CS691CL - Computational Linguistics: Syntax and Semantics
Final Project Report

Document Similarity Measures Utilizing Syntactics and Semantics as well as Embeddings

Nicholas Monath, Klim Zaporojets, Niklas Shulze

May 2, 2014

1 Introduction

There is no shortage of online databases of documents containing valuable information, but there is a need for more tools to organize this data to provide users a more accessible interface than a standard keyword search. Providing solutions to this problem of information overload has been the focus of years of research in information retrieval, natural language processing and machine learning in general. In particular, much work has been done on defining document similarity measures, which determine the relatedness of two documents based on their text content. Typical approaches to this problem are based in word usage statistics and are not able to capture the essence of a text, as the syntactic and semantic relationships of the words are disregarded. In this project, we will make use of dependency parsing and automatic semantic role labeling to extract syntactically and semantically related phrases from text. Using these phrases, we hope to create a more robust document representation and similarity measure than the traditional approaches.

We would like to apply our proposed document similarity measure to the problem of identifying relatedness between scientific research papers. The similarity of scientific research papers is a less studied problem than document similarity in general and a robust measure is in high demand [**?**]. Such a measure would be useful in both research paper recommender systems and search engines.

The rest of this document is organized as follows: a review of related work done in this field and explanation of how our work differs from this previous research; our proposed approach to defining a document similarity measure; and an explanation of our proposed experiments.

2 Related Work

Document similarity is a much studied problem in the field of information retrieval with a wide range of applications, such as document classification and clustering, searching in large unorganized datasets, and recommendation systems. The problem of document similarity extends beyond the measurement of relatedness of *unstructured* documents containing only text to *structured* documents, which contain hyperlinks and other annotations [**?**].

The most common approach to unstructured text document similarity is a vector space, statistics based method known as the *bag-of-words* approach. The details of this approach are represented as a

foundation for our model in Section 4.1.1. Work in [?] shows the effectiveness of this simple representation on a number of datasets in the problem of document clustering. Another common technique is the representation of documents with a *language model*. First presented in [?], language models make the assumption that the collection of individual terms in a document is a sample from a probability distribution. This approach is particularly useful in unsupervised document clustering. Language models are often made more robust by adding additional information such as underlying topic labels obtained algorithms such as *Latent Dirichlet Allocation* (LDA) or *probabilistic latent semantic analysis*. These models have been shown to be effective in document retrieval and categorization by [?]. These approaches are also used in our approach and presented in more detail in Section 4.1.1.

In these traditional approaches, the syntactic and semantic relationships of words and phrases in the text are ignored. The models are based on statistical information on the frequency of the occurrence of word sequences or n-grams. Often unigrams (i.e. $n = 1$) are used and so the measure is the frequency of the occurrence of single words. Bigrams (two-word sequences) and trigrams (three-word sequences) are also commonly used. Intuitively, the substitution of *n-grams* with syntactically related groups of words is a logically sound choice. Often, *n-grams* are disparate sequences of terms, while syntactically related groups of words can provide a more robust feature that preserves the semantic meaning of phrases. Initial work was done on this in the late 1990s, such as [?], [?]. The recent works of [?] and [?], show the method can have significant benefits over the traditional *n-gram* approach. Nastase et al in [?] use syntactically related pairs of words obtained using a dependency parser as the base elements in a vector-based bag-of-words approach to perform the task of supervised text classification on the Reuters-21578 dataset. They also experiment with a combination of using the syntactically related pairs and unigrams both with and without syntactic labels. Koster et al in [?] use a similar method of combining syntactically related triples obtained with a dependency parser with unigrams in a vector based model to perform the task of patent document classification.

The most recent work related to document similarity of scientific papers can be found in [?] and [?], it uses a unigram based language model to represent the documents. The model treats the text as *semi-structured*, taking advantage of the additional features of the keywords of the paper, authors' names, and the name of the journal in which the paper appears.

Our work separates itself from these previous studies in that we propose to use syntactically and semantically related sequences of words extracted from the text, using not only dependency parsing, but also automatic predicate argument structure detection, both in a language model and a vector based representation of documents. In creating a language model, we hope to provide a more robust representation of documents using LDA and other techniques. We will also apply the approach to unsupervised document clustering and pairwise document similarity for applications such as recommender systems.

