ClearSense:

Word Sense Disambiguation For Contextual Search

Project Report in Fulfillment of Requirements for the

Master's of Science Degree in Computer Science

Vadim von Brzeski

Department of Computer Science

University of California, Santa Cruz

December 1, 2005

Accepted by:

David Helmbold

Professor of Computer Science, UC Santa Cruz

Yi Zhang

Professor of Computer Science, UC Santa Cruz

ABSTRACT

This project presents *ClearSense*, a large scale, high performance, word sense disambiguation system. ClearSense uses the surrounding context around a target word to determine the sense or meaning of the word in that context. It can make use of a number of different statistical word sense disambiguation algorithms as plug-ins, and can itself be integrated as a component in a large-scale contextual search engine. In this paper we evaluate the system's accuracy and performance using two distinct models of text data: one, the well known Naive Bayes approach, and two, a more recent model of text corpora called Latent Dirichlet Allocation (LDA). The system is minimally supervised in the sense that the training contexts are not labeled, but are retrieved from the Web and from the Open Directory Project using seed phrases in search queries.

We present results for two scenarios. First, given a term, a context, and a desired target meaning, we aim to recognize whether or not the term has indeed the target meaning (binary decision). Second, we aim to determine the specific meaning of a term given a context (M-way decision).

Results based on a large editorial study comprising over 3,000 meaning judgments suggest that ClearSense works well in both scenarios and achieves an accuracy over 90% at a rate of 0.2ms per judgment. We present the design of a high performance and high accuracy word sense disambiguation system which requires a minimal amount of supervision and has the flexibility to support a number of different statistical models of text.

1. INTRODUCTION

Word sense disambiguation is an important component of contextual search. Contextual web search, specifically the Y!Q contextual search system (Kraft et al. [16]), aims to improve the relevancy of web search results by considering the information context of the user's current task along with the user's query. If context is to be taken into account during a search query for an ambiguous term (a term with more than one possible meaning or sense), it is imperative that the correct meaning of the term be accurately determined. For example, an effective search process for "Saturn" in the context "Click here to see a list of dealer showrooms" has to realize that the Saturn sought is the brand of car, not the planet. Therefore, the objective of this project was to build a system capable of performing this task.

Objective: Build and evaluate the effectiveness of a large scale, high performance word sense disambiguation system. The system should be able to accept a word and a piece of text containing the word (the context) as inputs, and return the most likely meaning of the word given the context. The desired use case is that the system can serve as the topic disambiguation component of the Y!Q contextual search system, although it should also be able to function as a stand-alone service. In the Y!Q use case, the system needs to service disambiguation requests for a large and growing set of ambiguous terms, and maintain an overall accuracy of over 90%.

In this project we present such a word sense disambiguation system, called *ClearSense*. At its core, ClearSense can accept a number of different *disambiguation service providers* (DSP) that accept a term, a context, and a list of possible meanings as input, and output the most likely meaning (from the list) of the term in the context. We integrate two such DSPs into ClearSense, and measure the accuracy (in terms of number of correct predictions) and performance (in terms of time per prediction) of each. Each DSP builds a separate probabilistic model for an ambiguous term, and can then service requests for word sense disambiguation.

A DSP builds its models in a minimally supervised fashion using documents obtained from the Web. First, seed phrases and templates for each term-meaning combination are manually constructed. Then web documents are automatically fetched from the Open Directory Project (ODP) [1] and the general Internet using the seed phrases and templates as search queries. No further *human* labeling, or verification that the documents retrieved actually correspond to the desired meaning of the term, is done. Instead the documents are automatically assigned labels corresponding to the queried term-meaning, i.e. we assume the query results are 100% accurate and noise-free (which of course is not the case). The (training) contexts used to construct the models consist of the complete (parsed and de-tagged HTML) text obtained from the documents, randomly sampled from the complete document database, ranging from 5 characters to 688K characters in length, with an average of 6K characters. The (test) contexts used during evaluation are much smaller pieces of text containing the term in question (ranging from 5 to 1.4K in length, with an average of approximately 400 characters).

The first DSP we applied is based on the standard Naive Bayes (Manning et al. [18]) bag-of-words model of text in which all words in a context are conditionally independent given the meaning. In this case, a model is built from simple word frequencies in the training corpus, given the assumed labels above. The second DSP we evaluated is based on a recently developed technique called Latent Dirichlet Allocation (LDA) developed by Blei et al. [7], which models a document (also a bag-of-words) as a mixture over an underlying, hidden (latent) number of topics.

We evaluate ClearSense with each DSP in two scenarios, each of which has two variants, for a total of four results per DSP. The first scenario involves a binary decision: is the meaning of the term in the given context the *target meaning* we seek? The second, more difficult, scenario involves an M-way decision: which of the meanings of the term is meant by the given context? The reason we are interested in the first scenario is that it is relevant in the Y!Q framework. The Y!Q contextual search engine has access to meta-data for a collection of terms as specific named entities, e.g. musicians, companies, etc. If we can determine that a context refers to a term as a specific named entity - called the *target meaning* - then Y!Q can leverage that meta-data to refine its search (see section 3.3).

Each of the test scenarios has two variants, depending on how the test contexts' meaning labels were assigned. The test contexts are also taken from the pool of documents retrieved from the Web (but different from the training set), and thus the true meanings of the terms in them are unknown. In the first variant, we again assume that the document retrieval process was 100% accurate and noise-free, and we label the test contexts with the meanings used in the search queries. In the second variant, we

asked an editorial group at Yahoo! to manually review and label the entire set of test contexts.

Our experiments show the following accuracy results: the Naive Bayes provider ranged from 88.6% (M-way decision, editorial label) to 95.4% (binary decision, assumed meaning label); the LDA provider ranged from and 86.5% (M-way decision, editorial meaning label) to 93.4% (binary decision, assumed meaning label). However, in terms of performance, Naive Bayes performed much better, at an average rate of 0.2 ms per prediction, compared to an average of 3.2 ms per prediction for LDA.

Our key contributions in this paper are as follows:

- A word sense disambiguation system which uses a minimal amount of labeled training data and still performs at par with fully supervised methods
- A system that demonstrates high accuracy and high performance in large scale experimental results on 100+ words, 300+ meanings, and 3,000+ judgments
- A system that can utilize a number of different statistical models of text as its core disambiguation engine
- A system that can be integrated with a major commercial contextual search engine, like Y!Q contextual search

The next section defines the terminology we will be using throughout this paper. The subsequent sections discuss related work concerning word sense disambiguation and contextual search, provide a brief overview of Naive Bayes and LDA, describe the architecture of the system, and explain the experimental setup and results. The paper concludes with a discussion of some of the challenges and proposals for future work.

2. TERMINOLOGY

This section defines the specific terminology we will be using in this paper.

- **word** : an item from a vocabulary indexed by {1..V}
- **term** : the ambiguous word or named entity whose meaning we are trying to discern
- **meaning** : the sense of the term in the given context
- **context** : a piece of text (e.g. a few words, a sentence, a paragraph, a complete document) that has been authored by someone
- **target meaning** : the specific meaning we are trying to identify for the purposes of Y!Q named entity matching; used in the binary classification setting
- **disambiguation service provider (DSP)** : a component of ClearSense which predicts the meaning of a term given a context and a list of possible meanings for the term
- **context meaning vector** : a dense, vector space representation of the meaning of a term in the context; used in the LDA setting
- **representative meaning vector** : the centroid (average) of the context meaning vectors of the training contexts for a given meaning of a term; used in the LDA setting
- **baseline meaning** : the most popular meaning of a term, regardless of context, as measured by our system
- **baseline percentage** : the relative frequency of the baseline meaning of a term as compared to the term's other meanings

3. BACKGROUND AND RELATED WORK

This section presents an overview of previous work in word sense disambiguation and how our work differs. We review related work in the area of contextual search, and briefly discuss the statistical models underlying the two disambiguation service providers, showing how each model is put to use in our system. We also briefly describe the Y!Q contextual search system and how it can benefit from ClearSense.

