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

Sign Up Today!
24HOURFITNESS.COM
A 24 Hour Fitness membership gets you tons of studio classes for the price of none.

Open Your Online Store
www.volusion.com
Get everything you need to succeed with Volusion’s all-in-one ecommerce platform. Start yo...
Example: Tweet

Twitter still rules the short text world.

Openwashing, where a company secretly plans to restrict access to "open source", is one of most toxic ideas in CS. Don't do it!

But there a whole series of issues regarding their use in machine learning.
Example: Title of a Blog Post

Snap stumbles toward a volatile IPO
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.
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) $q$;
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 $q'$ (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.

<table>
<thead>
<tr>
<th>Topic Type</th>
<th>Base</th>
<th>Expanded with</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>WordNet only</td>
</tr>
<tr>
<td>Title</td>
<td>0.1175</td>
<td>0.1299 (+10.6%)</td>
</tr>
<tr>
<td>Description</td>
<td>0.1428</td>
<td>0.1525 (+6.8%)</td>
</tr>
<tr>
<td>All</td>
<td>0.1976</td>
<td>0.2018 (+2.1%)</td>
</tr>
</tbody>
</table>

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]:

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” → 
   {"visit apple", "apple store", "visit store"})

2. researchers believed that such a method models cooccurrence patterns and not just cooccurences.

3. 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.

**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).**

<table>
<thead>
<tr>
<th>$T$</th>
<th>5</th>
<th>10</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>$-55.0 \pm 0.4$</td>
<td>$-236.4 \pm 2.0$</td>
<td>$-1015.7 \pm 5.9$</td>
</tr>
<tr>
<td>LDA-U</td>
<td>$-54.2 \pm 0.8$</td>
<td>$-234.8 \pm 1.1$</td>
<td>$-1009.4 \pm 4.4$</td>
</tr>
<tr>
<td>Mix</td>
<td>$-53.8 \pm 0.1$</td>
<td>$-233.0 \pm 1.4$</td>
<td>$-1007.6 \pm 6.7$</td>
</tr>
<tr>
<td>BTM</td>
<td>$-52.4 \pm 0.1$</td>
<td>$-227.8 \pm 0.3$</td>
<td>$-990.2 \pm 3.8$</td>
</tr>
</tbody>
</table>

**Table 8**

<table>
<thead>
<tr>
<th>$K$</th>
<th>50</th>
<th>100</th>
<th>150</th>
<th>200</th>
<th>250</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>38.07</td>
<td>74.38</td>
<td>108.13</td>
<td>143.47</td>
<td>178.66</td>
</tr>
<tr>
<td>BTM</td>
<td>128.64</td>
<td>250.07</td>
<td>362.27</td>
<td>476.19</td>
<td>591.24</td>
</tr>
</tbody>
</table>

**Table 9**

<table>
<thead>
<tr>
<th>$K$</th>
<th>50</th>
<th>100</th>
<th>150</th>
<th>200</th>
<th>250</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>3177</td>
<td>5524</td>
<td>7890</td>
<td>10218</td>
<td>12561</td>
</tr>
<tr>
<td>BTM</td>
<td>927</td>
<td>946</td>
<td>964</td>
<td>984</td>
<td>1002</td>
</tr>
</tbody>
</table>
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.

4many 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...
Semantic Bootstrap Proposal
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
   - with replacement;
   - independent;
   - 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].

\[ n = 30 \]

\[ \Omega \]

\[ s_1 \]

\[ s_2 \]

\[ s_N \]
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.

![Diagram](diagram.png)
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.

![Diagram of the Semantic Bootstrap]

\[D \rightarrow \text{Sample Terms from a Parallel Document} \rightarrow S \rightarrow \text{Augmented Bag-of-Words} \rightarrow D^*\]

- \(d_1\): the quick brown fox
- \(d_2\): jumps over the lazy
- \(d_3\): dog

- \(d_1^*\): the quick fast brown gray fox fox
- \(d_2^*\): jumps over leaps the lazy lethargic hops lazy
- \(d_3^*\): dog canine

**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 $X$ with $|D|$ rows, one for each document $d_i$, and $|V|$ columns, one for each term $t_j$.

