An Intelligent Information Retrieval System Using Automatic Word Sense Disambiguation

Prasanna G. Ramasubramanian, Arvin Agah and Susan E. Gauch

Department of Electrical Engineering and Computer Science
The University of Kansas, Lawrence, KS 66045 USA

ABSTRACT

This paper aims to establish that an intelligent contextual information retrieval (IR) system can improve the quality of search results by retrieving more relevant results than those obtained with traditional search engines. Search engines capable of implicit, explicit, and no contextual retrieval were designed and implemented and their performances studied. Experimental results showed that search engines with contextual IR produce results that are more relevant, and the outcomes further indicate that there is no perceived gain in choosing specifically any one of the two approaches of implicit or explicit. The performance of the indexing mechanism, as it classifies document tokens with their appropriate contexts/word sense, was evaluated. The effectiveness of the word sense disambiguation process was found to depend to a great extent on the process (implementation) as well as the raw data (thesaurus).

KEYWORDS

icontextual information retrieval, word sense disambiguation.

1. INTRODUCTION

The World Wide Web is a treasure trove of information. The frontiers of cyberspace have grown at such a rapid rate that it has become virtually

impossible to keep track of all the web sites that are at one's disposal, let alone all the information. Search engines make a conscious attempt to cover as much ground as possible and keep track of as much information as they possibly can. Despite all this, the search engines still fall short. When one queries the web, how often does one get only what one wants? Nothing more...nothing less. The answer is not often enough. More often than not, the number of documents returned by search engines for a user search is phenomenal. Most users, however, do not go beyond the top 20 results and may be surprised by some of the result pages that are totally irrelevant. One contributing source for irrelevant results is the search engine process. Any attempt by the search engine to generalize a query from something more specific results in more search results (recall) at a cost, i.e., lower quality of results (precision). One approach aims to group many words under an umbrella concept. Because a specific query retrieves matches among all members of the concept, the number of search results is significantly more than before but the accuracy (defined as the ratio of the number relevant results to the number of returned results) suffers. Another contributing source to this problem is ambiguous querying – queries that mean something in some context and something else in another context.

Contextual information retrieval (IR) is a technique that aims at understanding the user's query before fetching the results. The query is disambiguated so that there is certainty as to what the user meant is what the search engine assumed it to be. This results in the realm of search results being narrowed to a subset of the original. Contextual IR may be explicit or implicit, where in former the user, when prompted, clarifies the usage sense of the query term; and in latter, the usage sense is determined from the user's current context. Both these techniques use word sense disambiguation to aid contextual retrieval. This paper aims to examine whether or not contextual IR using automatic word sense disambiguation increases quality of search results by providing more relevant results. This goal is achieved through building systems that implement both implicit and explicit contextual IR and contrasting them with a system that does not incorporate contextual IR. There are two aspects to the working of any IR system – indexing and retrieval. Just as it is important to evaluate the retrieval process in the light of the relevance of search results, it is also necessary to verify the working of the process that

helps implement contextual IR, namely, indexing. Context determination by word sense disambiguation is the value-added operation at the indexing level that facilitates contextual IR. This paper sets out to accomplish these means so that the objectives can be met.

2. RELATED WORK

The two primary areas of IR that this paper addresses are ambiguity of words and contextual retrieval. Ambiguity of words may be syntactic or semantic (Lexicon Interest Group, 1998). Syntactic ambiguity is easier to solve by computers as it draws from the language grammar, which is clearly well defined. "Part of speech taggers" work to resolve syntactic ambiguity by associating a part of speech to the words in the sentences that form the corpus, as dictated by the lexical and contextual probability (Daelemans et al., 1996). Lexical probability refers to a part of speech being more probable when there is no context present. Contextual probability helps make a part of speech decision based on words in close proximity. Various approaches to part of speech tagging include statistical, memory based and rule based approaches (Daelemans et al., 1996). Semantic ambiguity is more difficult to solve using computers and efforts in word sense disambiguation precisely attempts to do this. This paper does not implement a "part of speech" tagger and aims to resolve only semantic disambiguity in order to aid contextual retrieval.

Research in the field of word sense disambiguation has indicated that any word sense disambiguation technique can be broadly fitted, following (Lexicon Interest Group, 1998), into one of three approaches of knowledge-based, corpus-based and a hybrid approach. The knowledge-based approach draws from using a knowledge base, which is usually a machine-readable dictionary or a thesaurus. Examples include WordNet (Miller et al., 1993) and Roget's thesaurus (Roget, 1991). WordNet is a lexical database for the English language (Miller et al., 1993), which organizes nouns, verbs, adjectives and adverbs into synonym sets representing an underlying lexical concept rather than doing it alphabetically with cross-references. Because data are not organized in a single text file, command line and graphical user interfaces to the database provide a way for querying the database and obtaining results.

