Phrase Clustering for Discriminative Learning

Dekang Lin and **Xiaoyun Wu**
Google, Inc.
1600 Amphitheater Parkway, Mountain View, CA
{lindek,xiaoyunwu}@google.com

Abstract

We present a simple and scalable algorithm for clustering tens of millions of phrases and use the resulting clusters as features in discriminative classifiers. To demonstrate the power and generality of this approach, we apply the method in two very different applications: named entity recognition and query classification. Our results show that phrase clusters offer significant improvements over word clusters. Our NER system achieves the best current result on the widely used CoNLL benchmark. Our query classifier is on par with the best system in KDDCUP 2005 without resorting to labor intensive knowledge engineering efforts.

1 Introduction

Over the past decade, supervised learning algorithms have gained widespread acceptance in natural language processing (NLP). They have become the workhorse in almost all sub-areas and components of NLP, including part-of-speech tagging, chunking, named entity recognition and parsing. To apply supervised learning to an NLP problem, one first represents the problem as a vector of features. The learning algorithm then optimizes a regularized, convex objective function that is expressed in terms of these features. The performance of such learning-based solutions thus crucially depends on the informativeness of the features. The majority of the features in these supervised classifiers are predicated on lexical information, such as word identities. The long-tailed distribution of natural language words implies that most of the word types will be either unseen or seen very few times in the labeled training data, even if the data set is a relatively large one (e.g., the Penn Treebank).

While the labeled data is generally very costly to obtain, there is a vast amount of unlabeled textual data freely available on the web. One way to alleviate the sparsity problem is to adopt a two-stage strategy: first create word clusters with unlabeled data and then use the clusters as features in supervised training. Under this approach, even if a word is not found in the training data, it may still fire cluster-based features as long as it shares cluster assignments with some words in the labeled data.

Since the clusters are obtained without any labeled data, they may not correspond directly to concepts that are useful for decision making in the problem domain. However, the supervised learning algorithms can typically identify useful clusters and assign proper weights to them, effectively adapting the clusters to the domain.

This method has been shown to be quite successful in named entity recognition (Miller *et al*. 2004) and dependency parsing (Koo *et al*., 2008).

In this paper, we present a semi-supervised learning algorithm that goes a step further. In addition to word-clusters, we also use phrase-clusters as features. Out of context, natural language words are often ambiguous. Phrases are much less so because the words in a phrase provide contexts for one another.

Consider the phrase “Land of Odds”. One would never have guessed that it is a company name based on the clusters containing *Odds* and *Land*. With phrase-based clustering, “Land of Odds” is grouped with many names that are labeled as company names, which is a strong indication that it is a company name as well. The disambiguation power of phrases is also evidenced by the improvements of phrase-based machine translation systems (Koehn *et. al.*, 2003) over word-based ones.

Previous approaches, e.g., (Miller *et al*. 2004) and (Koo *et al*. 2008), have all used the Brown algorithm for clustering (Brown *et al*. 1992). The main idea of the algorithm is to minimize the bigram language-model perplexity of a text corpus. The algorithm is quadratic in the number of elements to be clustered. It is able to cluster tens of thousands of words, but is not scalable enough to deal with tens of millions of phrases. Uszkoreit and Brants (2008) proposed a distributed clustering algorithm with a similar objective function as the Brown algorithm. It substantially increases the number of elements that can be clustered. However, since it still needs to load the current clustering of all elements into each of the workers in the distributed system, the memory requirement becomes a bottleneck.

We present a distributed version of a much simpler K-Means clustering that allows us to cluster tens of millions of elements. We demonstrate the advantages of phrase-based clusters over word-based ones with experimental results from two distinct application domains: named entity recognition and query classification. Our named entity recognition system achieves an F1-score of 90.90 on the CoNLL 2003 English data set, which is about 1 point higher than the previous best result. Our query classifier reaches the same level of performance as the KDDCUP 2005 winning systems, which were built with a great deal of knowledge engineering.

2 Distributed K-Means clustering

K-Means clustering (MacQueen 1967) is one of the simplest and most well-known clustering algorithms. Given a set of elements represented as feature vectors and a number, *k*, of desired clusters, the K-Means algorithm consists of the following steps:

Step	Operation
i.	Select *k* elements as the initial centroids for *k* clusters.
ii.	Assign each element to the cluster with the closest centroid according to a distance (or similarity) function.
iii.	Recompute each cluster's centroid by averaging the vectors of its elements
iv.	Repeat Steps ii and iii until convergence

Before describing our parallel implementation of the K-Means algorithm, we first describe the phrases to be clusters and how their feature vectors are constructed.