3.1 Word Sense Disambiguation

Related work in word sense disambiguation can be broken down into the following general categories: knowledge-based disambiguation, supervised disambiguation, minimally supervised disambiguation, and unsupervised disambiguation / discrimination (Mihalcea and Pedersen [25]).

Knowledge-based systems make use of external, often hand-crafted, lexical databases such as dictionaries, thesauri, and semantic networks, e.g. WordNet [3]. An example of an algorithm using such knowledge bases is the Lesk Algorithm (Lesk [17]). Lesk's initial results showed 50%-70% accuracy on short samples of text from a manually labeled set.

The use of a semantic network for word sense disambiguation requires the definition of a *semantic similarity metric* (Rada et al. [26]) over the semantic network, and this metric is typically based on the path and the distance between two terms in the network hierarchy. Using such a metric on the WordNet semantic network, Mihalcea et al. [21] report an average accuracy of 80% for the first ranked sense, and an average accuracy of 91% for the first two ranked senses of an ambiguous word.

Supervised methods for word sense disambiguation encompass a variety of machine learning techniques applied to manually labeled training data. For a good overview, we refer the reader to an excellent tutorial by Mihalcea and Pedersen [25]. To summarize, sense (meaning) labeled contexts are converted to feature vectors, and then a supervised learning algorithm (e.g. support vector machines, nearest neighbor, decision trees, etc.) builds a model from the labeled training set. For example, Pedersen [24] develops an ensemble of Naive Bayes classifiers to recognize the different meanings of the

words "interest" and "line", with six meanings per word, and reports an accuracy of 88% and 89% for the two words, respectively. Gale et al. [11] report accuracy results of about 90% when using Naive Bayes as described above for six nouns in the Hansard corpus.

Minimally supervised methods consist of techniques for building classifiers in situations where one would like to leverage a small, manually labeled training set to annotate a much larger unlabeled training set. These include bootstrapping, co-training, and self-training methods. Mihalcea [20] reports an average accuracy of 65.6% and 65.8% in self-training and co-training, respectively, for 29 SENSEVAL-2 nouns, some of which have up to 11 meanings. Yarowsky [29] describes a minimally supervised approach, in which a small seed set of training samples is labeled with meanings, and a bootstrapping technique is used to grow the seed set from unlabeled data. Yarowsky reports an average accuracy of 96.5% when disambiguating 10 nouns, each of which is limited to only two meanings.

Unsupervised methods operate without any manually labeled data, and rely on clustering algorithms to cluster related contexts together. These techniques are based on the Strong Contextual Hypotheses by Miller and Charles [23], which basically states that words with similar meanings tend to occur in similar contexts. Contexts are reduced to feature vectors, and a similarity metric is defined on the vector space. Clusters are created using this similarity metric, with the hope that each cluster represents one of the meanings of the term in question. Schutze [28] takes this approach to *word sense discrimination*, an easier problem where all one needs to do is to discriminate and group contexts by meaning, *not predict the actual meaning in the context.* In this discrimination setting, Schutze achieves average accuracy results of 83% to 91% on a test set of 20 terms which have only two major meanings.

Our approach falls somewhere in between the minimally supervised methods described above and the unsupervised discrimination method of Schutze [28]. We do not label the training data, but we do employ a set of seed phrases (which uniquely define the meaning), and use them in searching for documents on the Web and in ODP. The data is then automatically labeled according to the meanings contained in these seed phrases. This enables us, unlike Schutze, to perform full disambiguation (meaning prediction), not just discrimination. Schutze performs automatic unsupervised clustering to group related contexts together, and then measures the accuracy of those groupings. Since Schutze has no labels at his disposal, he cannot determine the particular meaning a given cluster represents. However, to be fair, Schutze's method is completely automated. Although we both use a vector space approach and map contexts and meanings to vectors (Schutze's "sense vectors" are our "representative meaning vectors"), we differ in how we construct our underlying vector space and our context vectors.

Compared to previous work, we also differ in the scale of our experiments. Schutze and Yarowsky performed experiments on 20 and 10 words, respectively, with two meanings each. We ran experiments on 108 words, where 54 words had 3 - 6 different meanings.

3.2 Contextual Search

The majority of contextual search work revolves around learning user profiles based on previous searches, search results, and web navigation patterns. The information system uses this learning to represent a user's interest for the refinement of future searches. Other efforts are focused on context learning (e.g. Goker [13], Belkin et al. [4]) based on judged relevant documents, query terms, document vectors, etc.

Context as a query, a different approach (e.g. Henzinger et al. [14], Rhodes et al. [27], Budzik et al. [8], Czerwinski et al. [9], Billsus et al. [5]), is to treat the context as a background for topic specific searches, and extract the query representing the context. Finkelstein et al. describe IntelliZap [10], a contextual search system, in which a query is the text that a user selects, and the surrounding context is then used to augment the query.

3.3 Y!Q Contextual Search

Y!Q (Kraft et al. [16]) is a large scale contextual search system integrated with a major web search engine (Yahoo! Search [1]). It captures search context for a term and uses the context to augment search queries to improve the overall relevancy of results. To accomplish this, Y!Q uses a content analysis (CA) engine based on a semantic network, a query planning and rewriting framework (QPW), and a contextual ranking engine (CR). The CA component is responsible for creating a weighted context term vector of the specified context and the optional query terms. QPW examines the term vector obtained from CA, determines the best places to search within a federated search environment, and constructs a rewritten query for each of the search targets.

One of steps in determining the best places to search is the disambiguation of the terms in the term vector. The terms in the context term vector are first cross-referenced using an *entity dictionary*, which contains a list of named entities in various categories, e.g. musicians, companies, etc. Then for each ambiguous term in the extracted list, a word sense disambiguation step is performed to determine the category (meaning) of the term given the context. It is in this step that Y!Q can take advantage

[1] http://search.yahoo.com

of ClearSense's functionality. Meta-data associated with the resolved named entities and categories is then used to steer the search process. The best search targets, along with the rewritten queries, are forwarded to the Y!Q front-end, which does a federated search using this information. The search results are finally processed and possibly re-ranked by the CR component based on a similarity distance measurement between a search result and the context term vector.

3.4 Naive Bayes

When applied to text classification, Naive Bayes is a simple model that posits that individual words (features) in a context are conditionally independent given the class (meaning) of that context. Specifically, the (log) probability that a context $\mathbf{c}$ (i.e. a bag of words w) has meaning m is given by:

$$\log P(m|\mathbf{c}) = \log P(m) + \sum_{w_i \in c} \log P(w_i|m)$$

$P(m)$ and $P(w_i|m)$ are empirically calculated from meaning and word frequencies in the training corpus and stored as the model for the term in question. The predicted meaning m for a new context $\mathbf{c}$ is the one corresponding to the largest value of $P(m|\mathbf{c})$. We evaluate the Mallet Naive Bayes Java library provided by McCallum [19].

3.5 Latent Dirichlet Allocation

Latent Dirichlet Allocation is a generative probabilistic model of text corpora. It is a natural extension of the probabilistic Latent Semantic Indexing (pLSI) model introduced by Hoffman [15]. LDA is similar to pLSI in that it is a hierarchical Bayesian model in which each document (treated as a bag of words) is a mixture of latent *topics*, where each topic (mixture component) is a multinomial random variable representing a distribution over words in a finite vocabulary. Thus each document can be reduced to a probability distribution over latent topics, and this probability distribution is represented by a vector of mixture proportions of these topics. However due to it being a true generative model, LDA resolves some of the perceived problems of pLSI, namely the inability of pLSI to assign probabilities to previously unseen documents (Girolami et al. [12]). For a complete description of LDA, we refer the reader to Blei et al. [7]. In summary, given a corpus consisting of *unlabeled* training documents, LDA computes the maximum-a-posteriori distribution of the latent topics in each document. We evaluate the LDA C library provided by Blei [6]. This library takes a variational inference approach

to estimating the posterior distribution of topics, and is a completely unsupervised process.