Yarowsky (1992) has successfully used statistical models of categories in Roget's Thesaurus (Roget, 1977) to carry out word sense disambiguation. The edition thesaurus divided the entire gamut of words into 1042 categories. Categories were equated to be an approximation of actual "word senses" and were used for the study. A "part of speech" tagger was used as a preprocessor to remove syntactic ambiguity. Because different conceptual classes tend to appear in distinguishable contexts and different word senses tend to belong to different conceptual classes, a context discriminator for classes would apply to word senses that belong to that class as well. If categories approximate classes, then context indicators for each of the categories would represent the word senses that are members of that category. Contexts representative of category were collected, salient words were identified and appropriately weighted and formed the basis for judging a word. The accuracy metrics performed on a collection of 12 ambiguous words ranged from 72% to 99% with a mean of 92% (Yarowsky, 1992).

WordNet has been used in many word sense disambiguation experiments. The primary divisions of WordNet include nouns, verbs, adjectives and adverbs. The key to WordNet organizing words by concept rather than alphabetically is the notion of *synset*. Synsets are nothing but a set of synonyms. Additionally, each synset in which a word appears is a different sense of the word (Voorhees, 1993). Voorhees attempted to disambiguate word senses of nouns by using the hypernomy/hyponomy relations on five standard test collections. Hypernomy/hyponomy are the "is a" relations. For example, board is a committee; board is a table, etc. The interface with WordNet used custom developed code that would return the synsets of which the word was a member. The disambiguation technique worked on the idea that ambiguous words occurring together could each determine the sense of the other. For instance, base, bat, glove and hit although ambiguous individually, when considered together signify the sense of baseball (Voorhees, 1993). If a set of categories could be built that covers the different senses of a word, then all words and their senses that fit in a category could be counted. The sense indicated by the category with the largest count is taken. For an "is a" relation identifying the category as the root or leaf node of the tree will not serve the purpose. A "hood" of a word is defined as the "largest connected subgraph that contains the word, contains only the

descendants of an ancestor of the word, and contains no synset that has a descendent that includes another instance of the word as a member" (Voorhees, 1993). It was found that there was degradation in retrieval performance when the indexing result was a sense-based vector rather than a stem-based vector. This was partly attributed to difficulty in disambiguation of short queries that caused the indexing procedure to yield an incorrect sense or no result. Also the hypernym/hyponym "is a" relation defines a generalization hierarchy that was inadequate in distinguishing the sense. Losing correct matches because of incorrect sense determination was more harmful for the retrieval performance than the spurious matches that stem-based vectors produced. Indications that indexing by concept rather than word form worked better was also observed when only the nouns in the document were disambiguated.

Another approach for word sense disambiguation is a corpus-based approach that emphasizes gathering information from a corpus by "training" on it rather than using a single knowledge base as knowledge-based approaches do. Training corpus may be raw, disambiguated, or prepared (artificial). While it is preferable to have a disambiguated corpus, it is not often possible because they are difficult to obtain and often expensive. Preparing a disambiguated corpus takes time and hand crafted ones are often limited to a few ambiguous words. If a corpus (such as a periodic publication) is present in two languages, then a mapping could be made between sentences of the two languages that mean the same. If the same word gets used in different contexts, i.e., different word senses, in the same language corpus, it is very likely that the translation of the senses would be different and can be used to distinguish the various senses of the word. Algorithms that match sentences across corpus of different languages exist and can be used. This corpus, referred to as sentence aligned parallel corpus can be used to group different occurrences of a word by its senses. A well-known example among these includes the Canadian Hansard available in French and English that was used by (Brown et al., 1991). An alternate approach works in the reverse way by grouping two words into a pseudo-word and replacing occurrences of either word by the combination and applying the algorithm on it. If the original corpus can be obtained, then this implies that the sense tagging was correct (Yarowsky, 1993).

Unsupervised disambiguation technique groups instances of a word by senses without actually knowing/revealing the senses. The process discriminates the word senses without knowing them. Pedersen and Bruce (1997) compared three algorithms (Ward's minimum variance, McQuitty's similarity analysis and the expectation maximization (EM) algorithm of Dempster, Laird and Rubine) performing unsupervised disambiguation. The corpus was largely derived from Wall Street Journal and the senses were primarily from the Longman Dictionary of Contemporary English (Longman, 1988). The three algorithms were tested on 13 different words using three feature sets with each experiment repeated 25 times. The algorithms clustered the instances of each word into different groups that could be mapped to the lexicon. The overall net accuracy (considering nouns, verbs, and adjectives) ranged from 65.3 to 66.2% for the three feature sets. The processes involved in using a disambiguated corpus include feature extraction, providing these features as input to a machine learning algorithm to form broad representation of the word senses and using these representations to disambiguate further. Features vary from one experiment to another and may be anything from all words in the proximity (50 words on either side) to a more limited subset (like only adjacent nouns, verbs, words on the left/right).