2.1 Phrases

To obtain a list of phrases to be clustered, we followed the approach in (Lin *et al.*, 2008) by collecting 20 million unique queries from an anonymized query log that are found in a 700 billion token web corpus with a minimum frequency count of 100. Note that many of these queries are not phrases in the linguistic sense.

Table 1 Cluster of "English lessons"

Window	Cluster members (partial list)
size=1	environmental courses, summer school courses, professional development classes, professional training programs, further education courses, leadership courses, accelerated courses, vocational classes, technical courses, technical classes, special education courses, …..
size=3	learn english spanish, grammar learn, language learning spanish, translation spanish language, learning spanish language, english spanish language, learn foreign language, free english learning, language study english, spanish immersion course, how to speak french, spanish learning games, …..

However, this does not seem to cause any real problem because non-linguistic phrases may form their own clusters. For example, one cluster contains {"Cory does", "Ben saw", "I can't lose", …..}.

To reduce the memory requirement for storing a large number of phrases, we used Bloom Filter (Bloom 1970) to decide whether a sequence of tokens is a phrase. The Bloom filter allows a small percentage of false positives to pass through. We did not remove them with post processing since our notion of phrases is quite loose to begin with.

2.2 Context representation

Distributional word clustering is based on the assumption that words that appear in similar contexts tend to have similar meanings. The same assumption holds for phrases as well. Following previous approaches to distributional clustering of words, we represent the contexts of a phrase as a feature vector. There are many possible definitions for what constitutes the *contexts*. In the literature, contexts have been defined as subject and object relations involving the word (Hindle, 1990), as the documents containing the word (Deerwester *et al*, 1990), or as search engine snippets for the word as a query (Sahami and Heilman, 2006). We define the contexts of a phrase to be small, fixed-sized windows centered on occurrences of the phrase in a large corpus. The features are the words (tokens) in the window. The context feature vector of a phrase is constructed by first aggregating the frequency counts of the words in the context windows of different instances of the phrase. The frequency counts are then converted into point-wise mutual information (PMI) values:

$$PMI(phr, f) = \log\left(\frac{P(phr, f)}{P(phr)P(f)}\right)$$

where *phr* is a phrase and f is a feature of *phr*. PMI effectively discounts the prior probability of the features and measures how much beyond random a feature tends to occur in a phrase's context window. Given two feature vectors, we compute the similarity between two vectors as the cosine function of the angle between the vectors. Note that even though a phrase *phr* can have multiple tokens, its feature *f* is always a single-word token.

We impose an upper limit on the number of instances of each phrase when constructing its feature vector. The idea is that if we have already seen 300K instances of a phrase, we should have already collected enough data for the phrase. More data for the same phrase will not necessarily tell us anything more about it. There are two benefits for such an upper limit. First, it drastically reduces the computational cost. Second, it reduces the variance in the sizes of the feature vectors of the phrases.

2.3 K-Means by MapReduce

K-Means is an embarrassingly parallelizable algorithm. Since the centroids of clusters are assumed to be constant within each iteration, the assignment of elements to clusters (Step ii) can be done totally independently.

The algorithm fits nicely into the MapReduce paradigm for parallel programming (Dean and Ghemawat, 2004). The most straightforward MapReduce implementation of K-Means would be to have mappers perform Step ii and reducers perform Step iii. The keys of intermediate pairs are cluster ids and the values are feature vectors of elements assigned to the corresponding cluster. When the number of elements to be clustered is very large, sorting the intermediate pairs in the shuffling stage can be costly. Furthermore, when summing up a large number of features vectors, numerical underflow becomes a potential problem.

A more efficient and numerically more stable method is to compute, for each input partition, the partial vector sums of the elements belonging to each cluster. When the whole partition is done, the mapper emits the cluster ids as keys and the partial vector sums as values. The reducers then aggregate the partial sums to compute the centroids.

2.4 Indexing centroid vectors

In a naïve implementation of Step ii of K-Means, one would compute the similarities between a feature vector and all the centroids in order to find the closest one. The kd-tree algorithm (Bentley 1980) aims at speeding up nearest neighbor search. However, it only works when the vectors are low-dimensional, which is not the case here. Fortunately, the high-dimensional and sparse nature of our feature vectors can also be exploited.