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

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 $X$ as follows:

$$X \approx D_k \Sigma_k W_k^T$$

We can then construct term and document similarities as follows

$$S_t = XX^T$$

$$= \left( D_k \Sigma_k W_k^T \right) \left( D_k \Sigma_k W_k^T \right)^T$$

$$= D_k \Sigma_k W_k^T W_k \Sigma_k^T D_k^T$$

$$= D_k \Sigma_k^2 D_k^T$$

and similarly,

$$S_d = X^T X = W_k \Sigma_k^2 W_k^T$$
The Semantic Space $S$: Cosine Similarity

Once we have computed $S_t$ and $S_d$, 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,

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

For documents,

$$\delta_{d_i, d_j} = \text{sim}(d_i, d_j) = \frac{d_i \cdot d_j}{||d_i||||d_j||}$$
A Sampling Scheme

Up to this point, we have decomposed $X$ 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 $\delta_t$,
2. we have a probability distribution over document similarities $\delta_d$,
3. we can also have a probability distribution over word discrimination $X_i$.,
A Conditional Probability Distribution over $S$

We can now propose a sampling from $S$:

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

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

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

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

$$P(d \rightarrow d^*) \propto \min \left\{ 1, \frac{1}{Z} \exp (I(d^*) - I(d)) \right\}$$
The Semantic Bootstrap Algorithm

The Semantic Bootstrap algorithm can be illustrated as the following flowchart.

\[
\text{do foreach } d \text{ in } D \\
\text{Select a term } t \text{ in } d \text{ as a target. We want to sample a word similar to it. } t = \{ \text{fox} \} \\
\text{Select a document } d' \text{ semantically similar to } d. d' = \{ \text{a fast wolf} \} \\
\text{Select a term } t' \text{ from } d' \text{ that is semantically similar to } t. t' = \{ \text{wolf} \} \\
\text{Accept } t' \text{ into the augmented bag-of-words } d^* \text{ with probability proportional to change in semantic cohesion.} \\
\text{the quick brown fox wolf} \\
\text{A new corpus } D^*_1 \text{ consisting of } \lfloor \varepsilon \rfloor \text{ documents as augmented bag-of-words.} \\
\]
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.

\[\begin{array}{c|ccccc}
   & yom & kippur & tel & aviv & style \\
   d \quad & \text{TF-IDF} & 9.6 & 9.6 & 8.1 & 9.5 & 7.2 \\
   & P(t \mid t_1, t_2, t_3, t_4, t_5) & 0.22 & 0.22 & 0.184 & 0.216 & 0.164 \\
\end{array}\]

\[\begin{array}{c|ccccc}
   & celebrating & rosh & hashanah & haifa & israel \\
   d' \quad & \text{TF-IDF} & 5.3 & 9.1 & 9.4 & 8.9 & 7.8 \\
   & P(t \mid t_1, t_2, t_3, t_4, t_5) & 0.13 & 0.225 & 0.232 & 0.22 & 0.19 \\
\end{array}\]
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 $S$.

$$
\begin{array}{cccccc}
\text{TF-IDF} & 9.6 & 9.6 & 8.1 & 9.5 & 7.2 \\
\text{P}(t \mid t_1, t_2, t_3, t_4, t_5) & 0.22 & 0.22 & 0.184 & 0.216 & 0.164 \\
\end{array}
$$

$$d \quad \text{yom kippur} \quad \text{tel aviv} \quad \text{style}$$

$$d' \quad \text{celebrating} \quad \text{rosh hashanah} \quad \text{haifa} \quad \text{israel}$$

$$\begin{array}{cccccc}
\text{cos similarity} & +0.09 & +0.72 & +0.76 & +0.21 & +0.62 \\
\text{P}(t' \mid t) & 0.04 & 0.30 & 0.32 & 0.09 & 0.26 \\
\end{array}$$
Now we have a candidate document $d^*$. We accept the new term $t'$ and thus $d^*$ according to the change in semantic cohesiveness, $P(d \rightarrow d^*)$:

- If $P(d \rightarrow d^*)$ 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

Ryan Robert Rosario

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

UCLA Department of Statistics
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 results? In other words, what are appropriate values for augmentation rate $\varepsilon$?

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 \textit{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:

<table>
<thead>
<tr>
<th>Variation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB1</td>
<td>Augment only the training data, use original testing data.</td>
</tr>
<tr>
<td>SB2</td>
<td>Use original training data and classifier, augment only testing data.</td>
</tr>
<tr>
<td>SB12</td>
<td>Matched case; augment both sets.</td>
</tr>
<tr>
<td>SB0</td>
<td>Raw data; neither is augmented.</td>
</tr>
</tbody>
</table>
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.
Introduction

Existing Methods

Semantic Bootstrap Proposal

Experiments

Data

Models

Results

Conclusion

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

UCLA Department of Statistics
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 of ten categories.