3 Document Representations

3.1 Document Preprocessing

In order to extract dependency pair and predicate argument structures from documents for use in a feature structure, we must run documents through a parsing algorithm. The parser we used in *ClearNLP*. We set the parsing mode to be *Semantic Role Labeling*, which produces not only an SRL of the text, but also the dependency structure. The parser also provides additional information about each word it the text: the lemmatized form, part of speech tag, dependency label, semantic role label, and other features.

3.2 Bag of "Units" Document Representation

In this work we extend the typical vector space model of a bag of words feature representation to include rich units than single words, namely dependency (head-modifier) pairs, and predicate argument

structures. By *units* or *base units* we are referring to the terms whose presence/absence in a document determine the values of the feature vector of the document, its document representation. In this way, each unit corresponds to an entry in the feature vector of a document. The value of that vector can be binary, which means the values of the feature vector are 0 or 1 depending on whether or not a unit appears in the document. It can be tf-idf, in which the values of the feature vector are the term-frequency-inverse-document-frequency of a unit. In this project, we used the augmented tf-idf value presented HERE (CITE).

3.3 Word Representation

We define an object structure representing a single *word* to have the attributes of the *word form* that appears in the document, the *lemmatized form* of the word, and the *part of speech tag* of the word.

3.4 Dependency Pair Representation

We define an object structure representing a head-modifier dependency pair to be an ordered tuple of *words* with the structured defined above. The first word in the tuple is the *head* of the dependency pair, and the second word in the tuple is the modifier of the dependency pair.

3.4.1 Extracting Dependency Pairs from Parsed File

The preprocessed document files parsed by the *ClearNLP* parser have the following format:

Given the input document:

```
My sister thought John Updike's writing was offensive. I disagreed.
```

The dependency pairs are discovered using the values in the

1	My	my	PRP$	_	2	poss	_
2	sister	sister	NN	_	3	nsubj	3:A0
3	thought	think	VBD pb=think.01	0	root	_	
4	John	john	NNP	_	5	nn	_
5	Updike	updike	NNP	_	7	poss	_
6	's	's	POS	_	5	possessive	_
7	writing	writing	NN	_	8	nsubj	8:A1=PPT
8	was	be	VBD pb=be.01	3	ccomp	3:A1=PPT	
9	offensive	offensive	JJ	_	8	acomp	8:A2=PRD
10	.	.	.	_	3	punct	_
1	I	I	PRP	_	2	nsubj	2:A0=PAG
2	disagreed	disagree	VBD	pb=disagree.01	0	root	_
3	.	.	.	_	2	punct	_

The dependency pairs are extracted making use of the intra-sentence word ID numbers (first column) and the corresponding dependency information (fifth column).

3.5 Feature Options Common to All Representations

Some feature options are common across both our baseline (bag of words) and proposed approaches. These options

The first options define equivalence between words

- **Lemmatization**:
- **Case Sensitivity**: Determines if capitalization effects the equality of words
- **Part of Speech Tags**:

3.6 Bag of Words Representation

As a baseline, we implemented a bag-of-words vector space model. In this model, we define a feature

4 Proposed Approach

There are two components of a **document similarity measure**: first, the definition of a *document representation*, which specifies the data structure used to represent the document in a way which is meaningful to a machine; and second, a *distance function*, which assigns to a given pair of documents a real number representing how different the two documents are. We propose several new document similarity measures which incorporate the output of linguistic tools such as dependency parsing and automatic semantic role labelers in traditional statistics-based approaches. To determine the effectiveness a document similarity measure, the measure must be applied to an information retrieval task. We will evaluate our proposed similarity measures with the tasks of **document clustering** and **document classification**.

4.1 Proposed Similarity Measures

4.1.1 Traditional Approaches & Baselines

The following three methods are used as baselines in our experiments. We use a unigram, bigram and trigram version of each approach. These approaches are also used as the foundation for each of our proposed similarity measures presented in Sections 4.1.2, 4.1.3, and 4.1.4.

Bag of Words

In a vector space n-gram bag of words approach, the *document representation* is a feature vector of length M, where M is the number of unique n-grams in the entire collection of documents. Each element of the feature corresponds to a word and value of the feature for each element is typically either binary or the term's tf-idf (term-frequence-inverse-document-frequency) value. Three *distance functions* can be used, the cosine distance, Jaccard Coefficient, and Pearson Correlation Coefficient as presented in [**?**].