It is important to understand the relationship between topics in LDA and meanings as we use them. In the simple LDA model we apply, the number of latent topics k is a fixed quantity. Different choices of k yield different final models. One can think of k as the resolving power of the model to discriminate between different topics, sub-topics, sub-sub-topics, etc. Increases in k yield to increases in LDA's ability to group and distinguish documents by increasingly specific topics, where again topics are distributions over words. In our use of LDA we map topics to meanings, with multiple topics mapped to a single meaning. The reason for this is that we typically have two to six major meanings per term, and we are not interested in the possible sub-meanings (sub-topics). For example, for Boston, one of the major meanings is "the city in Massachusetts". A given training corpus for Boston the city could include contexts about Boston museums, police stations, etc., all of which could be recognized by LDA as natural sub-topics or sub-meanings. However, for the purposes of disambiguation we are not interested in those sub-topics - *all we care about is that we recognize that "Boston police stations" refers to Boston as the city and not the rock band.*

4. CLEARSENSE ARCHITECTURE

The ClearSense system architecture is shown in Figure 4.1. We describe the system components, the collection of training data, individual DSP model construction, and the process of disambiguating an ambiguous term in a new context. The major components of ClearSense are a Data Collection Engine (DC), a Disambiguation Service Provider (DSP) Interface, a relational SQL database, and a number of Disambiguation Service Providers.

4.1 Data Collection

Before ClearSense can service requests for a particular ambiguous term, it must first build a model for the term, and before it can build a model, it must collect training data for the term in question. This is the role of the DC engine.

The first step is the definition of the set of meanings a given term can have in our system. The meanings for a term are taken from Wordnet and Wikipedia [2], and the top two to six most common meanings for a term are chosen, e.g. for the term "amazon", the possible meanings in the system are: the online retailer, the river, the parrot, and the women warriors.

The next step is the collection of high quality training documents for each term-meaning pair. Specifically, for a given term with M meanings, we would like to obtain on the order of 20 - 30 documents (per meaning) which refer to the term in that and only that exact meaning. Our initial tests on a smaller scale indicated that this quantity produced good accuracy results. Furthermore, after the entire pool of documents was collected, we performed a complete analysis of the accuracy on test data as a function of the amount of training data (see section 5.5 for detailed discussion of this analysis).

However, even though the number of training contexts per meaning is relatively small, we need to process a large number of terms, and thus we cannot manually create a training set for each term-meaning pair; we need an automated means of collecting sample documents. Thus as a compromise between 100% noise-free data and full automation, we developed a semi-automated system in which

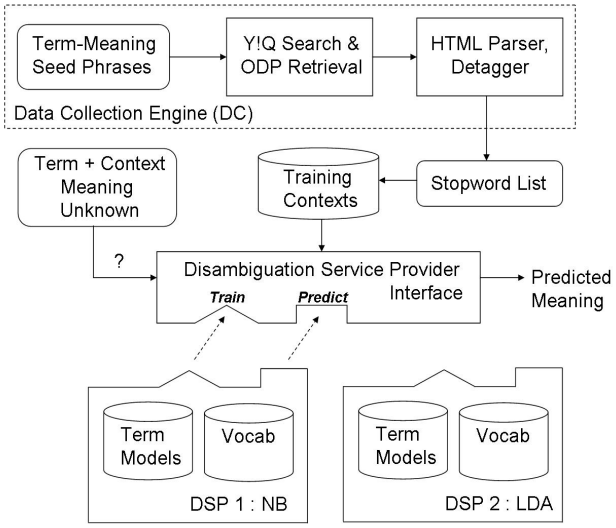

Fig. 4.1: ClearSense system architecture, showing how the Disambiguation Service Provider Interface is able to support different underlying statistical models.

we manually define seed phrases for each term-meaning pair, and then automatically fetch documents using search queries built from the terms and seed phrases. The documents are automatically retrieved in this fashion from two sources: the general Internet and the Open Directory Project. After being retrieved, each document's content is parsed and HTML tags are removed. This forms our entire training database (on the order of 100's of documents per term) from which training corpi (subsets) will be constructed. A training corpus for each term is a random selection of whole documents associated with this term, including all of its meanings. For a term with M meanings, we have a training corpus on the order of 20 - 30 * M large contexts per meaning (6K in size on average). All of the retrieved training data is stored in a SQL database for easy, reliable, and fast access.

Note that at this point we do not do any additional verification or labeling of the training data. Our use of the Web for document collection is similar to that of Mihalcea et al. [22], except that we do no further post-processing or manual checking. We assume that the search queries retrieved relevant documents for each term-meaning pair, and automatically label the documents as such. This naturally injects noise into our training data, and it is the price of automation.

4.2 Disambiguation Service Provider Interface

The DSP Interface provides access to different underlying DSP libraries, exposing a single interface to the user for model construction (training) and prediction. Each DSP by its nature is based on

a unique statistical model, and thus has uniquely different requirements for data processing during training and prediction. The DSP Interface hides these details from the user (e.g. a contextual search engine). A default disambiguation service provider can be set, or one can be selected by the user at run-time. A model is built (trained) for each individual ambiguous term, which can be done in parallel and independently of other terms. The built models are again stored in a SQL database for efficient retrieval. Once the model for a term has been built by a given DSP, ClearSense can service disambiguation requests from a contextual search engine, e.g. Y!Q, for that term.

4.3 Naive Bayes DSP

The Naive Bayes DSP provides a wrapper around the Mallet Naive Bayes Java library (McCallum [19]). We make straightforward use of Naive Bayes as described in the literature. The final model for each term is stored as an array of M prior (log) probabilities $P(m)$ for each of the M possible meanings, and a V x M meanings matrix, where V is the number of unique words in the training corpus, excluding stopwords. Each entry of this matrix is the (log) conditional probability of the word given the meaning $P(w_i|m)$, again derived empirically from word counts in the training corpus. The context instances in the corpus are automatically labeled in the Data Collection phase as described above. The prediction of a meaning of a term in a new context $\mathbf{c}$ is based on the Bayes Decision Rule (Manning et al. [18]) as follows:

$$m_{predicted} = \arg\max_{m}[\log P(m) + \sum_{w_i \in c} \log P(w_i|m)]$$

Words in test contexts that do not appear in the vocabulary are ignored in the prediction step.

4.4 Latent Dirichlet Allocation DSP

LDA requires that we first build an indexed vocabulary for each term from its training data set. An indexed vocabulary is built for each term from this corpus by assigning a unique index to each unique word appearing in the corpus, under the following conditions:

- Each word is a string of characters less than 51 characters long.
- Each word must only contain the characters a-z,A-Z.
- Each word must appear at least twice in the corpus. We ignore words that only occur once in

the entire corpus since these words are not useful for discriminating between different meanings. This heuristic has the effect of reducing the vocabulary size for each term by 30% on the average.

- Words occurring in a static stopword list are ignored. The stoplist, containing English stopwords, was obtained from the editorial group at Yahoo!.

During the construction of a model for a particular term, all of the training contexts for the term (i.e. for all of the meanings) are lumped into one unlabeled corpus. The training contexts for each term are then processed according to the term's vocabulary, resulting in a sparse vector representation of each context as a list of (index, count) tuples, where "count" is the number of times the word at this index occurs in the context. LDA takes this sparse vector representation of the contexts as input, and outputs a k dimensional vector for each training context as an estimate of the posterior distribution of latent topics in that context. This vector, whose components are mixture proportions of each latent topic, becomes our *context meaning vector*. It is important to note that up to now, no labeling of any kind has been done to the training data.