Since the cosine measure of two unit length vectors is simply their dot product, when searching for the closest centroid to an element, we only care about features in the centroids that are in common with the element. We therefore create an inverted index that maps a feature to the list of centroids having that feature. Given an input feature vector, we can iterate through all of its components and compute its dot product with all the centroids at the same time.

2.5 Sizes of context window

In our experiments, we use either 1 or 3 as the size of the context windows. Window size has an interesting effect on the types of clusters. With larger windows, the clusters tend to be more topical, whereas smaller windows result in categorical clusters.

For example, **Table 1** contains the cluster that the phrase "English lessons" belongs to. With 3-word context windows, the cluster is about language learning and translation. With 1-word context windows, the cluster contains different types of lessons.

The ability to produce both kinds of clusters turns out to be very useful. In different applications we need different types of clusters. For example, in the named entity recognition task, categorical clusters are more successful, whereas in query categorization, the topical clusters are much more beneficial.

The Brown algorithm uses essentially the same information as our 1-word window clusters. We therefore expect it to produce mostly categorical clusters.

2.6 Soft clustering

Although K-Means is generally described as a hard clustering algorithm (each element belongs to at most one cluster), it can produce soft clustering simply by assigning an element to all clusters whose similarity to the element is greater than a threshold. For natural language words and

Table 2 Soft clusters for *Whistler*

cluster1: sim=0.17, members=104048 bc vancouver, british columbia accommodations, coquitlam vancouver, squamish vancouver, langley vancouver, vancouver surrey, …
cluster2: sim=0. 16, members= 182692 vail skiing, skiing colorado, tahoe ski vacation, snowbird skiing, lake tahoe skiing, breckenridge skiing, snow ski packages, ski resort whistler, …
cluster3: sim=0.12, members= 91895 ski chalets france, ski chalet holidays, france ski, catered chalets, luxury ski chalets, france skiing, france skiing, ski chalet holidays, ……
cluster4: sim=0.11, members=237262 ocean kayaking, mountain hiking, horse trekking, river kayaking, mountain bike riding, white water canoeing, mountain trekking, sea kayaking, ……
cluster5: sim=0.10, members=540775 rent cabin, pet friendly cabin, cabins rental, cabin vacation, cabins colorado, cabin lake tahoe, maine cabin, tennessee mountain cabin, …
cluster6: sim=0.09, members=117365 mary cassatt, oil painting reproductions, henri matisse, pierre bonnard, edouard manet, auguste renoir, paintings famous, picasso paintings, ……
……

phrases, the soft cluster assignments often reveal different senses of a word. For example, the word *Whistler* may refer to a town in British Columbia, Canada, which is also a ski resort, or to a painter. These meanings are reflected in the top clusters assignments for *Whistler* in **Table 2** (window size = 3).

2.7 Clustering data sets

We experimented with two corpora (**Table 3**). One contains web documents with 700 billion tokens. The second consists of various news texts from LDC: English Gigaword, the Tipster corpus and Reuters RCV1. The last column lists the numbers of phrases we used when running the clustering with that corpus.

Even though our cloud computing infrastructure made phrase clustering possible, there is no question that it is still very time consuming. To create 3000 clusters among 20 million phrases using 3-word windows, each K-Means iteration takes about 20 minutes on 1000 CPUs. Without using the indexing technique in Section 2.4, each iteration takes about 4 times as long. In all our experiments, we set the maximum number of iterations to be 50.

Table 3 Corpora used in experiments

Corpus	Description	tokens	phrases
Web	web documents	700B	20M
LDC	News text from LDC	3.4B	700K

3 Named Entity Recognition

Named entity recognition (NER) is one of the first steps in many applications of information extraction, information retrieval, question answering and other applications of NLP. Conditional Random Fields (CRF) (Lafferty *et. al.* 2001) is one of the most competitive NER algorithms. We employed a linear chain CRF with L_2 regularization as the baseline algorithm to which we added phrase cluster features.

The CoNLL 2003 Shared Task (Tjong Kim Sang and Meulder 2003) offered a standard experimental platform for NER. The CoNLL data set consists of news articles from Reuters[1]. The training set has 203,621 tokens and the development and test set have 51,362 and 46,435 tokens, respectively. We adopted the same evaluation criteria as the CoNLL 2003 Shared Task.

To make the clusters more relevant to this domain, we adopted the following strategy:

1. Construct the feature vectors for 20 million phrases using the web data.
2. Run K-Means clustering on the phrases that appeared in the CoNLL training data to obtain K centroids.
3. Assign each of the 20 million phrases to the nearest centroid in the previous step.

