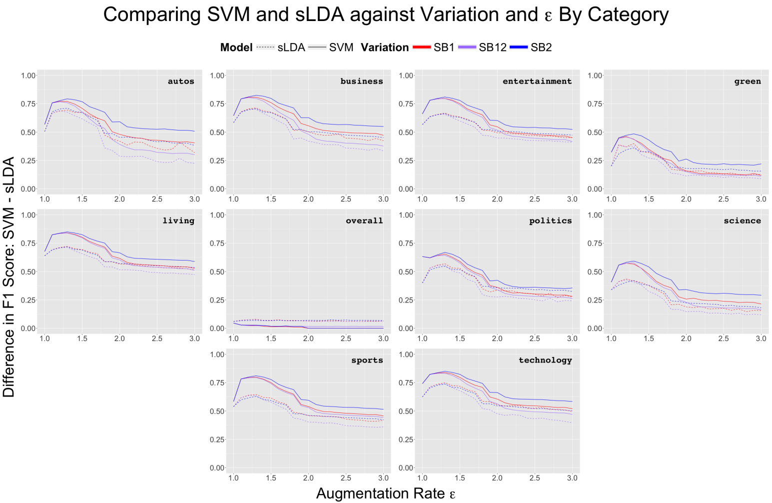
Since two models were chosen for experimentation, it makes sense to compare their performance. For this experiment, we compare the models using a common metric, F1 score.

Hypothesis
LDA has seen a lot of success in clustering words and texts into latent concepts called topics. SVM is a classic, but uses only symbolic representations, and can be seen as more rigid. The hypothesis is that sLDA will perform better than SVM with the Semantic Bootstrap and will see a bigger boost in performance.

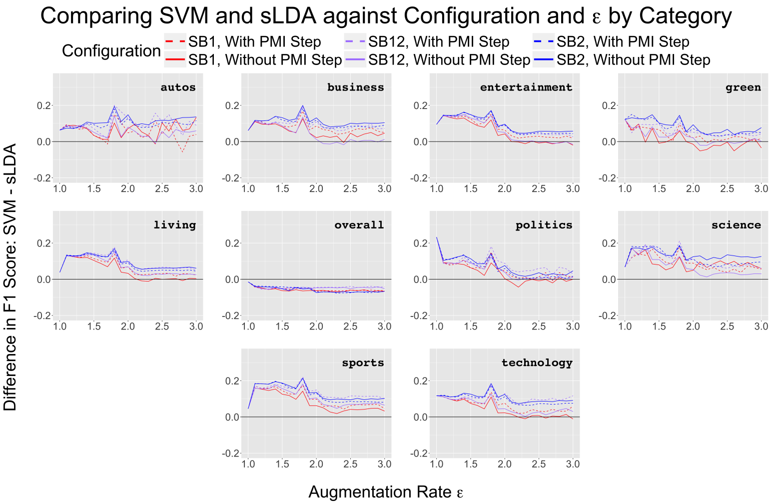

Effect of Model Type on Classifier Performance vs. Variation
F1 Score
0.5
0.6
0.7
0.8
1.0
1.5
2.0
2.5
3.0
Augmentation Rate ε
Model
sLDA
SVM
Variation
SB1
SB12
SB2


Improvement of SVM over sLDA vs. Experimental Configuration
Difference in F1 Score: SVM - sLDA
0.00
0.05
0.10
0.15
0.20
SB1
SB2
SB12
Experimental Configuration
Augmentation Strategy
Without PMI Step
With PMI Step


Improvement of SVM over sLDA vs. Category: Variation SB2, ε=1.3
Difference in F1 Score: SVM - sLDA
0.15
0.10
0.05
0.00
-0.05
autos
business
entertainment
green
living
overall
politics
science
sports
technology
Category


Comparing SVM and sLDA against Variation and ε By Category
Model sLDA SVM Variation SB1 SB12 SB2
autos
business
entertainment
green
living
overall
politics
science
sports
technology
Difference in F1 Score: SVM - sLDA
Augmentation Rate ε
0.00
0.25
0.50
0.75
1.00
1.0
1.5
2.0
2.5
3.0

Effect of Model Choice



Experiment 4 Conclusion

- Just like with `SB0`, SVM (`libshorttext`) performed superior to sLDA, except in the `overall` category.
- At $\varepsilon > 2$, the difference between best performing sLDA models and worst performing SVM models becomes negligible.
- If we must use matched training/testing sets (`SB12`), use SVM with the final term acceptance step.