At this point however, the training contexts are grouped by the term-meaning labels automatically assigned to them during the data collection phase, and for each group of contexts a representative meaning vector $\boldsymbol{v^m}$ is calculated as the centroid of the context meaning vectors in that group. The M representative meaning vectors ($\boldsymbol{v^1}$, $\boldsymbol{v^2}$, ..., $\boldsymbol{v^M}$) are normalized to each have unit length, and are stored as the *meaning model* for the particular term.

Prediction of the meaning of a term given a previously unseen context proceeds in similar fashion. First, the vocabulary for the given term is loaded, and the context is converted into a sparse vector representation as before. Words in the new context that do not exist in the vocabulary are ignored. LDA takes the sparse vector as input, and based on the previously built model parameters for this term, outputs a context meaning vector. The context meaning vector is normalized to have unit length. The meaning model for the term is loaded, and a cosine similarity is calculated between the context meaning vector $\boldsymbol{v}$ and each representative meaning vector $\boldsymbol{v^m}$ as:

$$\boldsymbol{sim(v, v^m) = \sum_{i=0}^{k} v_i v_i^m}$$

since both vectors are of unit length. Cosine similarity was used for two reasons: one, it was simple to implement, and two, it yielded higher accuracy on test data compared to another measure of similarity, the Kullback-Leibler divergence; see section 5.6 for a detailed analysis and comparison between the

two similarity measures. The predicted meaning is defined as:

$$m_{predicted} = \arg\max_{m} sim(v, v^m)$$

Note that although we are currently using LDA as a dimensionality reduction mechanism in a minimally supervised environment, LDA can be extended to cluster similar contexts in an unsupervised manner as in Schutze [28], and it is for this exact reason that we have chosen to integrate and evaluate LDA as one of the DSPs in our system.

5. EXPERIMENTAL SETUP AND RESULTS

This section describes the experiments we performed to evaluate the accuracy and performance of our system using different DSPs and presents the results of those experiments.

5.1 Data Sets

We collected a random set of 108 terms from the Y!Q entity dictionary that were labeled as ambiguous by the editorial team at Yahoo!. These terms were taken from the following categories: musicians and bands (50%), companies (32%), and animals (18%). As described above, we collected two to six meanings per term, for a total of 308 meanings, averaging 2.85 meanings per term. Our entire pool of retrieved documents for this set of terms contained 18,370 documents, of which we randomly selected 6,932 for training and 3,183 for test (see section 5.5 for a discussion of how the size of the training corpus was determined). Recall that for training we used whole documents, whereas for test we used smaller contexts that contained the term in question and were 398 characters long on average, starting and ending on word boundaries. The reason for using small contexts in test is that this is the typical use case scenario in Y!Q contextual search, based on Y!Q query log analysis.

5.2 Test Context Labeling

In order to evaluate the accuracy of our system we needed to know the "true" meanings of the terms in the test contexts. Since the test contexts also came from the pool of documents retrieved from the web, we split each of our experiments into two variants:

- **"Assumed Labels"** : The test contexts are assigned the meanings that were in the original search query requests. Here we assume the search queries were 100% accurate and noise-free. All test contexts are therefore unambiguously labeled, and no additional, previously unseen meaning labels exist. This required no human intervention.
- **"Editorial Labels"** : The test contexts were given to the Yahoo! editorial team to label. The

team was provided a list of terms, contexts, and a set of possible choices for the meaning of the term in the context, and instructed to choose the correct meaning. The choices consisted of the meanings originally defined in the data collection phase, but also included two other choices: "cannot tell" and "other". "Cannot tell" meant that the reviewer could not discern a single unique meaning for the term from the context, either because it was unclear or because the context contained multiple possible meanings, e.g. a dictionary definition. "Other" meant that the reviewer could identify a single unique meaning, but it was not one of the choices provided.

For example, given the term **"queen"** and the context:

"be moved in a straight line vertically, horizontally, or diagonally, any number of unoccupied squares as shown to the left, thus combining the moves of the rook and bishop . As with most pieces, the ***queen*** *captures by occupying the square on which an enemy piece sits. Ordinarily the* ***queen*** *is slightly more powerful than a rook and a bishop, while slightly less powerful than two rooks. Because the* ***queen****"*

the editors were asked to choose between "music group", "deck of cards", "chess", "insect", "royalty", "other", and "cannot tell". The instructions specified that the context need not be the definition of the meaning, but only needs to reference the term in the chosen meaning, e.g. "Boston Art Galleries" is not the definition of the city of Boston, but it nevertheless refers to Boston the city, not the 70s rock band. The team was not given the original documents from which the contexts were extracted.

Out of the 3,183 test contexts, 167 cases were labeled "cannot tell" (5.2%), and since it was impossible for a human reviewer to discern the meaning, these cases were eliminated from consideration and our accuracy calculations. Out of 3,183 test contexts, 110 were labeled "other" (3.5%); these cases were counted in our accuracy calculations as prediction errors, even though we are being a bit hard on our algorithm in penalizing it for making mistakes on meanings it never knew existed.

5.3 Test Scenarios

For each of the test cases listed below, we report two sets of results based on the two sets of labelings described above.

- **Binary Decision** : In this case we are only interested in determining whether the term in the

given context references the *target meaning* or not. As mentioned above, this is pertinent to Y!Q. Y!Q has access to meta-data for terms in its entity dictionary, organized in specific categories (e.g. music, company, etc.), and if we can determine that a context refers to the term in that category, then we can leverage that meta-data to improve search results. In this scenario, the models for each DSP were built using only two labels : one representing the target meaning and one representing all other meanings. All contexts in the training corpus that had automatic labels not matching the target meaning were lumped together under the label "other". The rest of the steps are as described above. For each of the 108 terms, the target meaning was chosen as the named entity referenced by the terms, not the generic meanings of the terms, e.g. Cranberries the rock band, not cranberries the fruit.

- **M-way Decision** : We were also interested in how well each technique performs in the more stringent case where we need to predict which of the M meanings is the true meaning of the term in the context. The steps involved here are exactly as already described above.

In addition to the accuracy evaluations above, we also performed several tests to validate our choices of training corpus size and cosine distance as the similarity measure. Finally we experimented with a "mixture of experts" strategy where we would make a meaning judgment only in cases where Naive Bayes and LDA agreed.

5.4 Results

5.4.1 Accuracy

We summarize our experimental results in Table 5.1. Accuracy is defined as the number of correct meaning judgments divided by the total number of judged contexts. The results show good performance of the system, even in the toughest scenario, the M-way decision with editorial labels. It is interesting to note that Naive Bayes has a higher accuracy than LDA in all four cases, outperforming LDA by approximately 1-2%, including the case of immediate interest to us, Binary / Editorial.

Since k, the number of latent topics in LDA, is the only parameter whose optimal value is not output by the LDA basic model, we ran all tests using eight different values of k, from 5 to 40 in increments of 5. We plot the accuracy results for the four evaluations (two test cases, each with two test labels) in Figure 5.1 as a function of k. We make the simplifying assumption that a single k value is optimal for all terms, which may not be the case. We define the optimal value k of as the

Test Scenario	Test Labeling	Total Contexts	Naive Bayes	LDA	LDA Optimal k
Binary	Assumed	3183	95.4%	93.4%	30
Binary	Editorial	3016	94.1%	92.6%	30
M-way	Assumed	3183	91.4%	89.2%	35
M-way	Editorial	3016	88.6%	86.5%	20

Tab. 5.1: Average accuracy results for the different DSPs in two test cases, each under two different test context labeling methods. Accuracy is defined as the number of correct meaning judgments divided by the total number of contexts. The number of total contexts differs in the Editorial and Assumed scenarios because "cannot tell" cases where not considered in the Editorial scenario.