3.1 Baseline features

The features in our baseline CRF classifier are a subset of the conventional features. They are defined with the following templates:

$$[y_s],\ [y_{s-1:s}],\ \{[y_s, w_u]\}_{u=s-1}^{s+1}, \{[y_{s-1:s}, w_u]\}_{u=s-1}^{s+1},$$
$$\{[y_s, sfx3_u]\}_{u=s-1}^{s+1},\ \{[y_{s-1:s}, sfx3_u]\}_{u=s-1}^{s+1},$$
$$\{\{[y_s, wtp_u^t]\}_{u=s-1}^{s+1}\}_{t=2}^{4},\ \{\{[y_{s-1:s}, wtp_u^t]\}_{u=s-1}^{s+1}\}_{t=2}^{4},$$
$$\{[y_s, w_{u-1:u}]\}_{u=s}^{s+1},\quad \{[y_{s-1:s}, w_{u-1:u}]\}_{u=s}^{s+1},$$
$$\{\{[y_s, wtp_{u-1:u}^t]\}_{u=s}^{s+1}\}_{t=1}^{3}, \{\{[y_{s-1:s}, wtp_{u-1:u}^t]\}_{u=s}^{s+1}\}_{t=1}^{3}$$

Here, s denotes a position in the input sequence; y_s is a label that indicates whether the token at position s is a named entity as well as its type; w_u is the word at position u; *sfx3* is a word's three-letter suffix; $\{wtp^t\}_{t=1}^{4}$ are indicators of

[1] http://www.reuters.com/researchandstandards/

different word types: *wtp*[1] is true when a word is punctuation; *wtp*[2] indicates whether a word is in lower case, upper case, or all-caps; *wtp*[3] is true when a token is a number; *wtp*[4] is true when a token is a hyphenated word with different capitalization before and after the hyphen.

NER systems often have global features to capture discourse-level regularities (Chieu and Ng 2003). For example, documents often have a full mention of an entity at the beginning and then refer to the entity in partial or abbreviated forms. To help in recognizing the shorter versions of the entities, we maintain a history of unigram word features. If a token is encountered again, the word unigram features of the previous instances are added as features for the current instance as well. We have a total of 48 feature templates. In comparison, there are 79 templates in (Suzuki and Isozaki, 2008).

Part-of-speech tags were used in the top-ranked systems in CoNLL 2003, as well as in many follow up studies that used the data set (Ando and Zhang 2005; Suzuki and Isozaki 2008). Our system does not need this information to achieve its peak performance. An important advantage of not needing a POS tagger as a preprocessor is that the system is much easier to adapt to other languages, since training a tagger often requires a larger amount of more extensively annotated data than the training data for NER.

3.2 Phrase cluster features

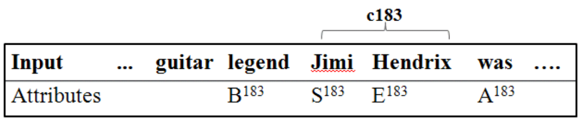
We used hard clustering with 1-word context windows for NER. For each input token sequence, we identify all sequences of tokens that are found in the phrase clusters. The phrases are allowed to overlap with or be nested in one another. If a phrase belonging to cluster *c* is found at positions *b* to *e* (inclusive), we add the following features to the CRF classifier:

$$[y_{b-1}, B^c], [y_{e+1}, A^c], [y_{b-2:b-1}, B^c], [y_{e:e+1}, A^c]$$

$$[y_b, S^c], \{[y_u, M^c]\}_{u=b+1}^{e-1}, [y_e, E^c]$$

$$[y_{b-1:b}, S^c], \{[y_{u-1:u}, M^c]\}_{u=b+1}^{e-1}, [y_{e-1:e}, E^c]$$

where *B* (before), *A* (after), *S* (start), *M* (middle), and *E* (end) denote a position in the input sequence relative to the phrase belonging to cluster *c*. We treat the cluster membership as binary. The similarity between an element and its cluster centroid is ignored. For example, suppose the input sentence is "… guitar legend Jimi Hendrix was …" and "Jimi Hendrix" belongs to cluster 183. **Figure 1** shows the attributes at different input positions. The cluster features are the cross product of the unigram/bigram labels and the attributes.