Brown et al. (1991) attempted to carry out word sense disambiguation while translating across languages using statistical methods. The study considered French and English sentences of the Canadian Hansard collection. The study earmarked seven features for a French word like word to left, right, first noun to the left, first noun to the right, first verb to the left, first verb to the right and the tense (if word is a verb) or the tense of the word two to the left. Features for an English word were the immediate left word and the word to the left of that. Answers to questions that specifically ask something of the informant (feature) can help disambiguate the sense. For example, a question could be "is the first noun to the right <something>?" Based on this a decision as to the most likely sense can be made because it is known that the word <something> dictates one sense of the ambiguous word more than any other sense. Preparing a list of candidate questions and choosing the most informative question solved the problem of constructing the binary question that provides the maximum information that can be used to disambiguation purposes. The flip-flop algorithm was used for this purpose. Considering a

French word *w* chosen from a vocabulary of V words, the number of probable questions is 2^V. The most informative question can be determined using the splitting theorem of (Breiman et al., 1984). The algorithm divided the English translation into two classes and used the splitting theorem to determine the best question that in turn divided the French vocabulary into two parts. The splitting theorem then divided the English translations so that maximal mutual information was shared with the French sets. The flip-flop algorithm alternated to improve on maximizing mutual information and the process converged to a partition of the French vocabulary with very high mutual information. The results showed a marked improvement of 13% from an accuracy of 0.37 when no disambiguation was done to an accuracy of 0.45 after this experiment (Brown et al., 1991).

Sanderson (1996) focused on word sense disambiguation and information retrieval. Although it included word sense disambiguation in good depth, its major focus was on study of retrieving from an additionally ambiguous collection and retrieving from a disambiguated collection. The first set of experiments on an ambiguous collection used WordNet as the corpus and implemented a slightly modified Yarowsky's design (Yarowsky, 1992) to accommodate WordNet instead of the Roget's thesaurus. The second set of experiments used the Reuter's document collection as its corpus. Experiments to determine retrieval effectiveness for variable query size were also performed. Additionally, pseudo-word based experiments on raw corpus to simulate ambiguous words were performed. It was observed that frequency distribution of occurrence of pseudo-word senses and ambiguous words was the same. It was found that query size affected word sense disambiguation and shorter queries were more affected by ambiguity than longer queries. It was also observed that disambiguator errors had a deleterious effect on retrieval effectiveness. Finally, Resnik and Yarowsky (1997) have presented and substantiated some of their studies on word sense disambiguation in IR. They have felt that evaluation of word sense disambiguation systems is yet to be standardized. They reiterate a fact we came across earlier which was the difficulty in obtaining adequately large sense-tagged data.

To perform contextual retrieval, the user's current working context must be determined. There have been many efforts that track the user to guess and facilitate the user's future course of action based on the past and the present.

Two such efforts, although not directly concerning IR, are Letizia and Webwatcher. These efforts provide an insight on "How does one determine the context?" and "How does the context help?".

Lieberman's (1995) Letizia is an agent that assists a user browsing the WWW. Letizia keenly follows users actions and tracks their behavior. It attempts to intelligently guess the user's next move and interests. The agent works in the background by concurrently exploring links and filtering them based on user, the interests and mood and makes suggestions of links that it feels would be more relevant to the user (Lieberman, 1995). The user can still override the system. While the user can evaluate the links, appropriateness the agent can work very fast in the background, *thinking* like the user. Retrieval is only partially dependent on current context and depends to a great extent on the users' taste and past record and other factors. Some indicators used in tracking a user include time spent on a page, book marking a page, saving a page, frequent returns to a document, and links overlooked.

Armstrong et al. (1997) have worked on developing a learning apprentice for the web called Webwatcher. This uses machine-learning methods to lead users to their goal and bases decision on user reaction and on success and failure of the user's action. Retrieved pages are enclosed in a template with hyperlinks replaced by one pointing to the Webwatcher. The user input serves as training information and is logged. Like Letizia, Webwatcher recommends links and pre-fetches further links based on the assumption that the user will follow its advice.

In this paper, a knowledge-based word sense disambiguation approach was chosen using existing knowledge bases such as the Unix dictionary and the WordNet lexicon (Miller et al., 1993). As this paper aims to enhance relevance of search results, user's current context is expected to facilitate the search because the system gets to know what the user was doing before searching. A complex system such as Letizia or Webwatcher is not required to watch the user's activities. If the document that the user was editing or browsing could be made available in a central location, such as the clipboard, then this would suffice to determine the context of the search query thereby, thinking like the user in much the same way as other systems, except that the system only considers a portion of the web agent's functionality.

3. RESEARCH APPROACH

This section details the approaches taken towards implementing the word sense based contextual retrieval system. This section also discusses the experiments done to verify the working of the system and evaluating its performance.