Meaning Count	Number of Terms	Naive Bayes	LDA (k = 20)
2	54	93.4%	91.6%
3	27	88.7%	87.5%
4	19	88.1%	85.3%
5	5	82.2%	79.0%
6	3	71.3%	68.5%

Tab. 5.2: Average accuracy as a function of the number of meanings for a term in the M-way / Editorial scenario.

value yielding the maximum accuracy on the test data, and report this value in Table 5.1 for each test scenario.

Furthermore, as Table 5.2 shows, the accuracy remains quite good even for terms with up to four meanings, and definitely much better than random guessing. To calculate our margin of error in the accuracy estimates, we calculate the 95% confidence intervals for two of the cases. Table 5.3 shows that our margin of error ranges from 0.9% to 1.3% at the 95% confidence level for two of the cases.

5.4.2 Accuracy Baseline

To judge the significance of our results we compare them to a baseline. The baseline for all of our scenarios is a naive algorithm which always selects the most common and most popular meaning, the *baseline meaning*, for an ambiguous term. For example, if a term A references meaning B 80% of the time, then the baseline algorithm will always interpret term A as meaning B, and thus it will have an accuracy of 80%. Therefore we need to determine the frequency of the most common meaning for each term.

We take advantage of our own trained algorithm for this purpose. For each term, we first perform a regular web query using that single term alone, and we collect the first 1,000 results. From each of the results, we fetch just the summaries (snippets) as shown by the web search engine. Each of these summaries forms a context, which we classify using our system. We then compute the proportions of

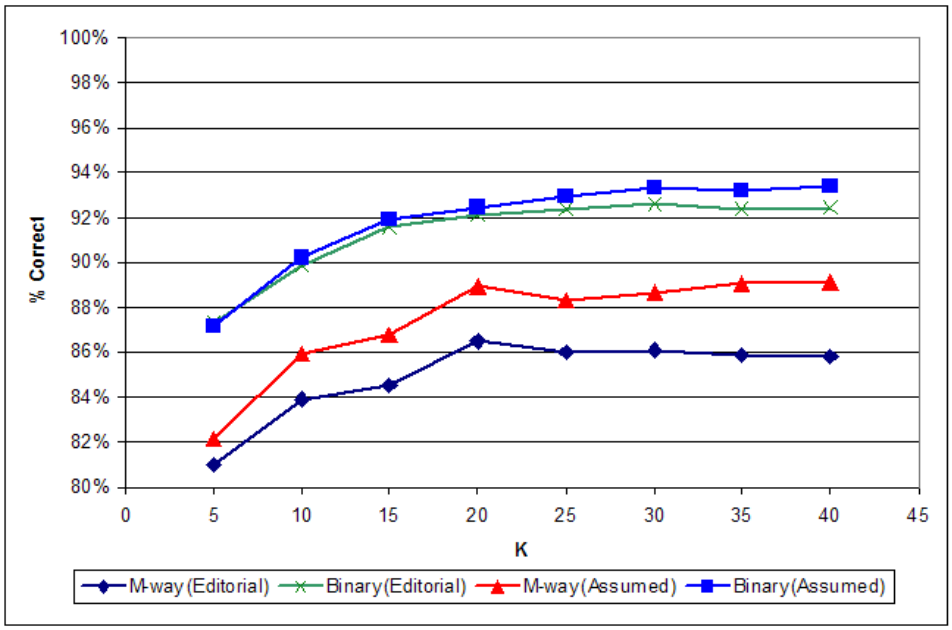

Fig. 5.1: LDA accuracy as a function of k. As one can see, initial increases in k improve accuracy significantly, but soon reach a point of diminishing returns.

Test Scenario	**Test Labeling**	**Naive Bayes**	**LDA**
M-way	Assumed	[90.3%, 92.3%]	[88.0%, 90.2%]
M-way	Editorial	[87.4%, 89.7%]	[85.2%, 87.7%]

Tab. 5.3: 95% confidence intervals for the accuracy results in the M-way scenario.

each of the resulting meanings. Since the results from a search engine are biased because of ranking, the distributions may be quite skewed, and may not reflect all possible meanings for a term. We are really measuring the distribution of search results, and we are assuming that our system is an accurate classifier. Nevertheless, as the table in Appendix A shows, the baseline results map quite well to popular perceptions about the most common meanings of the given terms. We compare our results to this baseline in Table 5.4, which shows the average baseline and improvement over the baseline as a function of the number of meanings per term. The average baseline over all terms was quite high, at 60.8%; our average improvement (over all terms) over the baseline is 29.1% using Naive Bayes and 27.1% using LDA in the M-way / Editorial scenario.

5.4.3 Performance

Since training and model construction can be done offline, we are interested in how long each DSP takes to make a prediction in real-time. We measured performance as the time in milliseconds required

Meaning Count	Average Baseline	Improvement LDA	Improvement NB
2	70.2%	21.4%	23.2%
3	55.1%	32.4%	33.6%
4	50.0%	35.3%	38.1%
5	44.0%	35.0%	38.1%
6	38.4%	30.1%	32.9%

Tab. 5.4: Improvement over baseline in the M-way / Editorial scenario as a function of the number of meanings for a term.

to compute a prediction. Due to Naive Bayes's simpler inference process (N table lookups plus N additions per possible meaning, where N is the number of words in the context) it is a clear winner in performance, averaging around 0.2 ms per prediction. LDA needs to run a variational (Expectation-Maximization like) inference algorithm for each context, which is also linear in k, and runs at 3.2 ms per prediction for k = 20 and 6.5 ms per prediction for k = 40.

5.4.4 Selection of DSP

Accuracy is of paramount importance in our use case. Given that Naive Bayes achieves higher accuracy in the Binary / Editorial scenario, and given that its prediction performance (less than 1 ms per prediction) is quite good in an absolute sense, it it our current DSP of choice for making binary predictions, as in the projected Y!Q use case. However, Y!Q may also take advantage of M-way predictions with a different call to ClearSense. ClearSense can also easily integrate and employ other DSPs in the future if their accuracy proves to be better.

5.5 Accuracy vs. Training Corpus Size

Figure 5.2 shows the accuracy on test data in the M-way scenarios as a function of training corpus size for both the Naive Bayes and LDA models. The five data points represent an average of 6, 12, 18, 24, and 33 training contexts per meaning (the average is over all terms). As the chart shows, initially there are significant gains in accuracy, but then the gains flatten out or even decrease. Maximum accuracy on the test set using Naive Bayes is achieved at 24 meanings per term, and since Naive Bayes achieves better overall accuracy compared to LDA, this is the size of the training data set that we selected for all of our experiments.

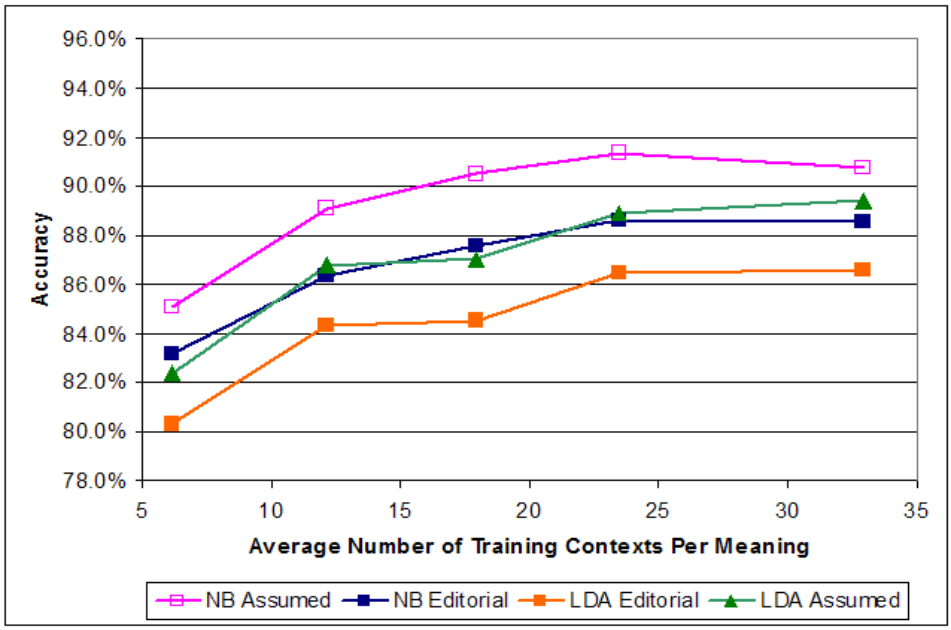

Fig. 5.2: Accuracy on test data (M-way decision) as a function of the average number of training samples (contexts) per meaning.

