Exploiting Semantic Word Relationships for Improved Unsupervised Academic Document Classification

by

David W. Yan

Department of Computer Science
Duke University

Date: ______________________
Approved:

__________________________
Xiaobai Sun, Supervisor

__________________________
Nikos Pitsianis

__________________________
John H. Reif

Dissertation submitted in partial fulfillment of the requirements for Graduation with Distinction in the Department of Computer Science at Duke University
2018
Abstract

Exploiting Semantic Word Relationships for Improved Unsupervised Academic Document Classification

by

David W. Yan

Department of Computer Science
Duke University

Date: ____________________

Approved:

__________________________
Xiaobai Sun, Supervisor

__________________________
Nikos Pitsianis

__________________________
John H. Reif

An abstract of a dissertation submitted in partial fulfillment of the requirements for Graduation with Distinction in the Department of Computer Science at Duke University 2018
Abstract

In our thesis work, we investigate on unsupervised theme-based categorization or classification of documents in a large corpus, where the documents are represented in a high-dimensional feature space. Unsupervised classification of text documents is in high demand with the data explosion of the modern age, yet it remains a challenging problem.

In particular, we re-examine two key areas: the manner in which the documents are represented and the process by which the documents are clustered. Two innovative methods are presented in the joint research work with Rob Martorano.

The first is on feature transformation, which is elaborated on in this thesis. We point out that existing approaches for document feature description serve well for author identification but pose limitations on theme-based classification. In our feature transformation, we discount esoteric use of words by authors and disclose and exploit semantic similarities and associations among different words used by different authors. We first locate semantically close words by utilizing word embedding techniques and products based on much larger word collections, external to the terms used in a particular document corpus. We then make numerical associations among term neighbors with similar semantic meanings; we denote these term neighborhoods as semantic elements. Using semantic elements, we use a self-tuning Gaussian blurring technique to increase association between documents that share similar context patterns.
The second contribution is on cluster revision, which is briefly discussed in this thesis and elaborated more in Rob Martorano’s thesis. Clustering algorithms are typically used after feature dimension reduction. Some properties are preserved, and some are lost in the reduced dimension space. Some clusters are fragmented into smaller ones, and some are merged. We revise the clustering results by going back to the high dimensional space. We characterize the cluster features with what we refer to as \textit{stochastic barcodes}.

We developed a software architecture composed of the following major components. The first component uses semantic elements to form a refined document feature space using our novel feature transformation method. The second component performs a dimension reduction on the document feature space, then forms and refines the subsequent document clusters.

We show, with experimental results on real-word document corpora, improvements made by our approach in comparison to existing and influential ones.
Dedicated to my family who has been endlessly supportive of my efforts: Jeremy Yan, Weili Qu, and Jessica Yan.

Dedicated to my friends who have provided me unwavering support: Kristianna Elbert, Bobby Wang, and Ben Schwennesen.

Dedicated to the professors and graduate students who were so generous with their time in helping me: Xiaobai Sun, Nikos Pitsianis, Alexandros Iliopoulos, and Tiancheng Liu. I could not imagine my Duke experience without the four of you.
## Contents

Abstract iv

List of Tables ix

List of Figures x

Acknowledgements xii

1 Overview 1

2 Feature Transformation 4

2.1 Feature Description .............................................. 4

2.2 Term Selection .................................................. 5

2.3 Semantic Term Associations ..................................... 6

2.3.1 Word Embeddings ............................................. 7

2.3.2 Semantic Term Association via k-NN ......................... 12

2.4 Term Weighting via Gaussian Blurring ......................... 13

3 Architecture 16

3.1 Phase 1 - From Corpus to DSE Matrix .......................... 17

3.1.1 Corpora Collection ........................................... 17

3.1.2 Forming DT Matrix ........................................... 18

3.1.3 Forming DSE Matrix .......................................... 19

3.2 Phase 2 - From DSE to Theme Clustering ...................... 19

3.2.1 Feature Dimension Reduction ............................... 20
# List of Tables

2.1 Doc × Term (DT) Frequency Representation ........................................ 4

2.2 Occurrence-Based DT Matrix ............................................................... 14

2.3 Doc × Semantic Element (DSE) ............................................................ 14

4.1 DSE Affinity Post-Evaluation Measures for Corpus 1. These measures were calculated based on the 5 post s-barcoding document clusters. The final purity score for this result was 0.9791. ........................................ 30

4.2 DT Affinity Post-Evaluation Measures for Corpus 1. These measures were calculated based on the 4 post s-barcoding document clusters. The final purity score for this result was 0.8945. ........................................ 30
# List of Figures

2.1 Term Neighborhoods Generated via k-NN ........................................ 6  
2.2 Word2Vec Training Window - Referenced from Chris McCormick’s Word2Vec Tutorial ......................................................... 8  
2.3 One Hot Lookup - Referenced from Chris McCormick’s Word2Vec Tutorial ................................................................. 10  
2.4 Full Model Representation of CBOW Training - Rong [10] ........ 11  
2.5 Full Model Representation of Skip-Gram Training - Rong [10] ... 12  
2.6 Theoretical Term Weight Distribution via Gaussian Blur .......... 14  
3.1 Architecture - Phase 1: From Corpus to DSE Matrix ............... 17  
3.2 Architecture - Phase 2: From DSE to Theme Clustering ............ 20  
4.1 3D view of spectral embeddings of documents with DSE representa-
tion (top row) and with DT representation (bottom row). The full 
embedding is done in a 5-dimensional space. ............................. 28
4.2 Document clustering results between DSE and DT for Corpus 1. The clustering results presented above are post-stochastic barcode revision clusters. The figure on the left shows the composition of DSE affinity document clusters. With the DSE affinity, we began with 15 initial clusters and ended up with 5 clusters after s-barcode revision, which directly matches the number of true labels in the corpus. A minimum cluster size of 5 was set for stochastic barcode revision. For DSE, a $k = 4$ was used for the k-nearest neighbors search. The DSE affinity clearly shows that each cluster is primarily dominated by a single class of document. The figure on the right shows the DT document clusters. Using the DT affinity, we began with 16 initial clusters and ended up with 4 clusters after s-barcode revision. Meanwhile, we see from the DT results that the music class is mistakenly clustered with the computer vision papers and thus missed altogether. Both the DSE and DT results used a 5 dimensional embedding space, and both were clustered using a mean shift bandwidth of 0.001.