<table>
<thead>
<tr>
<th>Category</th>
<th>% of Titles</th>
<th>Category</th>
<th>% of Titles</th>
</tr>
</thead>
<tbody>
<tr>
<td>autos</td>
<td>2.1</td>
<td>politics</td>
<td>7.5</td>
</tr>
<tr>
<td>business</td>
<td>7.4</td>
<td>science</td>
<td>3.0</td>
</tr>
<tr>
<td>entertainment</td>
<td>20.8</td>
<td>sports</td>
<td>14.3</td>
</tr>
<tr>
<td>green</td>
<td>2.1</td>
<td>technology</td>
<td>18.8</td>
</tr>
<tr>
<td>living</td>
<td>21.6</td>
<td>overall</td>
<td>2.4</td>
</tr>
</tbody>
</table>

7Blogs 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.
Introduction

Existing Methods

Semantic Bootstrap Proposal

Experiments

Data

Models

Results

Conclusion

References
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 $\beta_i$ 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.*

<table>
<thead>
<tr>
<th>Model / Metric</th>
<th>Accuracy Macro</th>
<th>Accuracy Micro</th>
<th>Precision Macro</th>
<th>Recall Macro</th>
<th>F1 Score Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear SVM L2, TF-IDF</td>
<td>0.811</td>
<td>0.596</td>
<td>0.720</td>
<td>0.624</td>
<td>0.668</td>
</tr>
<tr>
<td>Linear SVM L2, Binary</td>
<td>0.889</td>
<td>0.645</td>
<td>0.697</td>
<td>0.628</td>
<td>0.644</td>
</tr>
<tr>
<td>Linear SVM L2, Word Count</td>
<td>0.832</td>
<td>0.627</td>
<td>0.725</td>
<td>0.638</td>
<td>0.679</td>
</tr>
<tr>
<td>Linear SVM L2, Term Frequency</td>
<td>0.899</td>
<td>0.587</td>
<td>0.701</td>
<td>0.628</td>
<td>0.663</td>
</tr>
<tr>
<td>Linear SVM L1, TF-IDF</td>
<td>0.894</td>
<td>0.602</td>
<td>0.703</td>
<td>0.619</td>
<td>0.638</td>
</tr>
<tr>
<td>Linear SVM L1, Binary</td>
<td>0.933</td>
<td>0.614</td>
<td>0.691</td>
<td>0.625</td>
<td>0.640</td>
</tr>
<tr>
<td>Linear SVM L1, Word Count</td>
<td>0.885</td>
<td>0.622</td>
<td>0.691</td>
<td>0.625</td>
<td>0.639</td>
</tr>
<tr>
<td>Linear SVM L1, Term Frequency</td>
<td>0.900</td>
<td>0.601</td>
<td>0.691</td>
<td>0.626</td>
<td>0.641</td>
</tr>
<tr>
<td>Logistic Regression, TF-IDF</td>
<td>0.931</td>
<td>0.623</td>
<td>0.700</td>
<td>0.624</td>
<td>0.642</td>
</tr>
<tr>
<td>Logistic Regression, Binary</td>
<td>0.840</td>
<td>0.675</td>
<td>0.700</td>
<td>0.619</td>
<td>0.639</td>
</tr>
<tr>
<td>Logistic Regression, Word Count</td>
<td>0.898</td>
<td>0.622</td>
<td>0.700</td>
<td>0.620</td>
<td>0.639</td>
</tr>
<tr>
<td>Logistic Regression, Term Frequency</td>
<td>0.849</td>
<td>0.567</td>
<td>0.700</td>
<td>0.620</td>
<td>0.639</td>
</tr>
</tbody>
</table>

\(^8\)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 $\varepsilon$
   - Effect of Time of Sampling
   - Effect of the Probabilistic Term Acceptance Step
   - Effect of Model Choice

8 Conclusion

Ryan Robert Rosario

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

UCLA Department of Statistics
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 $\varepsilon$?