3.1 Implementation

The starting point for all search engines is a document collection. A spider picks up documents as it crawls the web. Because a spider implementation in beyond the scope of this paper, a small subset of 1100+ documents were chosen from an existing categorized corpus. The Open Directory Project was the document corpus used in this paper (Open Directory Project, 2002). Documents were picked up from the directory tree by recursively traversing it. One hundred and fifty documents were chosen from each of the top-level categories with no more than twenty from any subcategory.

3.1.1 *Search Engine Back End.* Contextual IR can be either explicit or implicit. User interaction is the key to explicit contextual IR. When one enters a search query that is ambiguous, one is prompted with a list of word senses that would help disambiguate the query. The user can then select the word sense that was meant and the search engine would modify the query term to reflect the word sense. Because the query term has been refined, retrieved documents are more in tune with the word sense rather than the general meaning of the search query. Simply put, explicit contextual IR ensures that the search engine understands the query the same way as the user does. If it is not sure, it asks.

Implicit contextual IR cuts down on the user communication. Instead, the user's current context is determined from the current activity. This activity could be web browsing or document editing or something else that the user was doing before the query of search engine. If the user's current activity could be established and the current document being browsed or edited is made available, for instance in a clipboard window, then implicit contextual IR could be carried out. This would mean query refinement that was done by

the user in explicit contextual IR would have to be done by the search engine somehow using the current working context (the document). In implicit contextual IR, the search engine assumes that what one is doing immediately before searching has a bearing on the search query and uses this to disambiguate the search query. How is the context determined from the document in the clipboard? In much the same way as any ambiguous token in a document would be classified by scanning for related words. Systems that tend to minimize interaction with the user sometimes tend to be intrusive and there is a fine line between making things easy and invasion of privacy. Cookies and clipboard access come in this realm. Browsers, due to safety considerations, disable clipboard access and an interface that can access the clipboard would have to be part of a separate application that the users will have to download, install, and be wary about. This would be a far cry from the standard web-based interface that search engines are known for. Added to this is the fact that Unix clipboards work differently (and are accessed differently) from the standard Windows clipboard. Implementing two interfaces (Web and stand alone application) is beyond the scope of this paper, especially because the idea is to evaluate contextual IR, keeping in mind the fact that design of the user interface is secondary to the contextual IR process. In this paper, users would be prompted to cut and paste their clipboard contents (their current working context) onto a text area on the web page and this will be the basis for implicit contextual IR.

To carry out contextual IR, the building blocks of a basic search engine needed to be modified, specifically, the module to create an inverted file. While the inverted file structure is being created from the tokens that appear in the various documents, it is necessary that all tokens that are ambiguous (having more than one meaning in different contexts) be classified with their appropriate word sense. The building blocks of the search engine with contextual IR capabilities are shown in Fig. 1. This approach presents two interesting issues: (1) How to decide whether a token is ambiguous? (2) How to determine the correct word sense in which the ambiguous token was used? In order to classify a word as an ambiguous word, once a list of all tokens that appear in all documents is available, then it is imperative that there be another list that contains all possible words that have more than one sense. In other words, it must be exhaustive. A simple lookup into this list of words would

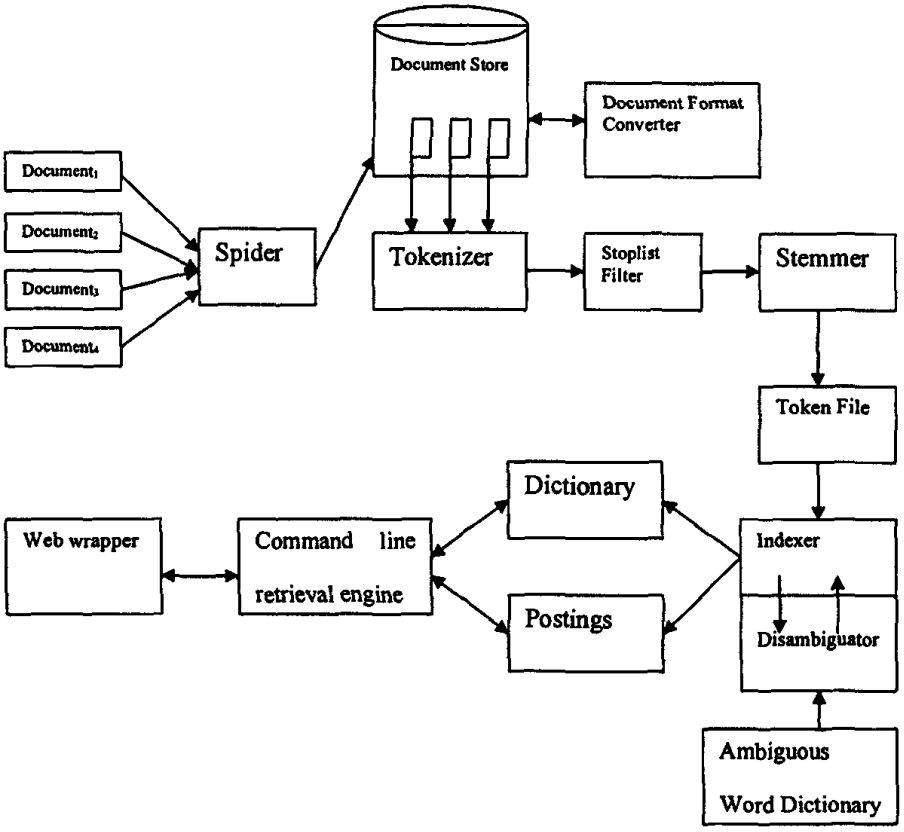


Fig. 1: Functional blocks of a search engine with contextual information retrieval.

reveal whether the token is ambiguous or not. This list would be stored on disk and henceforth shall be referred to as the ambiguous word file.