Language Model

Another canonical approach to document similarity is to use a n-gram language model. The *document representation* is the language model, which, for document d_i is defined as:

$$\mathcal{D}_i(w) = \lambda P(w|d_i) + (1-\lambda)P(w|C) \tag{1}$$

where w is an n-gram and C is the entire collection of documents. The *distance function* is the Kullback-Leiber divergence:

$$\mathrm{KL}(\mathcal{D}_i \| \mathcal{D}_j) = \sum_i \ln\left(\frac{\mathcal{D}_i(w)}{\mathcal{D}_j(w)}\right)\mathcal{D}_i(w) \tag{2}$$

Language Model with LDA

In this approach, the n-gram language model is made more robust with the use of Latent Dirichlet Allocation (LDA) [**?**]. Theoretically, LDA will allow us to discover latent topics in the documents, and further improve the language model of an article based on the revealed topical information. The *document representation* is now defined as:

$$\mathcal{D}_i(w) = \lambda_1 P(w|d_i) + \lambda_2 P(w|C) + \lambda_3 P(w|T) \tag{3}$$

where T is the set of topics underlying document d_i as defined by LDA. The *distance function* is still the Kullback-Leiber divergence.

4.1.2 Making use of Dependency Relations

A dependency parser such as the *Stanford Parser* [**?**] can be used to extract the inter word dependencies in a sentence. For example, given the sentence:

The quick brown fox jumped over the lazy dog

The outputted dependencies are:

det(fox-4, The-1)	amod(fox-4, quick-2)
amod(fox-4, brown-3)	nsubj(jumped-5, fox-4)
root(ROOT-0, jumped-5)	det(dog-9, the-7)
amod(dog-9, lazy-8)	prep_over(jumped-5, dog-9)

The word pairs of each dependency relation are used in place of n-grams in the *document representations* of the three document similarity measures described in Section 4.1.1. We will experiment with including the type of relationship along with the word pairs. Also, we will experiment with using the dependency pairs along with the unigrams in the three approaches.

4.1.3 Making use of Predicate-Argument Structure

A automatic semantic role labeler such as the *Illinois Semantic Role Labeler (SRL)* [**?**] can be used to determine the predicate argument structure of English sentences. For example, given the sentence:

After eating dinner, the quick brown fox saw the lazy dog, who was still sleeping.

The following predicate argument relationships are given by the Illinois Semantic Role Labeler:

Relation #1:	eat.01	meal [A1]	consumer/eater [A0]	
	eating	dinner	the quick brown fox	
Relation #2:	see.01	viewer [A0]	thing viewed [A1]	temporal [AM-TMP]
	saw	the quick brown fox	the lazy dog	after eating dinner
Relation #3:	sleep.01	sleeper [A0]	sleeper [R-A0]	temporal [AM-TMP]
	sleeping	the lady dog	who	still

The groups of terms making up the relations and arguments are used in place of the n-grams in the *document representations* of the three document similarity measures described in Section 4.1.1. We will experiment with including the role labels along with each group of terms. Also, we will try using unigrams in addition to these groups of terms in the three approaches.

4.1.4 Making use of Word2Vec

Vector space word models are useful in determining the syntactic and semantic relatedness between words. In these models, each word in a corpus is assigned a position in a high-dimensional space $\mathbb{R}^N$ through a training process. The hope is that words that are related to one another are placed at a near by in $\mathbb{R}^N$. The relatedness (or rather *difference*) between two words is typically measured by the cosine distance between their two vectors. Word2Vec is a state of the art vector space word model and training process [**?**]. We propose two modifications to the previously described models that make use of a vector space word model.

The first method is to perform clustering of the words and to use the centroid of each cluster in place of n-grams in the previously described approaches. Specifically, we let $\mathbf{W}$ be the set of words that appear in one or more of the collection of documents $d_1, d_2, \ldots, d_n$ and we let $\mathbf{V}$ be the set of vectors corresponding to the entries in $\mathbf{W}$. We cluster $\mathbf{V}$ using an algorithm such as k-means or DBSCAN and so each entry in $\mathbf{V}$ is associated with a specific cluster. We then calculate the center point or centroid of that cluster. Then each word in the corpus is replaced with the centroid of its associated cluster and the three approaches described in Section 4.1.1 are used.

A second method, but related method, is a less strong form of clustering. Clustering is done in just two passes through the documents. For each word, we find the closest k words in the vector space defined by Word2Vec. If the word is not already associated with a cluster and if any of these k words appear in the collection of documents, a cluster is defined for the related words. If any of the k words appear in other clusters, the clusters are merged together. We then calculate the center point or centroid of that cluster. Then each word in the corpus is replaced with the centroid of its associated cluster and the three approaches described in Section 4.1.1 are used.