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 $\varepsilon$

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 $\varepsilon$ on sLDA Classifier Performance

**Category**
- autos
- entertainment
- living
- politics
- sports
- business
- green
- overall
- science
- 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**

Augmentation Rate $\varepsilon$
Effect of $\varepsilon$ on SVM Classifier Performance

Category: autos, entertainment, living, politics, sports, business, green, overall, science, technology

F1 Score vs. Augmentation Rate $\varepsilon$

SB1 With PMI Step, SB2 With PMI Step, SB12 With PMI Step, SB1 Without PMI Step, SB2 Without PMI Step, SB12 Without PMI Step

Ryan Robert Rosario
The Semantic Bootstrap: Application of the Bootstrap for Small Text Classification
UCLA Department of Statistics
Effect of $\epsilon$ on sLDA Classifier Performance vs. Configuration, Averaged Over all Categories

Parameter: Augmentation Rate $\epsilon$

- SB1, With PMI Step
- SB1, Without PMI Step
- SB12, With PMI Step
- SB12, Without PMI Step
- SB2, With PMI Step
- SB2, Without PMI Step

Note: The graph shows the effect of varying $\epsilon$ on the AUC (Area Under the Curve) for different configurations of the sLDA classifier.
Effect of $\epsilon$ on SVM Classifier Performance vs. Configuration, Averaged Over all Categories

- SB1, With PMI Step
- SB1, Without PMI Step
- SB12, With PMI Step
- SB12, Without PMI Step
- SB2, With PMI Step
- SB2, Without PMI Step

Parameter: Augmentation Rate $\epsilon$

F1 Score

[Graph showing trends for different configurations across varying $\epsilon$ values]
The hypothesis of $1.5 \leq \varepsilon^* < 2$ was disproven:

1. The optimal value of $\varepsilon$ 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, $\varepsilon = 1.3$ 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

Classifier Performance via ROC Curve for $\varepsilon$=1.3

- **autos**: AUC: 0.87 $\sim$ 0.9
- **business**: AUC: 0.89 $\sim$ 0.92
- **entertainment**: AUC: 0.815 $\sim$ 0.859
- **green**: AUC: 0.75 $\sim$ 0.81
- **living**: AUC: 0.84 $\sim$ 0.88
- **overall**: AUC: 0.61 $\sim$ 0.61
- **politics**: AUC: 0.82 $\sim$ 0.87
- **science**: AUC: 0.76 $\sim$ 0.8
- **sports**: AUC: 0.81 $\sim$ 0.85
- **technology**: AUC: 0.92 $\sim$ 0.93

Ryan Robert Rosario

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

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

SVM Classifier Performance for $\varepsilon=1.3$

Experimental Variation
Effect of Time of Sampling

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.
sLDA: Effect of $\epsilon$ and Variation on Classifier Performance Averaged Over all Categories

Variation
- SB1
- SB12
- SB2

AUC

Augmentation Rate $\epsilon$

Effect of Time of Sampling

Ryan Robert Rosario

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

UCLA Department of Statistics
SVM: Effect of $\varepsilon$ and Variation on Classifier Performance Averaged Over all Categories

Variation: SB1, SB12, SB2

Effect of Time of Sampling

Ryan Robert Rosario

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

UCLA Department of Statistics
Effect of Variation and Augmentation Rate on sLDA Classifier Performance

Variation

**Autos**

**Business**

**Entertainment**

**Green**

**Living**

**Overall**

**Politics**

**Science**

**Sports**

**Technology**

AUC vs. Augmentation Rate $\varepsilon$
Effect of Variation and Augmentation Rate on SVM Classifier Performance

F1 Score vs. Augmentation Rate $\varepsilon$

Variation: SB1, SB12, SB2

Categories: autos, business, entertainment, green, living, overall, politics, science, sports, technology
sLDA Classifier Performance across Experimental Configurations at \( \varepsilon = 1.3 \)

Effect of Time of Sampling

Experimental Variation and Configuration
SVM Classifier Performance across Experimental Configurations at $\varepsilon=1.3$

Experimental Variation and Configuration
Results among the SBx variations were mixed:

1. At small $\varepsilon$ and $\varepsilon^*$, differences were negligible.
2. For sLDA, SB2 consistently performed better as $\varepsilon$ increased.
3. For SVM, SB2 performed best at large $\varepsilon \geq 1.8$.
4. SB2 performed the best at higher values for $\varepsilon$.
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.
sLDA Classifier Performance for Experimental Variation SB2 at $\varepsilon = 1.3$

Augmentation Strategy
- Original Data, No Augmentation
- Without PMI Step
- With PMI Step

AUC

Category
- autos
- business
- entertainment
- green
- living
- overall
- politics
- science
- sports
- technology

Ryan Robert Rosario
The Semantic Bootstrap: Application of the Bootstrap for Small Text Classification
UCLA Department of Statistics
Effect of Time of Sampling