To determine the sense, the system needs to know the context in which that token was used in a document. Determination of context can be done by the examination of the document (or tokens because tokens are representative of the document content). This examination, done to disambiguate the sense, would necessitate examining other words that appear in the same document and that are in close proximity to the ambiguous word. Because a sense or context can only be dictated by a group of words, and not by a single word, the sense of the ambiguous token can be established. This again presents a problem in that how to know what words (if and when they occur in the document) mean what sense and how many senses does each word have.

What this implies is that for every word that appears in the ambiguous word file all possible senses that the word may have and some words that uniquely identify that sense need to be stored. This way, before preparing the inverted file, all tokens can be put through a three-step process which:

1. Determines if they are ambiguous.
2. If ambiguous, then examines the document for presence of related words that can uniquely identify the intended word sense.
3. Tags the token with the determined word sense.

Step 1 has been implemented by a lookup into the ambiguous word file. The lookup could use a simple linear search if the file size is nominal or a binary search if the ambiguous word file is large and is already sorted. Because the number of words that have more than one meaning is not trivial, a binary search algorithm with partial match capabilities has been used to speed up the lookup. Step 2 has been implemented by scanning all tokens that occur in that document for related words that can help identify the sense. Because a set of related words are already stored along with every possible sense in the ambiguous word file, for each occurrence of a related word in the document a counter was incremented. At the end of this process, the counter contained a sum that was reflective of the number of related words and their frequencies that were found in the document. This counter happens to be a *weight* of that particular word sense. The same process when repeated for all possible word senses, taking into account all associated related words, determined the corresponding *weights*. The word sense that produced the maximum of these *weights* was chosen as the sense in which that ambiguous word was used. Step 3 would be a mere concatenation of the token with the sense determined in Step 2.

For the implementation of this system, it was necessary to have the ambiguous word file with possible senses with related words in the first place. Because there were no resources that solely collected ambiguous words with their possible senses available in the public domain, it was decided to prepare such a file. In order to generate such a file, it was necessary to implement a mechanism that could do the following:

1. Given a list of words in the English dictionary, determine some information about each word including: (a) Possible meanings of that

word. (b) Words associated with each meaning including synonyms and related words.

2. For all such words, consider only words that have more than one sense and add them to the ambiguous words list with their possible sense.
3. For each sense, concatenate all related words separated from each other.
4. Repeat Steps 1,2 and 3 for all words.

Whereas the words in the English dictionary were obtained by using the standard Unix dictionary with approximately 25,000 entries, their possible meanings and related words were obtained by passing each word of the dictionary through the command line interface of WordNet, a lexical database. The output of WordNet has different word meanings and their coordinate nouns depicted in a pictorial tree format. Only words that have more than one meaning were considered and for each of these words and their corresponding senses, the *pictorial tree* is parsed using Awk (Aho et al., 1988) and the ambiguous word file is created.

The ambiguous word file created using this method considering only coordinate nouns for related words produced over 21,700 ambiguous word senses, sorted alphabetically (because the Unix dictionary which is the input is already sorted). The number of related words was kept to a maximum of 30. It should be noted that using the WordNet database for producing the ambiguous word file was less than ideal because the *related words* were sometimes synonyms and analogous rather than being truly related (words that could uniquely help identify a context).

This approach, when implemented, facilitates the tagging of word senses to the token so that the token would be represented by its word senses in the inverted file. The preparations of the dictionary and postings file that together constitute the inverted file remain the same as in a basic search engine. Because the inverted file generated for contextual IR has the tokens tagged with their senses, it must be stored elsewhere from the inverted file generated by the basic search engine. Each of these search engines would access their corresponding inverted files for retrieval purposes. It should be noted that both explicit contextual IR and implicit contextual IR search engines use the same inverted file for retrieval. They only differ in the way they refine the query and are similar in all other aspects.