According to [**?**], we can define multiple word phrases in vector space as the sum of the vectors of the individual words. In so doing we can apply these two methods to the multi-word phrases extracted from the dependency parsing and semantic role labeling.

4.1.5 Text Processing Tools

Many of the current papers convert words to their lemmatized form that is removing inflections so that each word is in its base form. This helps to reduce the feature space and make the feature vectors (or probability distributions) more descriptive. This process will take place after the dependency parsing and semantic role labeling. We can experiment with whether or not it will be used in the Word2Vec model.

5 Experiments

Due to limited computational resources and the immense size of the datasets we were not able to evaluate how our methods did compared to the state of the art. Instead, we ran a set of experiments comparing our methods to a baseline unigram bag of words approach. These experiments will provide insight into whether or not the additional features extracted using dependency pairs and predicate argument structures is beneficial to a document similarity measure.

In these experiments, we compared the performance of four of our feature representations. We performed the experiments on the feature definitions with each of the following combinations of units

- Words
- Words and Dependency Pairs
- Words and Predicate Argument Structures
- Words, Dependency Pairs, and Predicate Argument Structures

We experimented with both a binary and a tf-idf valued feature vectors. Due to computational constraints, we used only the lemmatized version of words and did not use part of speech, dependency labels, nor argument labels. Features that appeared in only one document where removed from the feature definition.

5.1 Document Clustering

5.1.1 General Setup

The problem of **document clustering** is defined as the automatic organization of documents into logical groups based on latent features. For evaluation purposes, we use documents that are already divided into classes. We then use a clustering algorithm to see how well the clusters represent the true classes of the documents.

Formally, the experimental set up is as follows. Given a collection of documents D_1, D_2, ... D_N, which have been preprocessed in the method described above. The first step of the experiment is to define a feature for the documents, i.e. union of the set of all units (words, dependency pairs, predicate/argument structures), which appear in at least one document in the collection. We then extract a feature vector from each of these documents to produce the set of features for the documents X_1, X_2, ... X_N. Each document has an associated class label y_1, y_2, ..., y_N, which will be used to in evaluating the performance of the clustering algorithm.

In these, experiments we clustered the document features X_1, X_2, ... X_N using the k-means clustering algorithm with k being the number of unique class labels for the documents. For a distance function, we used the Cosine Similarity in that we normalized X_1, X_2, ... X_N to unit length and used a Euclidean distance measure. The output of the clustering algorithm is a cluster id label for each of the N documents, which we will refer to as C_1, C_2, ..., C_N.

5.2 Evaluation Measures

Following the experiments in [**?**] and [**?**], we evaluated the clustering of the documents with the measures of purity, normalized mutual information, and the adjusted Rand index. The definitions of these metrics are `http://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html`

5.2.1 Purity

Purity is a measure of cluster quality, that is how well the classes are defined into clusters. If we refer to Y as the set of unique class labels in the data set, and refer to W_i as the class labels of the documents in the i^{th} cluster. The purity is defined as follows:

$$\text{Purity}(W_{1:K}, Y) = \frac{1}{N}\left(\sum_{n=1}^{N} \max_{y \in Y}\left(\sum_{w \in W_n} [w = y]\right)\right) \tag{4}$$

where $[w = y]$ is the indicator function–it evaluates to 1 if w is equal to y and evaluates to 0 otherwise.

5.2.2 Normalized Mutual Information

The normalized mutual information between the cluster indexes C_1, C_2, ..., C_N and the class labels Y_1, Y_2, ..., Y_N is measured. This is defined as:

$$\text{NormalizedMutualInformation}(C_{1:N}, Y_{1:N}) = \frac{I(C_{1:N}, Y_{1:N})}{H(C_{1:N}) + H(Y_{1:N})} \tag{5}$$

Where I is the mutual information:

$$I(C_{1:N}, Y_{1:N}) = \sum_i \sum_j P(C_i, Y_j) \frac{P(C_i, Y_j)}{P(C_i)P(Y_j)} \tag{6}$$

$$\tag{7}$$

and the entropy H is:

$$H(C_{1:N}) = -\sum_i P(C_i) \log P(C_i) \tag{8}$$

$$H(Y_{1:N}) = -\sum_i P(Y_i) \log P(Y_i) \tag{9}$$

$$\tag{10}$$

and the probability distributions are their maximum likelihood estimates.