SVM Classifier Performance for Experimental Variation SB2 at $\varepsilon=1.3$

**Augmentation Strategy**
- Original Data, No Augmentation
- Without PMI Step
- With PMI Step

<table>
<thead>
<tr>
<th>Category</th>
<th>F1 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto</td>
<td>0.75</td>
</tr>
<tr>
<td>business</td>
<td>0.80</td>
</tr>
<tr>
<td>entertainment</td>
<td>0.65</td>
</tr>
<tr>
<td>green</td>
<td>0.40</td>
</tr>
<tr>
<td>living</td>
<td>0.85</td>
</tr>
<tr>
<td>overall</td>
<td>0.70</td>
</tr>
<tr>
<td>politics</td>
<td>0.75</td>
</tr>
<tr>
<td>science</td>
<td>0.60</td>
</tr>
<tr>
<td>sports</td>
<td>0.70</td>
</tr>
<tr>
<td>technology</td>
<td>0.80</td>
</tr>
</tbody>
</table>

Ryan Robert Rosario
The Semantic Bootstrap: Application of the Bootstrap for Small Text Classification
UCLA Department of Statistics
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

sLDA Classifier Performance vs. Augmentation Strategy at $\varepsilon = 1.3$

- Original Data, No Augmentation
- Without PMI Step
- With PMI Step

AUC
Effect of the Probabilistic Term Acceptance Step

SVM Classifier Performance vs. Augmentation Strategy at $\epsilon=1.3$

- Original Data, No Augmentation
- Without PMI Step
- With PMI Step
Effect of the Probabilistic Term Acceptance Step

sLDA: Augmentation Strategy vs. Classifier Performance Over all Variations and Categories

Augmentation Strategy

- - With PMI Step
--- Without PMI Step

AUC

1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 3.0
Augmentation Rate $\epsilon$
Effect of the Probabilistic Term Acceptance Step

SVM: Augmentation Strategy vs. Classifier Performance Over all Variations and Categories

Augmentation Strategy

- - With PMI Step
- - Without PMI Step

F1 Score vs. Augmentation Rate $\epsilon$

Ryan Robert Rosario

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

UCLA Department of Statistics
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 $\varepsilon$.

**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

- **Model**
  - sLDA
  - SVM

- **Variation**
  - SB1
  - SB12
  - SB2

- **F1 Score**
- **Augmentation Rate $\epsilon$**

Ryan Robert Rosario
The Semantic Bootstrap: Application of the Bootstrap for Small Text Classification
UCLA Department of Statistics
Effect of Model Choice

**Improvement of SVM over sLDA vs. Experimental Configuration**

- **SB1**
- **SB2**
- **SB12**

*Augmentation Strategy*
- **Without PMI Step**
- **With PMI Step**
Improvement of SVM over sLDA vs. Category: Variation SB2, $\varepsilon=1.3$
Comparing SVM and sLDA against Variation and $\varepsilon$ By Category

Model: sLDA — SVM
Variation: SB1 — SB12 — SB2

Augmentation Rate $\varepsilon$

Differences in F1 Score: SVM - sLDA
Comparing SVM and sLDA against Configuration and $\varepsilon$ by Category

- **SB1, With PMI Step**
- **SB12, With PMI Step**
- **SB2, With PMI Step**
- **SB1, Without PMI Step**
- **SB12, Without PMI Step**
- **SB2, Without PMI Step**

**Effect of Model Choice**

Ryan Robert Rosario

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

UCLA Department of Statistics
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.
The Semantic Bootstrap: Application of the Bootstrap for Small Text Classification

Ryan Robert Rosario

UCLA Department of Statistics
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 $\alpha, \beta$ for sLDA;
4. determining if $\varepsilon$ 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


References II


References III


References IV


References V


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 $w_{ij} \sim \text{Mult}(\phi_{z_{ij}})$. 

Ryan Robert Rosario

The Semantic Bootstrap: Application of the Bootstrap for Small Text Classification  
UCLA Department of Statistics
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:

\[ P(z_{ij} | z_{-(i,j)}, w, \alpha, \beta) \propto (n_{wi.} + \alpha_k) \frac{n.v + \beta_v}{\sum_{r=1}^{V} n.r + \beta_r} \]
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

\[
\bar{z} = \frac{1}{N_{wd_j}} \sum_{i=1}^{N_{wd_j}} z_{ij}, \quad \eta \text{ is a vector of regression coefficients and}
\]

\[
\delta \text{ is a scale parameter, like } \sigma^2 \text{ and GLM is a link function.}
\]