3.1.2 *Weight Calculation Enhancement.* The enhancement that has been done to the standard weight calculation process is the normalization of the token weights by document length. The calculations are based on term frequency (tf) and inverse document frequency (idf). The purpose of normalizing the term frequency is to prevent bias occurring against the small documents. The actual frequency of a term appearing in a small document will be lesser than when it appears in a large document even though the term may be more relevant to the small document. To circumvent this, term frequency is normalized. Normalization of weight is carried out using the standard normalization procedure (dividing by the square root of sum of squares of individual term weights). The Normalized weight is calculated using the following equations:

$$\textit{term frequency } tf = \textit{actual frequency of token in the document / maximum frequency of any token in the document}$$

$$\textit{inverse document frequency } idf_{term} = \log(\textit{Corpus size}/(\textit{Number of documents containing the term}))$$

$$weight_{term} = tf * idf_{term}$$

$$weight_{norm} = weight_{term} * (\Sigma weight_i^2 \textit{ for all token}_i \textit{ in the document})^{-0.5}$$

3.1.3 *Search Engine Front End.* The user interface of a search engine with contextual IR capabilities only differed slightly with respect to the basic search engine. For the implicit contextual IR search engine, a text area had been provided on the web form allowing the user to copy and paste the current working context from an alternate window. For the explicit contextual IR search engine, the initial screen with a text box for the query term was the same as the basic search engine. If the user entered a query term that was ambiguous, however, then the user was prompted with a list of word sense choices as alternatives. Once the user had made a selection, the query was refined and sent to the engine for retrieval.

3.1.4 *Testing Interface.* As this paper focused as much on the evaluation as on implementation, it was necessary to have a testing interface that provided a single and simple interface for querying. The interface queried the three search engines (no contextual IR, explicit and implicit contextual IR).

The interface also undertook the responsibility of sending the appropriate data to the appropriate search engine. For example, the user entered working context document to the implicit contextual IR engine and the word sense chosen by the user among a set of word sense alternatives to the explicit contextual IR engine. The user interface combined the interfaces of explicit contextual IR and implicit contextual IR. In addition, the interface collected all results provided by the three search engines, permuted them to eliminate human bias and presented them in a randomized order along with a set of alternatives that allowed the user to rank the search results as relevant or otherwise. The results were shuffled using a Fisher-Yates shuffler that permuted the retrieved results in a random order (Fisher and Yates, 1938). If there were any documents that were duplicated as a consequence of appearing in more than one search engine results, then these duplicates were removed. The user's evaluation could then be analyzed in order to conclude whether the contextual IR systems performed better than the basic search engine. Some screenshots of the user interface are shown in Figs 2, 3, and 4.

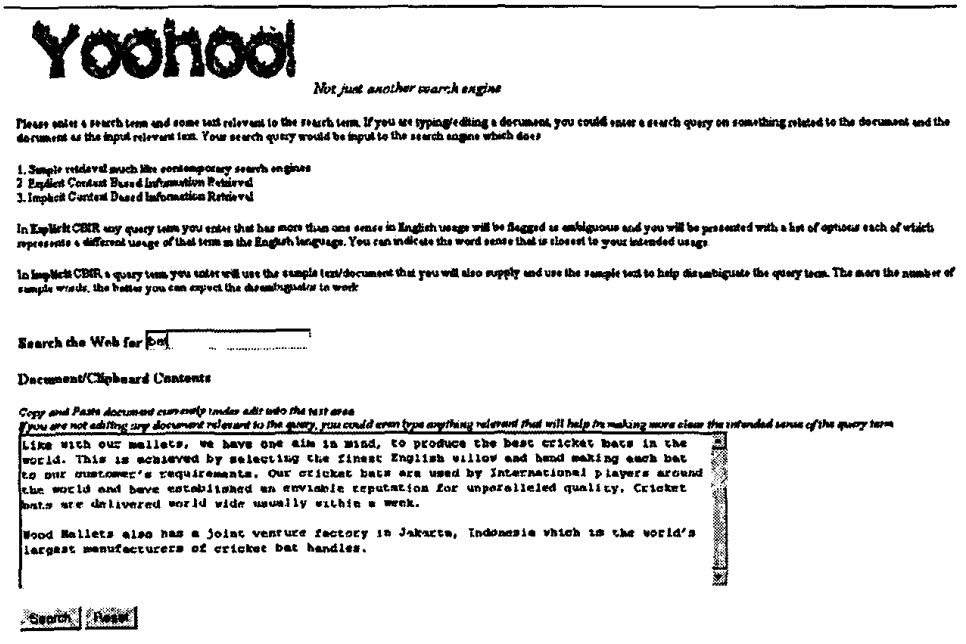


Fig. 2: User interface for implicit contextual IR (includes paste text area).