5.2.3 Adjusted Rand Index

The adjusted Rand index is a version of the Rand index that is adjusted for the chance grouping of elements. In general the Rand index evaluates the similarity between $C_{1:N}$ and $Y_{1:N}$. Letting $\mathcal{Y}$ be the set of unique class labels and $\mathcal{C}$ be the set of unique cluster labels, the adjusted Rand Index is calculated as:

5.3 Reuters-21578

We selected a portion of the test cases of the Mod-Apte split of the Reuters-21578 to use in a clustering experiment. We selected those documents from the top 8 most frequently appearing classes and that only belonged to one class. This selection was inspired by the experiments in **??** and **??** The distribution of documents in the data set can be see in figure **??**. The results for each of our feature vectors shown in Table 1.

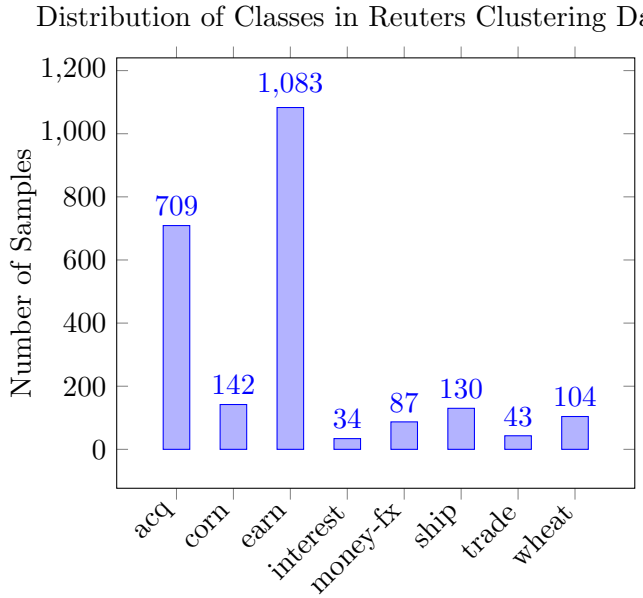

Table 1: Reuters Clustering Results

Feature Type	Value Type	Purity	Normalized Mutual Information	Adjusted Rand Index
Words	Binary	0.72727	0.44211	0.20537
Words	tf-idf	0.761149	0.50932	0.33894
Words & Dep. Pairs	Binary	0.711835	0.42011	0.15659
Words & Dep. Pairs	tf-idf	0.699828	0.34044	0.05001
Words & Pred. Arg.	Binary	0.761149	0.490585	0.32799
Words & Pred. Arg.	tf-idf	**0.786878**	**0.5485359**	**0.37705**
Words, Dep. Pairs, & Pred. Arg.	Binary	0.729845	0.442120	0.22446
Words, Dep. Pairs, & Pred. Arg.	tf-idf	0.75	0.491923	0.30966

5.4 Brown Corpus

The next clustering experiment we ran was on the Brown Corpus. We ran the same document clustering experiment described earlier on the 500 document corpus. The Brown Corpus has a slightly more even class distribution than the Reuters Corpus:

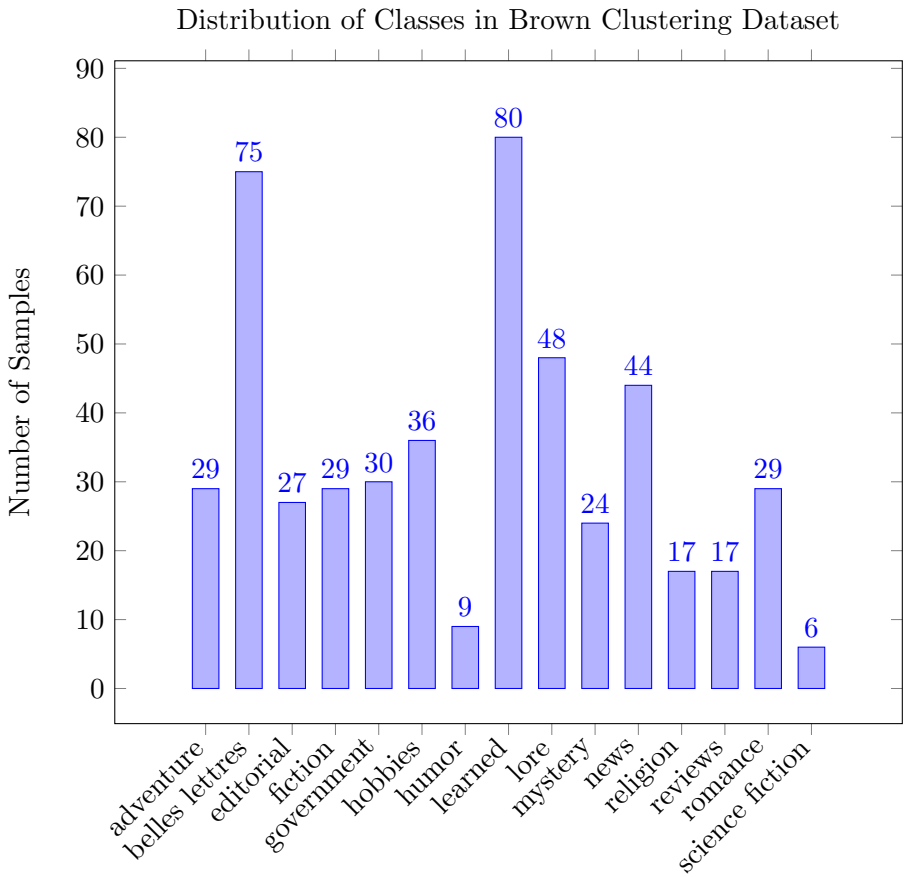

Table 3: Class Assignments in Scientific Paper Dataset

Paper Title	Class Name
Penn Treebank Overview	Phrase Structure Grammar
Government & Binding Theory Introduction	Phrase Structure Grammar
Introduction to Prop Bank	Predicate Argument Structure
On the Semantic Content of the Notion of Thematic Role	Predicate Argument Structure
Tree Adjoining Grammars	Tree Adjoining Grammars
Domains of Locality	Tree Adjoining Grammars
Synchronous Tree Adjoining Grammars	Tree Adjoining Grammars
Supertagging: An Approach to Almost Parsing	Tree Adjoining Grammars
Introduction to WordNet: An On-line Lexical Database	WordNet
Nouns in WordNet: A Lexical Inheritance System	WordNet
Adjectives in WordNet	WordNet
English Verbs as a Semantic Net	WordNet
Design and Implementation of WordNet Lexical Database and Searching Software	WordNet
Never Ending Language Learner	Ontology and Taxonomy
Which Noun Phrases Denote Which Concepts	Ontology and Taxonomy
Combinatory Categorial Grammar by Steedman and Baldridge	Combinatory Categorial Grammar
Equivalence of Four Extensions of Context Free Grammars	Combinatory Categorial Grammar
Vector Space Semantic Parsing	Combinatory Categorial Grammar
FrameNet II	Frame Semantics in FrameNet
A Frames Approach to Semantic Analysis	Frame Semantics in FrameNet
Learning to Map Sentence to Logical Form	Distributional Semantics
Relation Extraction with Matrix Factorization and Universal Schema	Distributional Semantics

Table 2: Brown Corpus Clustering Results

Feature Type	Value Type	Purity	Normalized Mutual Information	Adjusted Rand Index
Words	Binary	0.72727	0.44211	0.20537
Words	tf-idf	0.761149	0.50932	0.33894
Words & Dep. Pairs	Binary	0.711835	0.42011	0.15659
Words & Dep. Pairs	tf-idf	0.699828	0.34044	0.05001
Words & Pred. Arg.	Binary	0.761149	0.490585	0.32799
Words & Pred. Arg.	tf-idf	**0.786878**	**0.5485359**	**0.37705**
Words, Dep. Pairs, & Pred. Arg.	Binary	0.729845	0.442120	0.22446
Words, Dep. Pairs, & Pred. Arg.	tf-idf	0.75	0.491923	0.30966

5.5 Scientific Paper Dataset

We collected a small dataset of the abstracts, titles, and authors [1] of the papers that we read in this courses. Each paper was given the class label of the unit it was a part of–unit being "Tree Adjoining Grammar" or "Ontology and Taxonomy". The name of each paper and the class label assigned is shown in Table 3. While this is a relatively small dataset of just 22 documents, the domain was significantly different enough from the previous two datasets to be worthy of experimentation. The results are shown in Table 4

Note that for a small number of these papers, there was no abstract. In these cases, if the introduction was brief, the introduction was used in lieu of the abstraction, otherwise the document was not a part of the dataset. Also note that the conversion from PDF to plain text was done manually.