5.6 Choice of Similarity Measure

As part of the LDA evaluation, we compared two distinct similarity measures: cosine similarity and Kullback-Leibler divergence. As mentioned previously, cosine similarity between two vectors is defined as

$$\boldsymbol{sim(u,v) = \sum_{i=0}^{k} u_i v_i}$$

Kullback-Leibler divergence between two discrete distributions is defined as

$$\boldsymbol{KL(p,q) = \sum_{i=0}^{k} p_i \log \frac{p_i}{q_i}}$$

The Kullback-Leibler divergence, although not a true metric, is the expected amount of information that a sample from p gives about the fact that it is not from the distribution q. In our implementation, the p vectors correspond to distributions representing the test contexts, and the q vectors correspond to the distributions defined by the representative meaning vectors. Thus for each test context distribution p, we calculate how divergent it is from each of the representative meaning distributions q, and choose the representative meaning for which this divergence is *minimized*. In our experiments, none of the q vectors had a component exactly equal to zero.

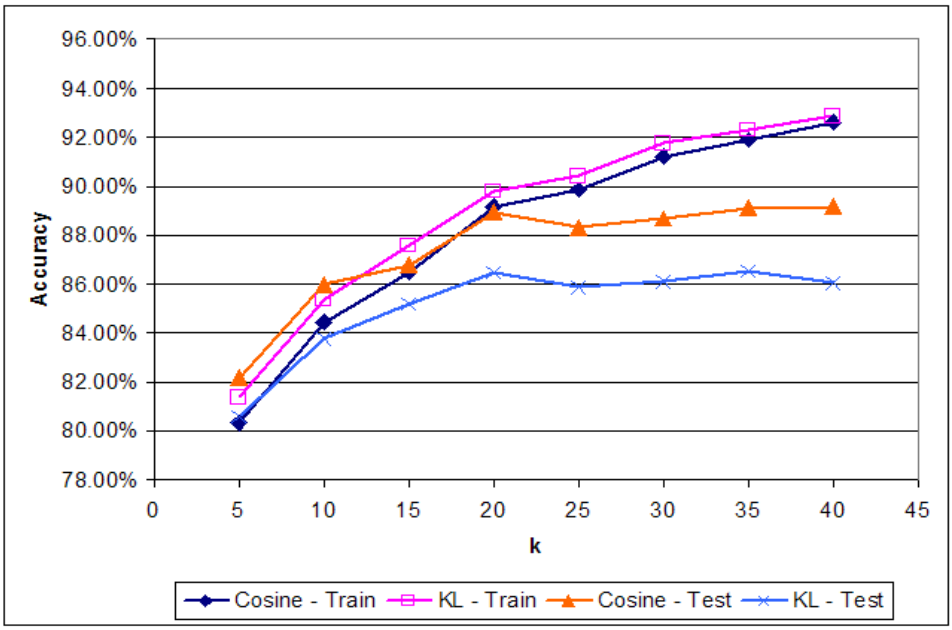

Fig. 5.3: Accuracy on training and test data sets using LDA with two different similarity measures: cosine similarity and Kullback-Leibler divergence (M-way / Assumed scenario).

Fugure 5.3 shows the accuracy results on training and test data sets using the two similarity measures. It is interesting to note that Kullback-Leibler divergence yielded better results on training data by approximately 1%, whereas cosine similarity performed better on test data by approximately 2%. We hypothesize that this discrepancy is due to the difference in size (number of words per context) between the training and test contexts. As one may recall, the training contexts were 6K in length on average, whereas the test contexts were 0.4K in length on average.

5.7 *Mixture of Experts*

Although our achieved accuracy was in excess of 90%, it is interesting to consider how we made predictions. Specifically, since when using both DSPs we made predictions based on a *maximum* value, we never failed to make a prediction, right or wrong. This may not be the optimal behavior since if one is unsure of the meaning of a term, it may be better not tag it with meta-data at all than to tag it with completely wrong meta-data and steer the search engine off course.

Therefore we experimented with a simple voting strategy in which we would only make a meaning prediction if a majority of disambiguation services providers agree. In this case we would make a prediction only if both Naive Bayes and LDA agreed. The results of this experiment are shown in

Test Scenario	No. of Times DSPs Disagree	NB Errors	LDA Errors	Accuracy on Remaining
M-way / Assumed	283	120	197	94.7%
M-way / Editorial	247	112	175	91.6%

Tab. 5.5: Accuracy on test contexts for which Naive Bayes and LDA make the same prediction. The number of times Naive Bayes and LDA disagree, and the number of errors in cases of disagreement are also shown under two different test scenarios. Since we eliminate cases where Naive Bayes and LDA disagree, the accuracy results for both DSPs will be identical, and thus we show only one accuracy number.

Table 5.5 for the M-way Editorial and Assumed scenarios.

Although the accuracy on the judgments ClearSense does make has improved by about 3% compared to using Naive Bayes alone (see Table 5.1), if ClearSense refrains from making judgments in certain cases where one of its DSPs could be correct, it loses something. Thus it is appropriate to ask if this voting strategy makes sense compared to using the best DSP (Naive Bayes) alone. To do this we introduce the following utility function:

$$\boldsymbol{U = N_c - C * N_w}$$

where $\boldsymbol{N_c}$ is the number of correct predictions and $\boldsymbol{N_w}$ is the number of incorrect predictions. We assume that making a correct prediction gives us +1 unit of utility, and the coefficient $\boldsymbol{C}$ specifies the negative utility we incur as a result of making an incorrect prediction, i.e. the ratio of negative penalty to positive reward. Figure 5.4 shows the utility curves as a function of $\boldsymbol{C}$ for the two different modes of operation, evaluated in the M-way / Editorial scenario. In one mode, we use the Naive Bayes DSP by itself; in the second mode, we use both Naive Bayes and LDA in a voting scheme. As the chart shows, the crossover value of $\boldsymbol{C}$ is 1.2. Thus, if the penalty for making an error is greater than 1.2 times the reward for a correct prediction, then ClearSense's voting scheme is more useful. Otherwise, using Naive Bayes by itself is more beneficial. The value of $\boldsymbol{C}$ will be determined by Editorial review in future work. Although we do not know the exact value of $\boldsymbol{C}$ at the moment, this analysis provides a clear way of making a decision (and customizing ClearSense) once we have ascertained the exact value of $\boldsymbol{C}$.