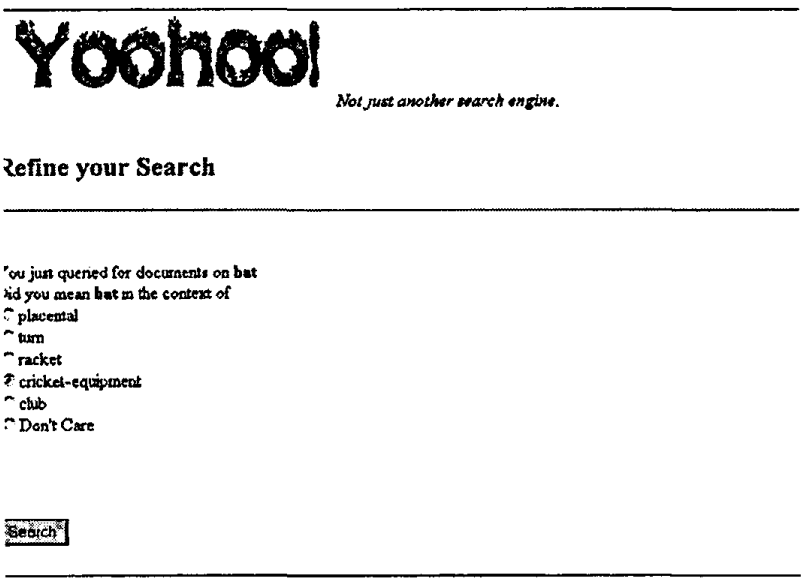


Fig. 3: User interface for explicit contextual IR.

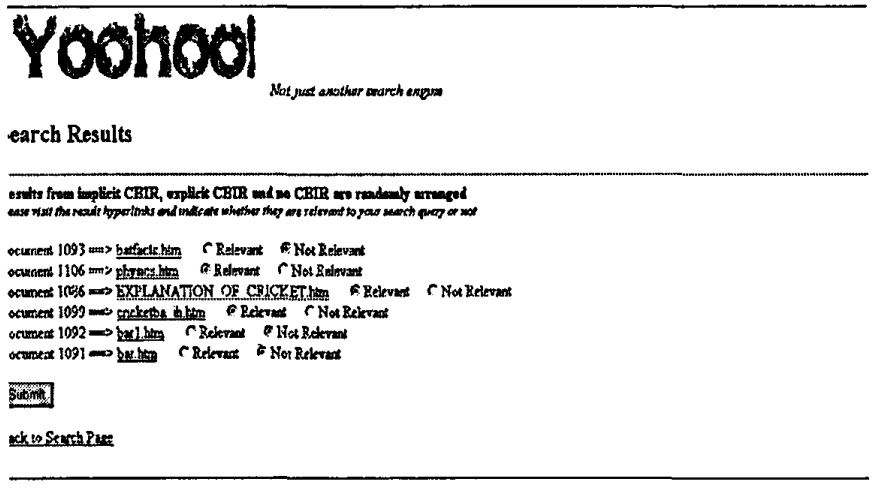


Fig. 4: Search results and user feedback page.

3.2 Process Verification and Performance Evaluation

A key component in this search engine is the word sense disambiguation module, which refines the tokens that find a place in the dictionary. The output of this module is the refined tokens that are used by the indexing routine to produce the dictionary and postings. In order to determine how effectively the disambiguation was being performed, a tap into the process was coded which wrote the output of the disambiguation block to a flat file. A

comparison could be made only when there were two or more entities, one of which had to be a standard. The standard chosen for the tests would be a file where all the ambiguous tokens have been disambiguated manually, using the reference file (ambiguous word file). This disambiguated file should have ideally had all tokens disambiguated. Practically due to the sheer size of data, however, approximately 42000 disambiguated words for the corpus, and the significant time for each disambiguation (because each token's occurrence in the web page and all its possible senses in the reference file must have been checked), a realistic measure of 5% of this size was chosen.

All ambiguous tokens were thus manually classified into one of the word senses defined for that term in the reference file. If no word sense was appropriate, the token was marked as an unclassified word sense. This file was then the basis for comparisons. The approach of the word sense disambiguation process could be further controlled using a preset threshold factor. This factor was based on how conservative the disambiguation algorithm was. A conservative algorithm would speculate as little as possible. Therefore, if a word's sense could not be exactly determined or if there were more than one sense that qualified as the answer, the algorithm would not choose either, but rather tag the token as unclassified. A conservative algorithm provided a lesser number of classified tokens but classified more accurately because speculation was minimal. A less conservative algorithm would select one of the choices even if it were not completely sure. Obviously, the number of classified tokens was more as were the number of correct results, but the ratio of correctly classified tokens to wrongly classified tokens was much less because speculations caused an increase in the number of wrong results just as it caused an increase in the number of correct results.

To verify the disambiguation and determine its performance, each of the files produced for varying preset threshold values were compared with the file generated by manually classifying the tokens. To simplify the process, the files were loaded into an ORACLE database (Cheevers and Tsai, 2002). A few SELECT query statements could fetch the necessary metrics including:

1. The number of classified and unclassified tokens
2. The number of correctly and incorrectly classified tokens among the tokens that were disambiguated

3. The number of correctly and incorrectly classified tokens among the tokens that could have been correctly disambiguated. This refers to only those tokens that had an answer (associated sense from the list of senses in the reference file) and which could have been correctly disambiguated had the disambiguator worked correctly.