4.3 S-Barcodes for the initial 15 Clusters formed from clustering DSE in the low dimensional embedding space. The barcodes for these 15 clusters are shown before S-Barcode revision.

4.4 S-Barcodes for the 5 clusters formed after S-Barcode revision. For this revision, we set a minimum cluster size of 10.

4.5 Document clustering results between DSE and DT for Corpus 2. The clustering results presented above are post-stochastic barcode revision clusters. The figure on the left shows the composition of DSE affinity document clusters. With the DSE affinity, we began with 54 initial clusters and ended up with 8 clusters. A minimum cluster size of 10 was set for stochastic barcode revision. For DSE, a $k = 4$ was used for the k-nearest neighbors search. DSE was clustered using a mean shift bandwidth of 0.00425. The DSE affinity clearly shows that most clusters are dominated by a single class of document. The figure on the right shows the DT document clusters. Using the DT affinity, we began with 55 initial clusters and ended up with 5 clusters after s-barcode revision. DT was clustered using a mean shift bandwidth of 0.003. Both the DSE and DT results used an 10 dimensional embedding space.
Acknowledgements

Thank you to the people who have spent countless hours toiling with me and aiding my academic growth:

- Alexandros Iliopoulos
- Tiancheng Liu
- Nikos Pitsianis
- Xiaobai Sun

Thank you to those who have provided me tremendous support behind the scenes:

- Jeremy Yan (Father)
- Weili Qu (Mother)
- Jessica Yan (Little Sister)
This thesis work centers on unsupervised theme classification of text documents in a large corpus, based on content similarity and dissimilarity. We address two fundamental, long-standing problems in typical theme-based document classification processes. We introduce our solutions to these problems, and we also present improved theme-classification results.

A typical unsupervised document classification process has three main components: (1) document representation in feature vectors, (2) feature space dimension reduction, and (3) document clustering in a reduced dimensional space. Document classification processes differ in algorithms for one or more of these components.

The first standing problem regards how to represent a document toward the objective of theme classification. Conventionally, a document in a corpus is represented in one of three ways. First, a document could be represented as a term-frequency vector, where each term is weighted by its occurrences. The second is as a term frequency-inverse document frequency (tf-idf) vector which scales proportionally to the number of times a term appears in a document in relation to the number of times it appears in the corpus. The third is as an occurrence vector that tracks
only whether a particular term has been used in the document or not. These representation schemes serve better for author identification problems rather than theme identification because term frequencies and the use of certain terms can be esoteric to particular authors.

We introduce a novel scheme of feature transformation for improved theme-based document classification. The key idea is to discount esoteric use of words by individual authors, and discover and exploit semantic associations between words.

The second standing problem is about how to revise the clustering results that are obtained in a dimension reduced space. We introduce an iterative method for cluster revision in a high-dimensional feature space. Such integrated use of low-dimension and high-dimension representation for clustering is new.

We developed a software architecture to acquire, process, cluster, visualize, and classify corpora data. We gather documents from a variety of sources including PubMed, Scopus, and ArXiv. After processing each document, we perform a dimension reduction on the document space using certain state-of-the-art algorithms, either geometric or stochastic, in the form of SVD or t-SNE respectively. For clustering, we use the mean shift clustering algorithm. The clustering results are shown and evaluated using a variety of representation tools.

We evaluate our approach using two document data corpora. The first corpus contains five dissimilar classes of documents. The second corpus contains more classes of documents where some of the classes are closer to each other, making the classification more difficult. I present results that demonstrate the effectiveness of our algorithm over traditional document representation models and provide evidence that our feature transformation and cluster revision schemes improve end classification results for both corpora.

In the remainder of this thesis, I will elaborate on feature transformation, our software architecture and functions, data acquisition, experiments, and evaluation of
2.1 Feature Description

Traditional document classification approaches use word occurrences and frequencies on a per-document basis. This is illustrated in the document-term Table 2.1 $DT$, where $DT(i, j)$ represents the number of times a term $j$ is used in document $i$. This representation is problematic for a multitude of reasons. First, the use of certain terms is esoteric to a particular author. For example, one author may prefer to use the descriptor “unsullied” while another may prefer to use the term “pristine” to describe the same thing. Furthermore, term frequencies are also esoteric for a particular author as certain authors may be more verbose than others. Additionally, this method of document representation misses semantic associations.

<table>
<thead>
<tr>
<th></th>
<th>Untouched</th>
<th>Unsullied</th>
<th>Pristine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doc 1</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Doc 2</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Doc 3</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 2.1: Doc $\times$ Term (DT) Frequency Representation
In Table 2.1, we show three documents in which three terms “Untouched”, “Unsullied”, and “Pristine” are used. Despite the fact that the words “Unsullied” in Document 2 and “Pristine” in Document 3 have similar semantic meanings, the two documents are represented by orthogonal feature (row) vectors, \( DT(2,:) \cdot DT(3,:) = 0 \). In other words, the document representation scheme fails to capture such association between documents. We introduce, in the rest of the chapter, our approach to resolving this problem.