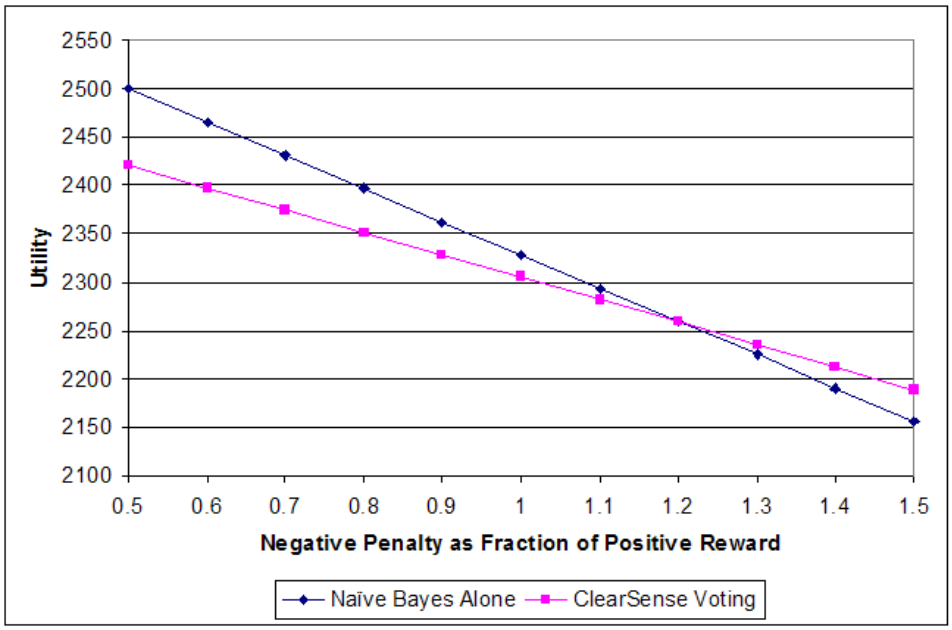

Fig. 5.4: Utility as a function of the ratio of negative penalty to positive reward.

6. CONCLUSION AND DISCUSSION

In conclusion, we have built and demonstrated the effectiveness of a system for word sense disambiguation. ClearSense can service disambiguation requests for terms from a contextual search framework, e.g. Y!Q. ClearSense has the flexibility of employing various statistical modeling algorithms as its disambiguation service providers, at the same time providing a simple interface to users. It is a large scale system, able to learn many terms and meanings from a minimally supervised training set. ClearSense demonstrates over 90% prediction accuracy without any manual verification or inspection of its training set. It is also a high performance system, able to make a meaning judgment in less than 1 millisecond for terms with many meanings. For example, for the term "queen", which has five possible meanings as described above, ClearSense performs very well. In the M-way / Editorial scenario, it accurately determined the correct meaning of "queen" 91.0% of the time, out of a total of 56 judgments.

However, the fact the we do no verification of the training data but assume the labels correspond exactly to the meanings in the seed phrases leads to some interesting consequences. For example, one of terms, "apollo" had the following meanings: a city in the US, a Greek god, a company that makes bathing equipment for hospitals, the NASA space missions, a movie by that name. In the M-way test using LDA, ClearSense predicted the "company" meaning for 13 test contexts. However, the editorial team labeled none of the test contexts for apollo as referring to this specific meaning: Apollo Corporation, the company making bathing equipment for hospitals. The editorial team did however label 12 apollo contexts as "other". Inspection of these 12 "other" contexts showed that they all referred to a chain of hospitals in India called Apollo Hospitals. Thus our noisy training data had led ClearSense completely off course, and it learned the "Indian chain of hospitals" meaning without our knowledge. When tested with the editorial labeling it was penalized for no fault of its own. In fact, when using the "Assumed" test context labels, ClearSense accurately determined the "correct" meaning 12 out of 12 times.

Another interesting case has to do with the term "amazon", either meaning the online retailer (Amazon.com), the river, the parrot, or the women warriors. It had an accuracy of 56.8% in the

M-way / Editorial scenario using both Naive Bayes and LDA, so we decided to investigate further. Since Amazon.com sells books on practically any subject, including the parrots, the river, and the women warriors, much of the training and test data for the term "amazon" came from pages from the Amazon.com website. An inspection of the test contexts for "amazon" revealed that out of the 33 contexts which we assumed referenced the river, the parrot, or the women warriors, 13 (40%) actually came from a page on the Amazon.com website. Since training contexts are sampled from the same pool as test contexts, it is difficult for the system to learn what it means for a context to refer to the *company* versus *something the company sells that has the same name.*

To give the reader another perspective on the results, we note that human review of 3,183 contexts identified 277 (8.7%) of them as either ambiguous or referencing a completely different meaning from the one that was requested in the data collection phase. This gives us some indication of how well we can expect an ideal algorithm to perform. Specifically, given 8.7% of noise in the data, the most we should expect from our system is 91.3% accuracy in the M-way / Editorial scenario. As it turns out, *ClearSense came within 2.7% of this ideal goal* using Naive Bayes. Further analysis also revealed that in the M-way / Editorial scenario using LDA, only 12 terms (11%), averaging four meanings each, were responsible for 40% of the errors. Given this skewed distribution of errors, we surmise that by improving the training data collection process for particular terms we can significantly improve overall results.

Perhaps it is not surprising that Naive Bayes performed at par with a more sophisticated model. There are numerous examples of just this fact in the literature. Naive Bayes has been compared to neural networks, decision trees, disjunctive and conjunctive normal form learners, and rule based learners. In all cases, "Naive Bayes performed as well as any of the other methods" (Mihalcea et al. [25]). We can now add LDA to this list as well.

7. FUTURE WORK

Even though ClearSense currently does not require hand labeling of data, it does require manual meaning collection and seed phrase construction. However, the advantage of our system, compared to a hand-crafted knowledge base like a thesaurus, is that it can easily be extended to be fully automated. Our future work will focus on this automation, allowing the system to run completely unsupervised. It will include the ability to learn new terms on-the-fly by automatically building a *meaning universe* - a collection of all possible meanings - for each term, along with a set of seed phrases. Nevertheless, our goal will still be the collection of very high quality training data, albeit fully unsupervised. Once the fully automated system is complete, we will run an end-to-end evaluation of ClearSense as part of the Y!Q contextual search engine. This will involve another evaluation by the Editorial team, but this time comparing relevancy of search results of Y!Q, with and without ClearSense.

Furthermore, given that ClearSense can integrate a number of various DSPs, we will also investigate a few more disambiguation techniques such as logistic regression and boosting. Once we have three or four reliable DSPs at our disposal, we will again look at a voting strategy to see if we can improve overall utility of the system.

Another interesting direction to pursue involves improving the quality of the training data. As an initial start in this direction we decided to run a quick experiment to see if we could eliminate some of the noisy training contexts, and measure the accuracy impact on the test set. We ran the Naive Bayes and LDA prediction algorithms on the training data set, and tagged traning contexts where the two disagreed as (potentially) "noisy". There were 768 such contexts, out of a total of 6,932. We then re-trained (from scratch) using only the non-noisy training contexts. However, the test set accuracy results (M-way / Assumed scenario) were not encouraging: Naive Bayes dropped 0.6% and LDA dropped 0.3%. This is most likely due to the fact that along with "noisy" training data, we also removed good training data, and the effect of this can be seen in Figure 5.2. More work needs to be done in this area since data quality will become more important in the case where meaning universes and training corpi will be built without supervision.

8. ACKNOWLEDGMENTS

We would like to thank Reiner Kraft, David Helmbold, Yi Zhang, Roy Shan, Farzin Maghoul, and Arkady Borkovsky for their helpful comments and discussions, and to Yahoo!, Inc. for making this research possible. Furthermore, we are especially grateful to Teresa Block and the Yahoo! editorial team for their assistance in the evaluation process.

BIBLIOGRAPHY

[1] Open Directory Project. http://www.dmoz.org.

[2] Wikipedia, the free encyclopedia. http://en.wikipedia.org.

[3] WordNet, an online lexical reference system. Princeton University, Cognitive Science Laboratory. http://wordnet.princeton.edu.

[4] N. J. Belkin, R. Oddy, and H. M. Brooks. ASK for Information Retrieval: Part I. Background and Theory, Part II. Results of a Design Study. *Journal of Documentation*, 38(3):61–71, 145–164, Sep. 1982.