[1] There are no meta-data tags of in the documents, only text. This experimental setup was designed to partially mimic the experiments in **??**

Table 4: Scientific Paper Clustering Results

Feature Type	Value Type	Purity	Normalized Mutual Information	Adjusted Rand Index
Words	Binary	0.72727	0.44211	0.20537
Words	tf-idf	0.761149	0.50932	0.33894
Words & Dep. Pairs	Binary	0.711835	0.42011	0.15659
Words & Dep. Pairs	tf-idf	0.699828	0.34044	0.05001
Words & Pred. Arg.	Binary	0.761149	0.490585	0.32799
Words & Pred. Arg.	tf-idf	**0.786878**	**0.5485359**	**0.37705**
Words, Dep. Pairs, & Pred. Arg.	Binary	0.729845	0.442120	0.22446
Words, Dep. Pairs, & Pred. Arg.	tf-idf	0.75	0.491923	0.30966

5.6 Analysis

6 Document Classification

6.1 General Problem Description and Evaluation Criteria

In general the problem of document classification is defined as: given set of training samples, pairs of documents and associated class labels: (D_1, Y_1), (D_2, Y_2), ..., (D_N, Y_N), predict the class label for a document D_{N+1} not a part of the training set. From the training documents we first define a feature vector in the same way described in clustering experiment. We then extract features from the training set to get the set of features for all the training documents: X_1, X_2, ..., X_N. We use the same feature definition to extract features from the testing documents.

The classifier we used in this experiment was an SVM with a Linear Kernel. We used a value of $C = 1$. In future experiments we will use a lower value of C. We again used Cosine similarity by normalizing the features and using the Euclidean distance.

The classification task was evaluated on the standard criteria of accuracy, and the average per-class precision and recall scores.

6.2 NewsGroups Classification Experiment #1

The entire Twenty News Groups Corpus consists of about 18 thousand documents, about 11 thousand training and 7 thousand testing. The data set was too large to run our machines and so we selected a smaller portion of the data set to use in our experiments. From the original training and testing sets, we created new sets of the first 200 samples of the 15 most frequently occurring classes. The results from our experiment are shown in Table 5.

Table 5: NewsGroups Classification Experiment #1 Results

Feature Type	Value Type	Accuracy	Avg. Precision Per Class	Avg. Recall Per Class
Words	Binary	0.72727	0.44211	0.20537
Words	tf-idf	0.761149	0.50932	0.33894
Words & Dep. Pairs	Binary	0.711835	0.42011	0.15659
Words & Dep. Pairs	tf-idf	0.699828	0.34044	0.05001
Words & Pred. Arg.	Binary	0.761149	0.490585	0.32799
Words & Pred. Arg.	tf-idf	**0.786878**	**0.5485359**	**0.37705**
Words, Dep. Pairs, & Pred. Arg.	Binary	0.729845	0.442120	0.22446
Words, Dep. Pairs, & Pred. Arg.	tf-idf	0.75	0.491923	0.30966

6.3 NewsGroups Classification Experiment #2

As a second classification experiment, we selected the first 100 samples of all of the twenty classes in both the training and testing sets. The results for this experiment are shown in Table 7.

Table 6: NewsGroups Classification Experiment #2 Results

Feature Type	Value Type	Accuracy	Avg. Precision Per Class	Avg. Recall Per Class
Words	Binary	0.72727	0.44211	0.20537
Words	tf-idf	0.761149	0.50932	0.33894
Words & Dep. Pairs	Binary	0.711835	0.42011	0.15659
Words & Dep. Pairs	tf-idf	0.699828	0.34044	0.05001
Words & Pred. Arg.	Binary	0.761149	0.490585	0.32799
Words & Pred. Arg.	tf-idf	**0.786878**	**0.5485359**	**0.37705**
Words, Dep. Pairs, & Pred. Arg.	Binary	0.729845	0.442120	0.22446
Words, Dep. Pairs, & Pred. Arg.	tf-idf	0.75	0.491923	0.30966

7 Document Retrieval

As a final experiment, we ran a simplified version of a document retrieval system. The problem definition is as follows: Given a collection of documents D_1, D_2, ..., D_N, each with a class label Y_1, Y_2, ..., Y_N, query the collection on each document D_i and retrieval the top K documents with lowest Cosine distance. The precision/recall of each retrieval is scored and the average across all queries is reported. The feature vectors representing the documents are defined using the entire corpus.

7.1 Scientific Paper Retrieval

We first ran the retrieval experiment on the Scientific Paper dataset described in Section 5.5. In this experiment we retrieved the top $K = 5$ closest documents. This experiment was meant to be a precursor to the work we will do this summer replicating a similar experiment presented in [**?**].