2.2 Term Selection

Before any transformations on the document features, we perform an initial term selection, i.e., to filter out certain terms used. The first is to remove background terms, which have no or little differentiation power. There are two types of background terms. Background terms of the first type are stop words such as “a”, “an”, “the”, “is”, “at”, “and”, “that”, “which”, “on”, etc. Such words are used ubiquitously across all documents in English. We used a stopword list provided to us by Professor N. Pitsianis’s group.

Background terms of the second type are corpus specific, where each term is used by all or nearly all documents in the corpus. We found and removed these words using a MF (most-frequent) filtering where we remove the most frequent term features. Since terms are represented as our column features, the column sum for a particular term feature represents the total usage of that particular term. Using MF filtering, we set an upper threshold and remove any term features that surpass it. Often, these are terms that are used by all or nearly all documents. These terms offer little description and, were their embeddings kept, could imply semantic associations between completely unrelated documents.

We also perform a LF (least-frequent) filtering. These least frequent features are filtered out because they provide too little insight into discovering a shared theme
across multiple documents. That is, their frequency implies that these terms do not offer enough association or correlation between documents.

Term selection affects numerical measures of document similarity or dissimilarity. As a consequence, it also reduces the number of unique terms, and thus our feature space, substantially.

2.3 Semantic Term Associations

A novel contribution of this thesis work is to make associations between terms with similar semantics and discount the difference due to esoteric use of words by different authors. In the example illustrated in Table 2.1, documents using semantically similar words are represented as orthogonal feature vectors. Our principle is that different words with similar semantic meanings shall lead to higher correlation. We describe how to materialize this idea.

![Figure 2.1: Term Neighborhoods Generated via k-NN](image)

First, we attempt to find neighbor terms, as exemplified in Figure 2.1. Next, we make numerical associations among a word’s neighbor terms by re-distributing its weight. By the existing work on co-clustering by Dhillon [4], terms are clustered by
their features, the feature of term-\( j \) is column-\( j \) of the same DT matrix, describing how term-\( j \) is used in the corpus. The term feature description is corpus-specific, affected by, and limited to, the use of the words by the authors affiliated with the corpus. Other limitations of this approach lie in the clustering algorithm, which renders non-overlapping term clusters of different sizes. A large term cluster may contain many words with semantically diverse meanings. We lift these limitations with a different approach.

We find neighbors in a word embedding space, external to the corpus. Word embedding techniques emerged in recent years. In a nutshell, a word is represented by a numerically-encoded vector of sufficient length, i.e., embedded in a high-dimensional feature space. The first embedding property is to make semantically similar words have numerically close feature vectors. Word embeddings also established rudimentary arithmetic operations for semantic reasoning, derivation and prediction. A notable example of an arithmetic operation via word embeddings is “king”-“man”+“woman” = “queen”. While the second property is more often used elsewhere on short sentence analysis or completion, we use the first property to establish term associations for theme-based classification of documents.

### 2.3.1 Word Embeddings

We briefly review two word embedding schemes: **Word2Vec** and **FastText**. We utilize the latter, which follows and improves upon the former.

**Word2Vec**

**Word2Vec** was introduced and popularized by Mikolov et al. [8]. Word embeddings allow the semantic meaning of a word to be represented as a real-valued vector. One of the key properties of word embeddings is their ability to represent words in a vector space such that semantically similar words are represented by geometrically
close vectors.

The key idea behind the training of word embeddings was based off an idea proposed by Harris [5] in a publication entitled “Distributional Structure.” The theory proposed by Harris was that words that appear in similar contexts also have similar meanings.

Word2Vec by Mikolov et al. in 2013 was the first popular implementation of this idea. Mikolov and his group at Google Research proposed two neural-network based models for producing word embeddings. The overall goal of Word2Vec is to produce a space of real-valued vectors to represent the semantic meanings of terms. Training is done via context windows around a certain word, as illustrated in Figure 2.2:

Source Text

```
The quick brown fox jumps over the lazy dog.
```

```
The quick brown fox jumps over the lazy dog.
```

```
The quick brown fox jumps over the lazy dog.
```

```
The quick brown fox jumps over the lazy dog.
```

**Figure 2.2:** Word2Vec Training Window - Referenced from Chris McCormick’s Word2Vec Tutorial

In Figure 2.2, the blue highlighted term in each line is referred to as that window’s “target” word. A window size of two is used, meaning that the two words before and after the target, if they exist, are used as “context” words. The Word2Vec model
ultimately generates word embeddings in the form of 300-dimensional vectors for roughly 2 million terms, as discovered through Google’s Common Crawl. The size and scope of this dataset encompasses many contextual uses of these terms, which increases our confidence that words with similar semantic meanings will have related contexts regardless of whether the corpora we are using provides similar contexts for semantically related terms. The vector space of 300 dimensions was found to be just large enough to capture diverse meanings while remaining a manageable size.

Word2Vec randomly initializes the embeddings for each term, then trains and refines the term representations via context windows from each document in the corpora. There are two forms of training that occur, Skip-Gram and Continuous Bag of Words (CBOW). Training is done on two neural networks. The first layer $W$ is a $|V| \times N$ weight matrix, where $|V|$ represents the number of unique terms in the training data and $N$ represents the number of word embedding dimensions. It is important to note that this representation layer $W$ will also be used as the final word embedding matrix. In the case of Word2Vec, $N = 300$. The second layer $W'$ is a $N \times |V|$ matrix that maps a vector representation back to words used in natural language.