Input	**...**	**guitar**	**legend**	**Jimi**	**Hendrix**	**was**	**....**
Attributes			B^{183}	S^{183}	E^{183}	A^{183}	

Figure 1 Phrase cluster features

The phrasal cluster features not only help in resolving the ambiguities of words within a phrase, the *B* and *A* features also allow words adjacent to a phrase to consider longer contexts than a single word. Although one may argue longer n-grams can also capture this information, the sparseness of n-grams means that long n-gram features are rarely useful in practice.

We can easily use multiple clusterings in feature extraction. This allows us to side-step the matter of choosing the optimal value k in the K-Means clustering algorithm.

Even though the phrases include single token words, we create word clusters with the same clustering algorithm as well. The reason is that the phrase list, which comes from query logs, does not necessarily contain all the single token words in the documents. Furthermore, due to tokenization differences between the query logs and the documents, we systematically missed some words, such as hyphenated words. When creating the word clusters, we do not rely on a predefined list. Instead, any word above a minimum frequency threshold is included.

In their dependency parser with cluster-based features, Koo *et al.* (2008) found it helpful to restrict lexicalized features to only relatively frequent words. We did not observe a similar phenomenon with our CRF. We include all words as features and rely on the regularized CRF to select from them.

3.3 Evaluation results

Table 4 summarizes the evaluation results for our NER system and compares it with the two best results on the data set in the literature, as well the top-3 systems in CoNLL 2003. In this table, W and P refer to word and phrase clusters created with the web corpus. The superscripts are the numbers of clusters. LDC refers to the clusters created with the smaller LDC corpus and +pos indicates the use of part-of-speech tags as features.

The performance of our baseline system is rather mediocre because it has far fewer feature functions than the more competitive systems.

Table 4 CoNLL NER test set results

System	Test F1	Improv.
Baseline CRF (Sec. 3.1)	83.78	
W^{500}	88.34	+4.56
P^{64}	89.73	+5.94
P^{125}	89.80	+6.02
$W^{500} + P^{125}$	90.62	+6.84
$W^{500} + P^{64}$	90.63	+6.85
$W^{500} + P^{125} + P^{64}$	**90.90**	**+7.12**
$W^{500} + P^{125} + P^{64}$+pos	90.62	+6.84
LDC^{64}	87.24	+3.46
LDC^{125}	88.33	+4.55
LDC^{64} +LDC^{125}	88.44	+4.66
(Suzuki and Isozaki, 2008)	89.92	
(Ando and Zhang, 2005)	89.31	
(Florian *et al.*, 2003)	88.76	
(Chieu and Ng, 2003)	88.31	
(Klein *et al.*, 2003)	86.31	

The Top CoNLL 2003 systems all employed gazetteers or other types of specialized resources (e.g., lists of words that tend to co-occur with certain named entity types) in addition to part-of-speech tags.

Introducing the word clusters immediately brings the performance up to a very competitive level. Phrasal clusters obtained from the LDC corpus give the same level of improvement as word clusters from the web corpus that is 20 times larger. The best F-score of 90.90, which is about 1 point higher than the previous best result, is obtained with a combination of clusters. Adding POS tags to this configuration caused a small drop in F1.

4 Query Classification

We now look at the use of phrasal clusters in a very different application: query classification. The goal of query classification is to determine to which ones of a predefined set of classes a query belongs. Compared with documents, queries are much shorter and their categories are much more ambiguous.

4.1 KDDCUP 2005 data set

The task in the KDDCUP 2005 competition[2] is to classify 800,000 internet user search queries into 67 predefined topical categories. The training set consists of 111 example queries, each of which belongs to up to 5 of the 67 categories. **Table 5** shows three example queries and their classes.

Three independent human labelers classified 800 queries that were randomly selected from the complete set of 800,000. The participating systems were evaluated by their average F-scores (F1) and average precision (P) over these three sets of answer keys for the 800 selected queries.

Table 5 Example queries and their classes

ford field Sports/American Football Information/Local & Regional Sports/Schedules & Tickets
john deere gator Living/Landscaping & Gardening Living/Tools & Hardware Information/Companies & Industries Shopping/Stores & Products Shopping/Buying Guides & Researching
justin timberlake lyrics Entertainment/Music Information/Arts & Humanities Entertainment/Celebrities

$$P = \frac{\sum_i \text{\# queries correctly tagged as } c_i}{\sum_i \text{\# queries tagged as } c_i}$$

$$R = \frac{\sum_i \text{\# queries correctly tagged as } c_i}{\sum_i \text{\# of queries labeled by as } c_i}$$

$$F1 = \frac{2 \times P \times R}{P + R}$$

Here, 'tagged as' refer to systems outputs and 'labeled as' refer to human judgments. The subscript *i* ranges over all the query classes.