Likely Reason: The SVM C parameter has been cited in [2] as being important when data is noisy, and SVM pays more attention to data points closer to the decision boundary, whereas sLDA focus on all data points as a generative algorithm. Adding additional terms may add noise that weakens term-document co-occurrences.

1. Introduction
2. Existing Methods
3. Semantic Bootstrap Proposal
4. Experiments
5. Data
6. Models
7. Results
8. Conclusion

Potential I

These results show a lot of potential for the Semantic Bootstrap:

1. allows augmenting of small texts to improve classification, a big problem in industry;
2. does not require any data external to the corpus of interest, reducing querying times and computation power;
3. does not require any niche or clever processing (e.g. NER) of the text, saving computational power;

Potential II

4. eliminates the need to combine small texts into one large document for standard processing;
5. the documented improvements apply to *unigrams* and it is hypothesized to be even better with n-grams;
6. is fast and embarrassingly parallel, including in the construction of S;
7. due to easy implementation and general purpose nature, there are tons of avenues for future research.

Future Work

This research was designed to take a big idea from statistics, the Bootstrap, and try to apply it to a text. This work was very broad and there are several things worth pursuing:

1. using (s)LDA clusters as features in SVM, or some other machine learning model may provide better results; this is common in the literature;
2. evaluating performance on other standard machine learning classifiers such as random forests, Naive Bayes etc.;
3. using individually tuned asymmetric priors α, β for sLDA;
4. determining if ε is specific to the data, or specific to the distribution of document lengths, or a function of some other variable;

Future Work

5. for evaluation purposes only, we used 100 iterations of the Semantic Bootstrap on each corpus, and it is implied that 1 may be enough on average. How many iterations are best?
6. investigating if combining the results of the iterations using boosting or bagging improve performance;
7. investigating other matrix factorizations such as NMF, as well as non-matrix factorization methods to construct S.

Conclusion

In this work, we attempted to adapt the classical bootstrap from statistics for use with text, short text in particular. We call this method the Semantic Bootstrap. We

1. introduced the concept of a semantic space S to serve as the population from which we sample new terms;
2. made extensive use of term discrimination (TF-IDF) features and semantic similarity to choose additional terms to add to the short text;
3. investigated how many terms must be added for best performance and found that adding as few as one term significantly improves classification performance;

Conclusion

4. investigated whether or not using matched training and testing sets performed best and found that only under higher levels of augmentation, it is actually better to apply the method on only the testing/unseen data;
5. investigated if the Semantic Bootstrap yields semantically cohesive augmented texts and found no improvement when performing a final check that new terms improved cohesiveness;
6. used a variation of SVM for short texts and compared it to a contemporary and supervised topic model and found that the tried and true SVM performed better.

References I

[1] D.M. Blei, A.Y. Ng, and M.I. Jordan. "Latent dirichlet allocation". In: *the Journal of machine Learning research* 3 (2003), pp. 993–1022.

[2] Vladimir Cherkassky and Yunqian Ma. "Selection of meta-parameters for support vector regression". In: *International Conference on Artificial Neural Networks*. Springer. 2002, pp. 687–693.

[3] Michael R Chernick. *Bootstrap methods: A guide for practitioners and researchers*. Vol. 619. John Wiley & Sons, 2011.

References II

[4] Bradley Efron and Robert Tibshirani. *An introduction to the bootstrap*. Vol. 57. CRC press, 1993.

[5] Carlos R González and Yaser S Abu-Mostafa. "Mismatched training and test distributions can outperform matched ones". In: *Neural computation* (2015).