**Continuous Bag of Words**

The question CBOW seeks to answer is: given a set of context words, can we predict the target word? For example, looking at Figure 2.2, we should be able to predict the word “fox” given the terms “quick,” “brown,” “jumps,” and “over.”

First, a one-hot encoded input is generated for each context word. This one-hot encoded input serves as a lookup vector into the hidden layer $W$. The product of the lookup vector with the hidden layer $W$ will return the current term representation for the term. This is illustrated in Figure 2.3.
Given the N-dimensional representations for each context word, a feature-wise average is taken. That is, we perform this one-hot lookup for the terms “quick,” “brown,” “jumps,” and “over” and then take a feature-wise average of the vectors. We then multiply this averaged vector with the second layer weight matrix $W'$. The result is a $1 \times |V|$ output vector $Y$ of probabilities, where $Y(i)$ is the probability that term $i$ is the predicted word given the context. With this vector, we can judge the likelihood that ”fox” is predicted by the context words in the input vector: in this case, “quick,” “brown,” “jumps,” and “over.” The end goal is to maximize the probability produced for the particular target word. In this specific case, we want the probability for “fox” in the vector $Y$ to be maximized.

The full CBOW model is illustrated in Figure 2.4.
Skip-Gram

The question for Skip-Gram is essentially the opposite of the CBOW model: given a target word, can we predict the context words? The Skip-Gram model has a nearly identical process to CBOW, with the only difference being that the one-hot lookup vector corresponds only to the input term. Then, instead of optimizing probabilities in the output vector $Y$ for one particular term, we aim to maximize probabilities for all of the context terms within the window. For example, given the input vector for “fox,” we aim to maximize probabilities for the terms “quick,” “brown,” “jumps,” and “over.”

The full Skip-Gram model is illustrated in Figure 2.5.
FastText

FastText by Bojanowski et al. [2] was an improvement on Word2Vec released in 2016, also under the direction of Piotr Bojanowski and Tomas Mikolov. FastText is trained on the skip-gram model proposed by Word2Vec.

The key difference between the two is the size of the smallest unit upon which they can be trained. In Word2Vec, words are the smallest unit whereas FastText treats each word as a composition of n-grams. Instead of forming an embedding vector for each term, an embedding is generated for each subword n-gram. Then, each term can be treated as a composition of n-gram embeddings. For example, “apple” could be viewed as the sum of the embeddings for “ap”, “app”, “ple”, “le”, etc. This goes towards relating roots with similar roots.

2.3.2 Semantic Term Association via k-NN

We use the FastText embeddings for term associations. One of the key advantages of this embedding is that it allows for morphological variations of words to be captured correctly. Consider the words “happy” and “happiness”. Word embeddings
such as Word2Vec could generate different embeddings for “happy” and “happiness” despite the fact that there is only a slight alteration in the word structure. With FastText, these morphological variations between terms such as “happy” and “happiness” could be captured through the subword-based embeddings of “happ” and “hap” in addition to other shared n-grams. As a result, FastText is able to create stronger near-neighbor relationships between similar words that have slight morphological variations.

The near neighbor relationships gathered from FastText word embeddings serve as the foundation of our term weighting method. For each term, we find its k-nearest neighbors to form a “semantic element.” Then, for each term, we “blur” its weight across the other terms in its semantic element. The improvements presented by this blurring process will be shown in Section 2.4.

2.4 Term Weighting via Gaussian Blurring

We use a novel term reweighting via a Gaussian blurring process to improve associations between documents that share similar themes. This technique is the solution we present toward being able to better represent documents for the purpose of theme identification.

Our semantic term blurring technique draws inspiration from image processing, where the Gaussian blur filter is often used on images to remove high frequency noise while increasing correlations across different areas of an image.

I previously noted the issues with the traditional bag of words representation, as shown in Table 2.1. The first step in our feature transformation process is to nullify the term frequencies of the frequency-based representation model. We use a 0-1 binary value scale where a 1 indicates whether a document uses a particular term or not. This is shown in Table 2.2.
After nullification of term frequencies, there is still no correlation between any of the documents. However, using word embeddings, we are able to exploit the neighborhood relationships between words with similar semantic meanings. We refer to these near-neighbor terms as “semantic elements.” For each term in the corpus, we find its k-nearest neighbors among other terms that exist in the corpus. We utilize the full word embedding representations for each term in this k-nearest neighbors search.

Using a term and its k-nearest neighbors, we “blur” its weight across its most semantically similar terms in the corpus. More specifically, the weight for a particular term is distributed across its neighbors via a Gaussian distribution. An example of this “blurring” effect can be seen in Figure 2.6.

Performing this semantic “blurring” process, we gather a new document representation matrix in the form shown in Table 2.3.
After semantic blurring across terms for each document, we now have documents with stronger associations. The exact calculation and process by which the semantic blurring occurs is a focus of the thesis by Martorano [7].
We developed an architecture platform, i.e. an analytic engine, for the unsupervised theme-based classification process. This process can be split into two distinct phases. Phase 1 starts with raw document data in a corpus and generates the DSE matrix, which describes the documents in terms of our semantic elements. Phase 2 consists of reducing the document feature dimension space and subsequent document clustering and revision.