Table 7: Scientific Paper Retrieval Results

Feature Type	Value Type	Avg. Precision	Avg. Recall
Words	Binary	0.72727	0.20537
Words	tf-idf	0.761149	0.33894
Words & Dep. Pairs	Binary	0.42011	0.15659
Words & Dep. Pairs	tf-idf	0.4044	0.05001
Words & Pred. Arg.	Binary	0.761149	0.32799
Words & Pred. Arg.	tf-idf	**0.5485359**	**0.37705**
Words, Dep. Pairs, & Pred. Arg.	Binary	0.442120	0.22446
Words, Dep. Pairs, & Pred. Arg.	tf-idf	0.491923	0.30966

7.2 Brown Corpus Retrieval

We ran an additional retrieval experiment, in which we selected the first 15 samples of the top 5 most frequently occurring classes in the Brown corpus. We then ran the retrieval experiment with $K = 15$. Note that in this case the precision and recall is the same, it is just the percentage of the top 15 documents that were of the same class of the query document.

Table 8: Brown Retrieval Results

Feature Type	Value Type	Avg. Precision	Avg. Recall
Words	Binary	0.72727	0.20537
Words	tf-idf	0.761149	0.33894
Words & Dep. Pairs	Binary	0.42011	0.15659
Words & Dep. Pairs	tf-idf	0.4044	0.05001
Words & Pred. Arg.	Binary	0.761149	0.32799
Words & Pred. Arg.	tf-idf	**0.5485359**	**0.37705**
Words, Dep. Pairs, & Pred. Arg.	Binary	0.442120	0.22446
Words, Dep. Pairs, & Pred. Arg.	tf-idf	0.491923	0.30966

8 Conclusions and Future Work

For example, given a corpus of computational linguistics research papers, we might hope that a clustering of the papers would place all the work on grammars and language representations in one cluster, the work on semantics in another, the work in morphology in a third, etc.

We will reproduce the experiment in [**?**], which uses a corpus of 209 artificial intelligence research papers manually tagged by experts. For each paper, the 30 papers considered most similar using the bag-of-words approach [2] with the cosine distance function were labeled by an expert as similar or dissimilar. A document similarity measure is evaluated by selecting, for each paper, the 30 closest papers and measure the Mean Average Precision and Mean Reciprocal Rank. We will the results of our approach to that of the authors.

We will also evaluate our document similarity measures on the Reuters-21578 corpus. Each news article in the Reuters corpus is tagged with a class label. We will perform an experiment in which we cluster the articles in the corpus and measure how well each cluster encompasses a class of article and how well each class is represented by a cluster, both in terms of precision and recall.

Finally, if time allows, we will perform an additional experiment on the clustering of research papers, in which we collect a corpus and for each paper return the top k most similar papers. The list of similar papers will then be evaluated by a human subject and the rate of dissimilar papers will be evaluated.

8.0.1 Document Classification

The problem of **document classification** is formally defined just like the general problem of classification in machine learning: Given input pairs of documents and class labels, $(d_1, c_1), (d_2, c_2), \ldots, (d_N, c_N)$, with $d_1, \ldots d_n \in \mathcal{D}$ and $c_1, \ldots, c_N \in \mathcal{C}$, where $\mathcal{D}$ is the collection of training documents and $\mathcal{C}$ is the set of class labels, assume there is a function f which maps any document d the correct class label c. The goal of learning is to approximate the function f with $\hat{f}$ a function created with the observed training data.

The authors of [**?**] present results from a document classification experiment on the Reuters-21578 dataset using a vector space model which makes use of dependency parsing. We will repeat their experiment to see how our model performs in comparison to theirs. The classification experiment is done using the 10 most frequent classes in the Reuters-21578 document collection with the ModApte split designating the specific documents for training and testing. The precision and recall scores are used to judge the performance of a document similarity measure.

8.0.2 Clustering and Classification Methods

We will use a standard set of machine learning tools to implement the clustering and classification procedures. For clustering, we will perform experiments using *k-means*, *DBSCAN*, and the *EM Algorithm*. For classification, we will perform experiments using *Support Vector Machines*, *Naive Bayes*, and *k-Nearest Neighbor* approaches. We will be programming most of our project in `Python` and will be using the machine learning tools available in `NumPy` and `SciPy`.

Note the below references includes papers that we read, but did not cite in this document. Those papers are listed for our own reference.

[2] With tf-idf as the value of the vector for each unigram