Table 6 shows the scores of each of the three human labelers when each of them is evaluated against the other two. It can be seen that the consistency among the labelers is quite low, indicating that the query classification task is very difficult even for humans.

To maximize the little information we have about the query classes, we treat the words in query class names as additional example queries. For example, we added three queries: *living*, *tools*, and *hardware* to the class Living\Tools & Hardware.

4.2 Baseline classifier

Since the query classes are not mutually exclusive, we treat the query classification task as 67 binary classification problems. For each query class, we train a logistic regression classifier (Vapnik 1999) with L_2 regularization.

Table 6 Labeler Consistency

	L1	L2	L3	Average
F1	0.538	0.477	0.512	0.509
P	0.501	0.613	0.463	0.526

[2] http://www.acm.org/sigs/sigkdd/kdd2005/kddcup.html

Given an input $\boldsymbol{x}$, represented as a vector of m features: (x_1, x_2, ..., x_m), a logistic regression classifier with parameter vector $\boldsymbol{w}$ =(w_1, w_2, ..., w_m) computes the posterior probability of the output y, which is either 1 or -1, as

$$p(y|\boldsymbol{x}) = \frac{1}{1 + e^{-y\boldsymbol{w}^T\cdot\boldsymbol{x}}}$$

We tag a query as belonging to a class if the probability of the class is among the highest 5 and is greater than 0.5.

The baseline system uses only the words in the queries as features (the bag-of-words representation), treating the query classification problem as a typical text categorization problem.

We found the prior distribution of the query classes to be extremely important. In fact, a system that always returns the top-5 most frequent classes has an F1 score of 26.55, which would have outperformed 2/3 of the 37 systems in the KDDCUP and ranked 13th.

We made a small modification to the objective function for logistic regression to take into account the prior distribution and to use 50% as a uniform decision boundary for all the classes. Normally, training a logistic regression classifier amounts to solving:

$$\arg min_{\boldsymbol{w}} \left\{ \lambda \boldsymbol{w}^T \boldsymbol{w} + \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + e^{-y_i \boldsymbol{w}^T \cdot \boldsymbol{x}_i}\right) \right\}$$

where n is the number of training examples and λ is the regularization constant. In this formula, $1/n$ can be viewed as the weight of an example in the training corpus. When training the classifier for a class with p positive examples out of a total of n examples, we change the objective function to:

$$\arg min_{\boldsymbol{w}} \left\{ \lambda \boldsymbol{w}^T \boldsymbol{w} + \frac{\sum_{i=1}^{n} \log\left(1 + e^{-y_i \boldsymbol{w}^T \cdot \boldsymbol{x}_i}\right)}{n + y_i(2p - n)} \right\}$$

With this modification, the total weight of the positive and negative examples become equal.

4.3 Phrasal clusters in query classification

Since topical information is much more relevant to query classification than categorical information, we use clusters created with 3-word context windows. Moreover, we use soft clustering instead of hard clustering. A phrase belongs to a cluster if the cluster's centroid is among the top-50 most similar centroids to the phrase (by cosine similarity), and the similarity is greater than 0.04.

Given a query, we first retrieve all its phrases (allowing overlap) and the clusters they belong to. For each of these clusters, we sum the cluster's similarity to all the phrases in the query and select the top-N as features for the logistic regression classifier (N=150 in our experiments). When we extract features from multiple clusterings, the selection of the top-N clusters is done separately for each clustering. Once a cluster is selected, its similarity values are ignored. Using the numerical feature values in our experiments always led to worse results. We suspect that such features make the optimization of the objective function much more difficult.

Table 7 Query Classification results

System	F1
bow	11.58
bow+W^{3K}	34.71
bow+P^{500}	39.84
bow+P^{3K}	40.80
bow+P^{500}+P^{1K} +P^{2K} +P^{3K}+P^{5K}	**43.80**

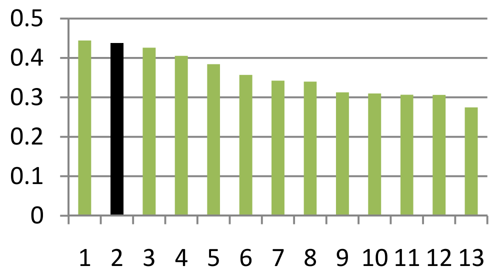


Figure 2 Comparison with KDDCUP systems

4.4 Evaluation results

Table 7 contains the evaluation results of various configurations of our system. Here, *bow* indicates the use of bag-of-words features; W^N refers to word clusters of size N; and P^N refers to phrase clusters of size N. All the clusters are soft clusters created with the web corpus using 3-word context windows.