Our architecture was built on top of KIWI, a software package developed by Pitsianis and Sun [9].
3.1 Phase 1 - From Corpus to DSE Matrix

The first phase of our architecture is shown in Figure 3.1:

![Figure 3.1: Architecture - Phase 1: From Corpus to DSE Matrix](image)

### 3.1.1 Corpora Collection

For document collection, we make use of a variety of APIs including ArXiv, Scopus, Pubmed, and Google Scholar wrapped in the KIWI software package. KIWI supports two search methods: the first is a keyword search, where relevant documents are gathered based on if they contain the keywords. The second is a seed paper search where documents are gathered if they cite a particular seed paper or are cited by that particular seed paper.

We use the seed paper search more often. Papers that directly cite or are cited by a notable paper in a particular subject are likely to have a deeper association than papers gathered through keyword search.
3.1.2 Forming DT Matrix

Our pipeline begins with two inputs: the corpus of documents that contain the title, abstract, and authors of a particular paper and the word embedding model. We use the FastText model for word embeddings, which contains 2 million term vectors trained on Common Crawl. In the FastText model, each term is represented as a 300-dimensional vector. This word embedding model can be downloaded at https://fasttext.cc/docs/en/english-vectors.html. We store the word embedding data in the form of a MATLAB Map container, where the key is a term and the value is the 300-dimensional representation of that term.

To form our initial DT matrix, we scan over the documents in our corpus. For each document, we look only at the abstract and title. We filter out any punctuation from each document to preserve full terms. We represent each document by a custom MATLAB object. In each of these objects, we store a MATLAB Map container where the keys are terms and the values are the number of times each term is used within that document. The MATLAB Map object allows us to store not only the unique set of terms used for each document but frequency information as well. We are then able to gather the set of all unique terms used within the entire corpus by performing a global union operation across the unique terms used by each document. I will denote the set of all unique terms used within the corpus as $T_{complete}$. However, we only keep words that are contained in the FastText embedding space. I will denote the FastText terms as $T_{FastText}$. The set of terms we keep $T = T_{complete} \cap T_{FastText}$. We are also able to fill in the frequency values of DT using the maps stored in each document object. For each document $i$, we iterate through each term used and fill in its frequency value within row $i$ of DT.

The final step in forming our complete DT is filtering out stop words as well as the most frequent/least frequent term features, a process that was described in
section 2.2. The output of this stage in our architecture is a DT matrix of size $|D| \times |T|$, where $D$ is the set of documents in the corpus and $T$ is the set of selected terms.

3.1.3 Forming DSE Matrix

Forming the DSE matrix from the DT matrix is an innovative but straightforward process. To form DSE, our routine takes in three inputs: the DT matrix, the terms $T$, and a matrix $TV$ that contains the full 300-dimensional representation for each term in $T$. $TV$ is formed from looking up the 300-dimensional representations for each term from the word embedding map that we generated.

To form the DSE matrix, we first turn the term frequencies from DT into a binary valued scale that indicates only whether a term has been used for a particular document. That is, an entry $(i, j)$ is 1 if term $j$ is used in document $i$ and 0 otherwise. We then find the k-nearest neighbors for each term in $T$ within the 300-dimensional FastText word embedding space. Specifically, we set a default value of $k = 4$ for the k-nearest neighbors search, but this can be modified depending on the granularity or size of the corpus. To do this, we used a fast approximate k-NN implementation built in KIWI. We form semantic elements of similar terms as illustrated in Figure 2.1.
We then use these newly formed multi-term semantic elements as document features and give them weights. The exact process and intuition by which the resulting DSE matrix is formed was discussed in Section 2.4.

3.2 Phase 2 - From DSE to Theme Clustering

The second phase of this architecture can be seen in Figure 3.2:
3.2.1 Feature Dimension Reduction

Text data is extremely high dimensional and thus reduction is indispensable. Our architecture accommodates two different types of dimension reduction. The first is a stochastic near-neighbor preserving embedding technique, t-Distributed Stochastic Neighbor Embedding, or t-SNE by van der Maaten and Hinton [11]. The second type is a geometric embedding technique, in particular the bipartite spectral embedding by Dhillon [4]. The function that we use to do the feature dimension reduction takes in two inputs: the high dimensional document representation matrix as well as an input that allows the user to specify which of these dimension reduction methods to use. The output of this stage of our architecture are the document representations in the low-dimensional embedding space.

The goal of t-SNE is to preserve stochastic near-neighbor relationships. t-SNE
will first convert Euclidian distances in the higher dimensions into probabilities that represent stochastic similarities between two document data points. Consider a point \( x_i \) and point \( x_j \). Let \( p_{ij} \) represent the probability that for a Gaussian centered at \( x_i \), that \( x_i \) would pick \( x_j \) as its neighbor. For similar points, \( p_{ij} \) will be high. More formally,

\[
p_{ij} = \frac{\exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-||x_i - x_k||^2/2\sigma_i^2)}
\]

To calculate the actual similarity between two points, a symmetric measurement of similarity is used.

\[
p_{ij} = \frac{p_{ji} + p_{ij}}{2N}
\]

Now let \( y_i \) and \( y_j \) represent the low-dimensional representation of the high-dimensional points \( x_i \) and \( x_j \). Using a Student t-distribution, the similarity between two points in the lower-dimension space is defined as:

\[
q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq i} (1 + ||y_k - y_i||^2)^{-1}}
\]

If there is a perfect low-level embedding of the higher-level data, then \( p_{ij} = q_{ij} \). The goal of t-SNE is to minimize the difference between \( p_{ij} \) and \( q_{ij} \). The cost function for t-SNE is thus defined as

\[
\hat{C} = \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1}
\]

We use t-SNE embeddings for reduction to a 3-dimensional space only. We also use a custom t-SNE routine supplied by Pitsianis and Sun [9].