[5] D. Billsus, D. Hilbert, and D. Maynes-Aminzade. Improving proactive information systems. In *International Conference on Intelligent User Interfaces (IUI 2005)*, January 2005.

[6] D. Blei. LDA C library. http://www.cs.berkeley.edu/∼blei/lda-c/.

[7] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research 3*, 2003.

[8] J. Budzik and K. Hammond. Watson: Anticipating and contextualizing information needs. In *62nd Annual Meeting of the American Society for Information Science*, Medford, NJ, 1999.

[9] M. Czerwinski, S. Dumais, G. Robertson, S. Dziadosz, S. Tiernan, and M. van Dantzich. Visualizing implicit queries for information management and retrieval. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 560–567, New York, NY, USA, 1999. ACM Press.

[10] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. Placing search in context: the concept revisited. *ACM Trans. Inf. Syst.*, 20(1):116–131, 2002.

[11] C. K. Gale W. and Y. D. A method for disambiguating word senses in a large corpus. *Computers and the Humanities*, (26), 1993.

[12] M. Girolami and A. Kabn. On an equivalence between pLSI and LDA. pages 433–434.

[13] A. Goker. Capturing information need by learning user context. In *Sixteenth International Joint Conference in Artificial Intelligence: Learning About Users Workshop*, pages 21–27, 1999.

[14] M. Henzinger, B.-W. Chang, B. Milch, and S. Brin. Query-free news search. In *Twelfth international World Wide Web Conference (WWW-2003), Budapest, Hungary*, May 20-24 2003.

[15] T. Hofmann. Probabilistic Latent Semantic Analysis. In *Proc. of Uncertainty in Artificial Intelligence, UAI'99*, Stockholm, 1999.

[16] R. Kraft, C. C. Chang, and F. Maghoul. Y!Q: Contextual search at the point of inspiration. In *CIKM '05, 14th Conference on Information and Knowledge Management, Bremen, Germany*, November 2005.

[17] M. Lesk. Automatic sense disambiguation using machine readable dictionaries : How to tell a pine cone from an ice cream cone. *SIGDOC*, 1986.

[18] C. D. Manning and H. Schutze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.

[19] A. K. McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[20] R. Mihalcea. Co-training and self-training for word sense disambiguation. In H. T. Ng and E. Riloff, editors, *HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, pages 33–40, Boston, Massachusetts, USA, May 6 - May 7 2004. Association for Computational Linguistics.

[21] R. Mihalcea and D. Moldovan. A method for word sense disambiguation of unrestricted text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99) (Maryland, NY, June 1999).*

[22] R. Mihalcea and D. Moldovan. An automatic method for generating sense tagged corpora. In *AAAI/IAAI*, pages 461–466, 1999.

[23] G. Miller and W. Charles. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28, 1991.

[24] T. Pedersen. A simple approach to building ensembles of naive bayesian classifiers for word sense disambiguation. In *Proceedings of the First Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 63–69, Seattle, WA, May 2000.

[25] T. Pedersen and R. Mihalcea. Advances in word sense dismabiguation - Tutorial at AAAI-2005. http://www.aclweb.org/acl2005/index.php?tutorial-mp.

[26] E. B. R. Rada, H. Mili and M. Blettner. Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man and Cybernetics*, 19(1):17–30, 1989.

[27] B. Rhodes and T. Starner. The remembrance agent: A continuously running automated information retrieval system. In *The Proceedings of The First International Conference on The Practical Application of Intelligent Agents and Multi Agent Technology (PAAM '96), London, UK*, pages 487–495, April 1996.

[28] H. Schutze. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123, 1998.

[29] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Meeting of the Association for Computational Linguistics*, pages 189–196, 1995.

APPENDIX

A. BASELINES BY TERM

Figure A.1 shows the terms that were used in the evaluations as well as estimates of their baseline meaning frequencies as described in section 5.4.2.

Term	No. of Meanings	Baseline Meaning	Baseline %
acadia	4	natlpark	51.9%
act	6	product	45.5%
adobe	2	company	89.1%
alabama	3	state	61.3%
alamo	4	place	55.5%
alaska	4	state	72.7%
amazon	4	company	58.7%
american	2	ethnicity	69.0%
angora	4	goat	56.8%
anthrax	2	disease	95.6%
apache	5	wwwserver	70.9%
apollo	5	space_missions	30.3%
apple	3	company	62.8%
armstrong	6	person_biker	37.9%
arrow	3	company	36.1%
asia	2	continent	76.8%
autonomy	2	company	71.9%
b-52	2	airplane	51.9%
balinese	2	bali	70.6%
bell	4	alexander	45.7%
berlin	3	eucity	62.9%
bombay	2	indiacity	70.0%
boston	3	uscity	52.0%
boxer	4	dog	53.9%
brittany	4	dog	29.6%
burmese	2	burma	56.1%
cake	2	food	76.9%
carpenters	4	band	55.5%
caterpillar	4	company	52.3%
chase	5	bank	31.6%
cherokee	3	native_american	57.4%
chicago	2	uscity	76.1%
cinderella	2	story	82.9%
citizen	2	person	72.5%
coach	4	sports_teacher	61.8%
colt	6	firearm_inventor	31.9%
cranberries	2	fruit	63.3%
credence	3	p2pobject	55.8%
cure	2	medicine	56.0%
doors	2	entry	72.9%
eagles	3	nfl	46.0%
europe	2	continent	60.5%
foreigner	2	band	90.6%
fossil	3	animal	46.3%
fox	3	company	57.4%
fuji	2	company	56.5%
gap	3	company	45.8%
garbage	2	trash	66.9%
genesis	4	bible	34.0%
gibraltar	3	rock	78.5%
greyhound	2	dog	73.8%
guardian	3	uknewspaper	54.2%
halifax	3	novascotia	62.5%
hanover	2	germany	67.1%
hansen	3	juiceco	59.2%
harrier	2	jet	63.1%
heart	3	organ	77.7%
hercules	4	greekmyth	46.0%
himalayan	3	nepal	64.6%
horizon	3	organicdairyco	47.3%
journey	2	band	63.5%
juniper	2	networkco	60.9%
kansas	2	state	78.8%
loverboy	2	band	56.1%
lynx	4	animal	43.7%
madonna	2	band	89.4%
magellan	4	mutualfund	30.3%
magma	3	lava	44.7%
maltese	2	dog	65.1%
maxwell	2	band	54.3%
meatloaf	2	food	83.0%
ministry	3	church	54.2%
munchkin	2	kids	78.8%
oasis	2	band	59.2%
offspring	2	band	70.3%
onyx	3	company	48.2%
oriental	2	asian	68.8%
peregrine	2	company	50.9%
persian	2	iranian	82.5%
pink	3	band	63.2%
pioneer	3	spacecraft	42.7%
pointer	2	programming	67.6%
poison	2	substance	62.4%
police	2	law	91.4%
prince	5	band	45.1%
prodigy	3	band	44.2%
quantum	2	physics	79.0%
queen	5	band	42.3%
raptor	4	jet	40.4%
saturn	4	planet	51.7%
savannah	3	uscity	59.9%
scorpions	2	band	50.0%
siberian	2	dog	54.4%
snowshoe	2	sport	84.2%
somali	2	somalia	82.8%
sphynx	2	cat	70.0%
starship	2	band	53.1%
sting	4	band	57.4%
target	2	store	73.8%
tesla	2	physicist	80.3%
testament	2	bible	78.2%
time	2	clock	69.9%
tool	2	band	53.9%
toto	2	band	65.7%
train	3	band	48.3%
warrant	2	police	82.1%
wasp	3	insect	54.8%
wolverine	4	xmen	51.8%

Fig. A.1: Reference table showing the term, number of meanings per term, the most common (baseline) meaning, and the percentage of that baseline meaning.