[6] Jian Hu et al. "Enhancing text clustering by leveraging Wikipedia semantics". In: *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2008, pp. 179–186.

References III

[7] Xia Hu et al. "Exploiting internal and external semantics for the clustering of short texts using world knowledge". In: *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM. 2009, pp. 919–928.

[8] Rila Mandala, Takenobu Tokunaga, and Hozumi Tanaka. "Combining multiple evidence from different types of thesaurus for query expansion". In: *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 1999, pp. 191–197.

References IV

[9] Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi. "Learning to classify short and sparse text & web with hidden topics from large-scale data collections". In: *Proceedings of the 17th international conference on World Wide Web*. ACM. 2008, pp. 91–100.

[10] Yonggang Qiu and Hans-Peter Frei. "Concept based query expansion". In: *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 1993, pp. 160–169.

[11] F Scholz. "The bootstrap small sample properties". In: *Tech. Rep.* (2007).

References V

[12] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting." In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[13] Jinxi Xu and W Bruce Croft. "Query expansion using local and global document analysis". In: *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 1996, pp. 4–11.

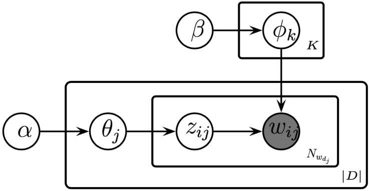
[14] Xiaohui Yan et al. "A biterm topic model for short texts". In: *Proceedings of the 22nd international conference on World Wide Web*. ACM. 2013, pp. 1445–1456.

Thank You

Backup Slides

Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) has seen much acclaim in text mining and natural language processing.



A generative process that assumes text is generated as follows

1. Draw a topic distribution $\theta_j \sim \text{Dir}(\alpha)$ for each document d_j.
2. Draw a word distribution $\phi_k \sim \text{Dir}(\beta)$ for each topic k.
3. For each word position w_{ij} in the document,
   1. Sample a topic indicator $z_{ij} \sim \text{Mult}(\theta_j)$.
   2. Sample a word $w_{ij} \sim \text{Mult}(\phi_{z_{ij}})$.

Latent Dirichlet Allocation

There are several ways to estimate LDA topic models and its many variations

1. (Collapsed) Gibbs Sampling
2. (Collapsed) Variational Inference
3. Expectation Propagation
4. Spectral Decomposition

With the goal begin sampling a correct topic indicator upon convergence

$$P(z_{ij}|z_{-(i,j)}, \mathbf{w}, \alpha, \beta) \propto (n_{w_{i\cdot}} + \alpha_k) \frac{n_{\cdot v} + \beta_v}{\sum_{r=1}^{V} n_{\cdot r} + \beta_r}$$

Latent Dirichlet Allocation

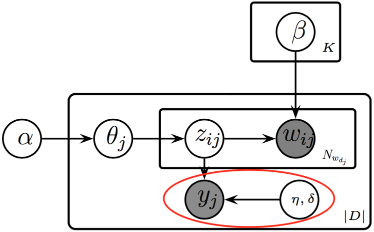
The LDA framework induces some limitations.

1. features must be word counts to retain a multinomial likelihood and maintain conjugacy with the Dirichlet priors.
2. LDA is unsupervised
3. as such, it is unlikely the topics discovered with LDA match the
4. topic labeling is inconsistent from run to run
5. topic labeling must be performed manually by eye (or with post-hoc analysis)

While researchers have successfully used TF-IDF with LDA, it was not attempted to remove another layer of complexity. Instead we focus on remedying all of the other limitations...

Supervised Latent Dirichlet Allocation (sLDA)

Graphical model:



The generative process is identical to that of LDA except it adds one final step:

3. Sample response variable $y_k | z_{ij}, \eta, \delta \sim \text{GLM}(\bar{z}, \eta, \delta)$, where $\bar{z} = \frac{1}{N_{w_{d_j}}} \sum_{i=1}^{N_{w_{d_j}}} z_{ij}$, η is a vector of regression coefficients and δ is a scale parameter, like σ^2 and GLM is a link function.