The bag-of-words features alone have dismal performance. This is obviously due to the extreme paucity of training examples. In fact, only 12% of the words in the 800 test queries are found in the training examples. Using word clusters as features resulted in a big increase in F-score. The phrasal cluster features offer another big improvement. The best result is achieved with multiple phrasal clusterings.

Figure 2 compares the performance of our system (the dark bar at 2) with the top tercile systems in KDDCUP 2005. The best two systems in the competition (Shen *et al.*, 2005) and (Vogel *et al.*, 2005) resorted to knowledge engineering techniques to bridge the gap between the small set of examples and the new queries. They manually constructed a mapping from the query classes to hierarchical directories such as Google Directory[3] or Open Directory Project[4]. They then sent training and testing queries to internet search engines to retrieve the top pages in these directories. The positions of the result pages in the directory hierarchies as well as the words in the pages are used to classify the queries. With phrasal clusters, we can achieve top-level performance without manually constructed resources, or having to rely on internet search results.

5 Discussion and Related Work

In earlier work on semi-supervised learning, e.g., (Blum and Mitchell 1998), the classifiers learned from unlabeled data were used directly. Recent research shows that it is better to use whatever is learned from the unlabeled data as features in a discriminative classifier. This approach is taken by (Miller et. al. 2004), (Wong and Ng 2007), (Suzuki and Isozaki 2008), and (Koo et. al., 2008), as well as this paper.

Wong and Ng (2007) and Suzuki and Isozaki (2008) are similar in that they run a baseline discriminative classifier on unlabeled data to generate pseudo examples, which are then used to train a different type of classifier for the same problem. Wong and Ng (2007) made the assumption that each proper named belongs to one class (they observed that this is true about 85% of the time for English). Suzuki and Isozaki (2008), on the other hand, used the automatically labeled corpus to train HMMs.

Ando and Zhang (2005) defined an objective function that combines the original problem on the labeled data with a set of auxiliary problems on unlabeled data. The definition of an auxiliary problem can be quite flexible as long as it can be automatically labeled and shares some structural properties with the original problem. The combined objective function is then alternatingly optimized with the labeled and unlabeled data. This training regime puts pressure on the discriminative learner to exploit the structures uncovered from the unlabeled data.

In the two-stage cluster-based approaches such as ours, clustering is mostly decoupled from the supervised learning problem. However, one can rely on a discriminative classifier to establish the connection by assigning proper weights to the cluster features. One advantage of the two-stage approach is that the same clusterings may be used for different problems or different components of the same system. Another advantage is that it can be applied to a wider range of domains and problems. Although the method in (Suzuki and Isozaki 2008) is quite general, it is hard to see how it can be applied to the query classification problem.

Compared with Brown clustering, our algorithm for distributional clustering with distributed K-Means offers several benefits: (1) it is more scalable and parallelizable; (2) it has the ability to generate topical as well as categorical clusters for use in different applications; (3) it can create soft clustering as well as hard ones.

There are two main scenarios that motivate semi-supervised learning. One is to leverage a large amount of unsupervised data to train an adequate classifier with a small amount of labeled data. Another is to further boost the performance of a supervised classifier that is already trained with a large amount of supervised data. The named entity problem in Section 3 and the query classification problem in Section 4 exemplify the two scenarios.

One nagging issue with K-Means clustering is how to set k. We show that this question may not need to be answered because we can use clusterings with different k's at the same time and let the discriminative classifier cherry-pick the clusters at different granularities according to the supervised data. This technique has also been used with Brown clustering (Miller *et. al.* 2004, Koo, *et. al.* 2008). However, they require clusters to be strictly hierarchical, whereas we do not.

6 Conclusions

We presented a simple and scalable algorithm to cluster tens of millions of phrases and we used the resulting clusters as features in discriminative classifiers. We demonstrated the power and generality of this approach on two very different applications: named entity recognition and query classification. Our system achieved the best current result on the CoNLL NER data set. Our query categorization system is on par with the best system in KDDCUP 2005, which, unlike ours, involved a great deal of knowledge engineering effort.

Acknowledgments
The authors wish to thank the anonymous reviewers for their comments.