Dhillon [4] represents the document and term relationship as a bipartite graph with document nodes and term nodes. In this graph, term frequencies represent the edge weights between documents and terms. To illustrate this, consider a bipartite
embedding of the document-term $DT$ matrix. If we look at the document and term cluster relationship as a bipartite graph, we have the following matrix.

$$A = \begin{bmatrix} 0 & DT \\ DT^T & 0 \end{bmatrix}$$

We then form the Laplacian

$$L = \begin{bmatrix} D_1 & -DT \\ -DT^T & D_2 \end{bmatrix}$$

$D_1$ represents the diagonal matrix of the degrees of the document nodes, and $D_2$ represents the diagonal matrix of the degrees of the term nodes.

Dhillon discusses how the min-cut of this particular graph can be relaxed to the optimal generalized partition vector which can be represented as the second eigenvector $\lambda_2$. To calculate $\lambda_2$, Dhillon uses the singular vector decomposition (SVD) of the DT matrix. This process is described below:

$$L \cdot z = \lambda \cdot D \cdot z$$

$$\begin{bmatrix} D_1 & -DT \\ -DT^T & D_2 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \lambda \cdot \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

With some algebraic manipulation this can be written as:

$$D_1^{-1/2} \cdot DT \cdot D_2^{-1/2} \cdot v = (1 - \lambda) \cdot u$$

$$D_2^{-1/2} \cdot DT^T \cdot D_1^{-1/2} \cdot u = (1 - \lambda) \cdot v$$

Which is the same as the SVD of the normalized $DT$

$$\tilde{DT} = D_1^{-1/2} \cdot DT \cdot D_2^{-1/2}$$

Using the singular vector with $\lambda_2$ as a min-cut will only yield 2 clusters. Dhillon also notes a multipartition technique where it is possible to yield more clusters by using more eigenvectors.
For the SVD option, we allow the user to specify how many eigenvectors to use for the embedding. With the SVD embedding technique, the user can embed in more than 3 dimensions. We use our own implementation of this SVD embedding technique using MATLAB linear algebra functions. Since our document feature spaces are still large and sparse, we use the MATLAB `svds` routine.

The output of this stage of our architecture is a low-dimensional representation of the documents. If t-SNE is used, then the output will be an embedding matrix of size $|D| \times 3$. If the user chooses to use the SVD embedding technique, then the embedding space is $|D| \times d$, $d \geq 1$, when $d$ is the number of singular vectors the user chooses to use, and $d$ is a modest number. We usually set $d$ to a value between 3 and 10.

### 3.2.2 Clustering via Mean Shift

After the low dimensional document embedding, we use the mean shift algorithm to generate our initial document clusters. Our architecture currently deploys only the mean shift clustering routine, which is compatible with both the t-SNE embedding method as well as the SVD embedding method. With little change, we can accommodate other clustering algorithms.

Mean shift is a clustering algorithm that assigns the data points to the clusters iteratively by shifting points towards statistical centroids or mode. Given a set of points in a feature space, mean shift iteratively assigns each point toward the direction of the closest cluster centroid. The direction to the closest cluster centroid is determined by the location of most of the nearby points. Each data point will move toward a particular cluster center until each point is assigned to a cluster [3].

More formally, mean shift considers a feature space as a probability density function, and the mean shift algorithm will locate the modes of this density function. Since mean shift is an iterative algorithm, let $x$ denote the initial estimate. A Gaus-
sian kernel function $K(x_i - x)$ will be used for the iteration, where $K$ determines the weight of the nearby points to re-evaluate the means. More formally, denote $K(x_i - x) = e^{-||x_i - x||^2}$. Using this kernel, the mean of the current window is calculated. Let $N(x)$ denote points in the current neighborhood.

\[
\text{mean}(x) = \frac{\sum_{x_i \in N(x)} K(x_i - x)x_i}{\sum_{x_i \in N(x)} K(x_i - x)}
\]

By the end of the mean shift clustering process, each point will be assigned to a mode, and these will serve as the point clustering returned by mean shift.

One important note to make is that mean shift differs from the K-means algorithm in that it does not demand to know how many clusters to look for in advance. Mean shift allows a user-specified mean shift bandwidth option. The number of clusters discovered is sensitive to the data itself as well as the specified bandwidth. A smaller bandwidth will yield a greater amount of small clusters whereas a high bandwidth will yield a smaller amount of large clusters.

In our architecture, the mean shift algorithm is used to return the initial cluster configuration in a low-dimensional document embedding space.

### 3.3 Iterative Stochastic Barcoding Cluster Revision

A novel step in our process is Iterative Stochastic Barcoding Cluster Revision. Most clustering techniques such as mean shift are only executed in reduced-dimensional spaces. However, when high-dimensional document-text data is reduced to a low-dimensional embedding space, we lose certain significant information that may be used to improve the classification. That is, since clustering techniques operate only on a low-dimensional embedding space, they are not able to fully utilize the information contained in the high-dimensional space. The motivation behind iterative barcoding
is to revise cluster configuration by using initial original high-dimensional data. We treat these low-dimensional clusters as initial estimates of the final clustering.

A particular issue with the mean shift clustering algorithm is the fact that mean shift will often return many small clusters, even singletons, which we look to alleviate in the higher dimensional space.

This iterative barcoding process is described in greater detail in the thesis by Martorano [7].

In our results, we show how utilizing high-dimensional data for stochastic bar-coding and revision not only helps us decrease the total number of final clusters, but also increases the quality of our final clusters. The new clusters returned by this stochastic barcode revision serve as our final document clusters.