[3] http://directory.google.com
[4] http://www.dmoz.org

References

R. Ando and T. Zhang A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data. *Journal of Machine Learning Research*, Vol 6:1817-1853, 2005.

B.H. Bloom. 1970, Space/time trade-offs in hash coding with allowable errors, *Communications of the ACM* 13 (7): 422–426

A. Blum and T. Mitchell. 1998. Combining labeled and unlabeled data with co-training. Proceedings of the Eleventh Annual Conference on Computational Learning Theory pp. 92–100.

P.F. Brown, V.J. Della Pietra, P.V. de Souza, J.C. Lai, and R.L. Mercer. 1992. Class-based n-gram models of natural language. Computational Linguistics, 18(4):467–479.

H. L. Chieu and H. T. Ng. Named entity recognition with a maximum entropy approach. In Proceedings CoNLL-2003, pages 160–163, 2003.

J. Dean and S. Ghemawat. 2004. MapReduce: Simplified data processing on large clusters. In Proceedings of the Sixth Symposium on Operating System Design and Implementation (OSDI-04), San Francisco, CA, USA

S Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. A. Harshman. 1990. Indexing by latent semantic analysis, Journal of the American Society for Information Science, 1990, 41(6), 391-407

R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. Named entity recognition through classifier combination. In Proceedings CoNLL-2003, pages 168–171, 2003.

D. Klein, J. Smarr, H. Nguyen, and C. D. Manning. Named entity recognition with character-level models. In Proceedings CoNLL-2003, pages 188–191, 2003.

P. Koehn, F.J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In Proceedings of HLT-NAACL 2003, pp. 127–133.

T. Koo, X. Carreras, and M. Collins. Simple Semi-supervised Dependency Parsing. *Proceedings of ACL*, 2008.

J. Lafferty, A. McCallum, F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proc. 18th International Conf. on Machine Learning, Morgan Kaufmann, San Francisco, CA (2001) 282–289

Y. Li, Z. Zheng, and H.K. Dai, KDD Cup-2005 Report: Facing a Great Challenge. SIGKDD Explorations, 7 (2), 2005, 91-99.

D. Lin, S. Zhao, and B. Van Durme, and M. Pasca. 2008. Mining Parenthetical Translations from the Web by Word Alignment. Proc. of ACL-08. Columbus, OH.

J. Lin. Scalable Language Processing Algorithms for the Masses: A Case Study in Computing Word Co-occurrence Matrices with MapReduce. Proceedings of EMNLP 2008, pp. 419-428, Honolulu, Hawaii.

J. B. MacQueen (1967): Some Methods for classification and Analysis of Multivariate Observations, Proc. of 5-th Berkeley Symposium on Mathematical Statistics and Probability", Berkeley, University of California Press, 1:281-297

S. Miller, J. Guinness, and A. Zamanian. 2004. Name Tagging with Word Clusters and Discriminative Training. In *Proceedings of HLT-NAACL*, pages 337–342.

M. Sahami and T.D. Heilman. 2006. A web-based kernel function for measuring the similarity of short text snippets. Proceedings of the 15th international conference on World Wide Web, pp. 377–386.

D. Shen, R. Pan, J.T. Sun, J.J. Pan, K. Wu, J. Yin, Q. Yang. Q2C@UST: our winning solution to query classification in KDDCUP 2005. SIGKDD Explorations, 2005: 100~110.

J. Suzuki, and H. Isozaki. 2008. Semi-Supervised Sequential Labeling and Segmentation using Giga-word Scale Unlabeled Data. In Proc. of ACL/HLT-08. Columbus, Ohio. pp. 665-673.

E. T. Tjong Kim Sang and F. De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In Proc. of CoNLL-2003, pages 142–147.

Y. Wong and H. T. Ng, 2007. One Class per Named Entity: Exploiting Unlabeled Text for Named Entity Recognition. In Proc. of IJCAI-07, Hyderabad, India.

J. Uszkoreit and T. Brants. 2008. Distributed Word Clustering for Large Scale Class-Based Language Modeling in Machine Translation. Proceedings of ACL-08: HLT, pp. 755-762.

V. Vapnik, 1999. The Nature of Statistical Learning Theory, 2nd edition. Springer Verlag.

D. Vogel, S. Bickel, P. Haider, R. Schimpfky, P. Siemen, S. Bridges, T. Scheffer. Classifying Search Engine Queries Using the Web as Background Knowledge. SIGKDD Explorations 7(2): 117-122. 2005.