3.4 Performance Evaluation

The final step is evaluating the results of our process on each corpus. For each corpus, we present the document embedding space using both DT and DSE document representations. We show how we are able to achieve an embedding space with stronger document associations using our DSE representation as opposed to the frequency based DT representation.

We also present the results of the formed document clusters using a wide variety of metrics. We calculate Precision, Recall, and F1 scores. Precision is a metric that determines, of all the positive predictions, how many were true positive predictions. Precision is calculated by $\frac{TP}{TP + FP}$. Recall, otherwise known as the true positive rate is calculated by $\frac{TP}{TP + FN}$. The F1 score is simply the harmonic mean of precision and recall, and is calculated as $2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$.

One metric that we use for evaluating our unsupervised document clusters is

\[ 1 \text{ Tiancheng Liu and his clear explanation of the precision, recall, and F1 scores were a great help in the development of this part of our architecture.} \]
known as the “purity score.” The calculation for purity itself is quite simple. Let $N$ represent the total number of documents. Let $k$ represent the number of clusters. To compute purity, each member of a cluster is assigned to the class which is most frequent in the cluster.

Purity is then computed as

$$\text{Purity} = \frac{1}{N} \sum_{i=1}^{k} \text{Max number of correct assignments in cluster } i$$

Thus, in a perfect clustering of $k$ classes in $k$ total clusters, then we would get a perfect purity score. However, it is also possible to get a perfect purity score if, given $N$ documents there were $N$ total clusters. Because of this, it is important to keep close attention to the number of clusters and appropriately re-classifying via the stochastic barcoding technique.
In this section, I present experimental results of document classification between a frequency based matrix and our $DSE$ matrix. I show that using the $DSE$ affinity matrix results in superior end document classification results. Results are presented using precision, recall, F1, and purity scores, and the parameter settings for each case is detailed.

We use two different document corpora.

4.1 Corpus 1

The first corpus consists of 14,502 documents spread out across five labels with the following breakdown:

- Brain Cancer - 3219
- Computer Vision - 4707
- Ecology - 2154
- Music - 1376
• Physical Activity - 3046

The Document-Term matrix $DT$ for this corpus is of size $14502 \times 26990$.

4.1.1 Low-Dimensional Embedding Results

Figure 4.1 shows the side-by-side embedding results between $DT$ and $DSE$ using the spectral SVD dimension reduction technique.

![Figure 4.1: 3D view of spectral embeddings of documents with DSE representation (top row) and with DT representation (bottom row). The full embedding is done in a 5-dimensional space.](image)

Both embeddings appear to show strong separation among the different categories. However, we can see that the DSE spectral embedding has a stronger sep-
aration between the music and computer vision categories than the DT spectral embedding. This property will be especially evident in the clustering results section.

### 4.1.2 Clustering Results

The composition of the resulting document clusters is shown in Figure 4.2.

**Figure 4.2:** Document clustering results between DSE and DT for Corpus 1. The clustering results presented above are post-stochastic barcode revision clusters. The figure on the left shows the composition of DSE affinity document clusters. With the DSE affinity, we began with 15 initial clusters and ended up with 5 clusters after s-barcode revision, which directly matches the number of true labels in the corpus. A minimum cluster size of 5 was set for stochastic barcode revision. For DSE, a $k = 4$ was used for the k-nearest neighbors search. The DSE affinity clearly shows that each cluster is primarily dominated by a single class of document. The figure on the right shows the DT document clusters. Using the DT affinity, we began with 16 initial clusters and ended up with 4 clusters after s-barcode revision. Meanwhile, we see from the DT results that the music class is mistakenly clustered with the computer vision papers and thus missed altogether. Both the DSE and DT results used a 5 dimensional embedding space, and both were clustered using a mean shift bandwidth of 0.001.

Table 4.1 and Table 4.2 summarize the results in terms of precision, recall, and F1 scores.
<table>
<thead>
<tr>
<th>DSE Affinity</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
<th>Num Docs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brain Cancer</td>
<td>0.986</td>
<td>0.99</td>
<td>0.988</td>
<td>3219</td>
</tr>
<tr>
<td>Computer Vision</td>
<td>0.9889</td>
<td>0.9691</td>
<td>0.9789</td>
<td>4707</td>
</tr>
<tr>
<td>Ecology</td>
<td>0.9865</td>
<td>0.9851</td>
<td>0.9858</td>
<td>2154</td>
</tr>
<tr>
<td>Music</td>
<td>0.9109</td>
<td>0.9665</td>
<td>0.9379</td>
<td>1376</td>
</tr>
<tr>
<td>Physical Activity</td>
<td>0.9842</td>
<td>0.9842</td>
<td>0.9842</td>
<td>3046</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DT Affinity</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
<th>Num Docs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brain Cancer</td>
<td>0.9792</td>
<td>0.9947</td>
<td>0.9869</td>
<td>3219</td>
</tr>
<tr>
<td>Computer Vision</td>
<td>0.7692</td>
<td>0.9906</td>
<td>0.866</td>
<td>4707</td>
</tr>
<tr>
<td>Ecology</td>
<td>0.9796</td>
<td>0.9842</td>
<td>0.9819</td>
<td>2154</td>
</tr>
<tr>
<td>Music</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1376</td>
</tr>
<tr>
<td>Physical Activity</td>
<td>0.994</td>
<td>0.9809</td>
<td>0.9874</td>
<td>3046</td>
</tr>
</tbody>
</table>

Table 4.1: DSE Affinity Post-Evaluation Measures for Corpus 1. These measures were calculated based on the 5 post s-barcoding document clusters. The final purity score for this result was 0.9791.

Table 4.2: DT Affinity Post-Evaluation Measures for Corpus 1. These measures were calculated based on the 4 post s-barcoding document clusters. The final purity score for this result was 0.8945.

These results show the improvements gained by using our DSE affinity over the DT affinity. Given the increased association between documents in the DSE affinity, we were able to capture and classify the music papers and improve our end clustering quality.

4.1.3 Stochastic Barcode Revision Results

In this section I demonstrate the improvements made upon the end clustering results using our stochastic barcode revision process. Our initial low-dimensional clustering results returned 15 clusters, with many of the 15 clusters being small clusters. Using our s-barcoding revision process, we were able to match and merge these small
clusters into the larger clusters. The results of the s-barcode revisions can be seen in Figure 4.4.

Figure 4.3: S-Barcodes for the initial 15 Clusters formed from clustering DSE in the low dimensional embedding space. The barcodes for these 15 clusters are shown before S-Barcode revision.

Figure 4.4: S-Barcodes for the 5 clusters formed after S-Barcode revision. For this revision, we set a minimum cluster size of 10.

We can see that in the 15 clusters before revision, we have many small, sparse clusters. After revision, we see that we have 5 distinctive barcodes that match the 5 distinct categories in the ground truth.

4.2 Corpus 2

The second corpus we worked is derived from articles with labels from Reuters News [1]. For this corpus, we performed some initial filtering of the documents. We only kept categories of documents with more than 100 documents and removed any entries with fewer than 50 total words.
The breakdown of this corpus is as follows:

- Mergers/Acquisitions, or ‘acq’ - 2098
- Crude oil, or ‘crude’ - 335
- Earnings/Earnings forecasts, or ‘earn’ - 3670
- Interest rates, or ‘interest’ - 182
- Money/Foreign Exchange, or ‘money-fx’ - 231
- Trade news, or ‘trade’ - 302

Our formed Document-Term matrix $DT$ for this corpus was of size $6818 \times 17546$

4.2.1 Clustering Results

Figure 4.5 compares the composition of the final document clusters formed using $DT$ and $DSE$ for Corpus 2.
Figure 4.5: Document clustering results between \textit{DSE} and \textit{DT} for Corpus 2. The clustering results presented above are post-stochastic barcode revision clusters. The figure on the left shows the composition of \textit{DSE} affinity document clusters. With the \textit{DSE} affinity, we began with 54 initial clusters and ended up with 8 clusters. A minimum cluster size of 10 was set for stochastic barcode revision. For \textit{DSE}, a $k = 4$ was used for the k-nearest neighbors search. \textit{DSE} was clustered using a mean shift bandwidth of 0.00425. The \textit{DSE} affinity clearly shows that most clusters are dominated by a single class of document. The figure on the right shows the \textit{DT} document clusters. Using the \textit{DT} affinity, we began with 55 initial clusters and ended up with 5 clusters after s-barcode revision. \textit{DT} was clustered using a mean shift bandwidth of 0.003. Both the \textit{DSE} and \textit{DT} results used an 10 dimensional embedding space.

We note here that, in the \textit{DT} affinity result, a large number of ‘interest’ and ‘money-fx’ categories were captured within one cluster. On the other hand, we see that the \textit{DSE} affinity was able to capture a large number of documents in these two classes within separate clusters. Thus, we see that the \textit{DSE} affinity is more effective than \textit{DT} in being able to separately capture every category of document.
I showed that we were able to address two fundamental, long-standing problems in typical theme-based document classification processes. We showed improvements toward theme-identification based document representation as well as a method to revise the clustering results that are obtained in a reduced dimension space.

While we were able to show that our process does create stronger associations between documents than traditional document representations, there are a number of interesting discoveries that we made throughout the course of this thesis.

One such discovery stems from how to represent semantic elements. Originally, our semantic elements consisted of non-overlapping term neighborhoods. That is, a term could only belong in one neighborhood, but we soon realized that this was problematic. A term could potentially share strong semantic relationships with different groups of words. This realization was the root of using a k-nearest neighbor search to form our semantic elements. Our Gaussian blurring technique relies on the notion that the terms within a semantic element would share strong term associations with one another. We were able to capture this by finding the k-nearest neighbors for each term in the full word embedding space. An important trait of the
semantic elements formed via k-nearest neighbor search is that semantic elements were now overlapping. A term could potentially belong to many different term neighborhoods. In forming these term neighborhoods, we typically used a small value of $k$, hoping that this would help us form stronger connections between terms. If we performed a k-nearest neighbors search using a large value of $k$, we could potentially get term neighborhoods with decreased association. Some examples of strongly associated small term neighborhoods were displayed in Figure 2.1. The choice to use a k-nearest neighbors representation of a semantic element proved to be a key factor in our improved results.

I also presented the architecture we developed in MATLAB to perform this document clustering process from corpora collection to cluster revision. This pipeline was the result of a year’s worth of active development along with the research process. I would like to specifically thank Alexandros Iliopoulos and Tiancheng Liu once again for all the help they provided in aiding us in the development of our architecture. This architecture leaves room for accommodating different term association schemes, dimension reduction methods, and clustering methods. It can be transported to another programming platform. In its future version, we would like to add a convenient user interface.

I hope to continue this work as I enter into industry.
Bibliography


Biography

• David W. Yan
• I was born in Jilin, China on April 14, 1997.
• I will graduate with a B.S. in Computer Science, Duke University, 2018.
• I have had the great fortune to work with Xiaobai Sun and her group for 2 years of my Duke career. I could not have been luckier.
• Post-Graduation Plans: I will be a software engineer at Evernote in Redwood City, California.