As this paper intends to examine whether the contextual information retrieval improves the quality of search results by returning results that are more relevant, the contextual IR system is compared with a basic search engine.

Five users were chosen and asked to execute two queries each on the testing interface. The users regularly used search engines as part of their studies and were interested in querying a search engine with word sense disambiguation capabilities. The users were then told that three different search engines would process their query and the results would be collected and shown collectively. Users were not able to see or perceive the effects of contextual information retrieval at all. Instead, they saw the search results just as they would have if they would query any search engine. The purpose of not letting the user in on which search engine produced which result was to eliminate human bias from influencing subsequent user feedback which was crucial in evaluating the performance. Their ranking of the result page as relevant or otherwise was used to calculate precision of the search engine results.

The user was first prompted to enter the search term in the text box and copy and paste the current working context onto the text area. Then, the user was asked to refine the query in the following screen for the sake of explicit contextual IR. The search query was then sent to all three retrieval mechanisms and was executed. The results were presented in a random order to the user. The results were presented as a hyperlink (clicking on which would open the result page in a separate window) along with two radio buttons with values "Relevant" and "Not Relevant". The user's ranking of a result page and information tagged with the result (e.g., the search mechanism(s) that produced that result and the position of the result among the entire set of results retrieved by the same search mechanism) were collected and precision for each of the search mechanisms could then be easily calculated as the number of relevant documents divided by the number of retrieved documents. The result rankings were sent via email to the author.

4. RESULTS

The experiments performed aimed to verify the process that performs word sense disambiguation and also evaluate the relevance of the retrieved results. The results obtained from search engines performing without contextual retrieval, with explicit contextual retrieval and with implicit contextual retrieval were compared. The extent of correct word sense disambiguation during the indexing process for varying preset threshold values was studied and used to determine the best possible value. The precision metric was also calculated for a sample of ten different search queries taking into consideration the user feedback as to whether a result was relevant or not.

The basis for the first series of experiments was the threshold factor. The calculated threshold factor can be mathematically represented in the following manner. For any ambiguous token i, if n_i is the number of useful tokens (excluding stop list words) and if the two highest weighted related word sums are represented by max_i and $next\_max_i$ with the highest being max_i and the second highest $next\_max_i$ respectively, then:

$$Calculated\ Threshold = (max_i - next\_max_i)/n_i$$

While indexing, tokens were associated with a word sense only if the calculated threshold was greater than or equal to the preset threshold. A higher preset threshold implied that the disambiguation algorithm was conservative because the token was not associated with a word sense unless the difference was clear-cut. A lower preset threshold suggested a speculative algorithm in that a word sense was chosen even if there were two or more equally likely word senses applicable for that term. To ensure larger documents warranted a higher difference between the maxima because a trivial difference in the weighted related word sum was inadequate to distinguish the sense in a large document, calculated threshold was normalized by dividing it by the number of tokens in that document. The preset threshold factor was varied from 0 to 0.04 and the intermediate results of the indexing phase, i.e., the disambiguated tokens were analyzed.

Precision was the metric taken to evaluate the quality of search results. The user's query was submitted to the three search engines, the results were randomly shuffled, and the user was asked to rank the results as relevant or not relevant. The user's ranking was then used to calculate precision for each of the user searches for all the three search engines. A statistical t-test was performed to verify the statistical reliability of the precision results.

4.1 Analysis of Disambiguation during Indexing Results

As preset threshold was varied from 0 to 0.04, there was a significant difference in the number of tokens classified. As preset threshold was ramped up, it could be seen that the algorithm was behaving more conservative and was classifying fewer tokens. At a preset threshold of 0.04, the number of classified tokens was 2530 (6%). Any preset threshold value greater than 0.04 would have resulted in even fewer tokens being classified, which would have meant that more than 95% of the tokens were unclassified. Hence, all experiments were carried out varying preset threshold to a maximum of 0.04. Marginal differences between weighted related word sums among different word senses were not recognized as significant enough for decision-making. Although the accuracy of classification slowly increased, the number of tokens being classified sharply dropped, as shown in Fig. 5. The number of tokens classified fell sharply from 77% to 6% when preset threshold was varied from 0 to 0.04. Thus at a preset threshold of 0, the maximum number of tokens were classified (77%). The reason why this was not 100% is because the other input (ambiguous word file) to this disambiguation process was inadequate. It did not cover all possible senses (verb forms) and the choice of related words was sometimes flawed and unsuitable.

Although this experiment was carried out for all the ambiguous tokens (42,092) in the corpus of 1,111 documents, a smaller subset of 2,105 ambiguous tokens representing 5% of the whole was chosen to measure the extent of correctness of this classification. Because the intent was to classify each of the tokens manually into any of the senses defined in the ambiguous word file, classifying 42,092 tokens would have been a difficult task and 2,105 seemed more manageable. As shown in Fig. 6, the shape of the curves depicting the extent of classification for varying preset threshold values for

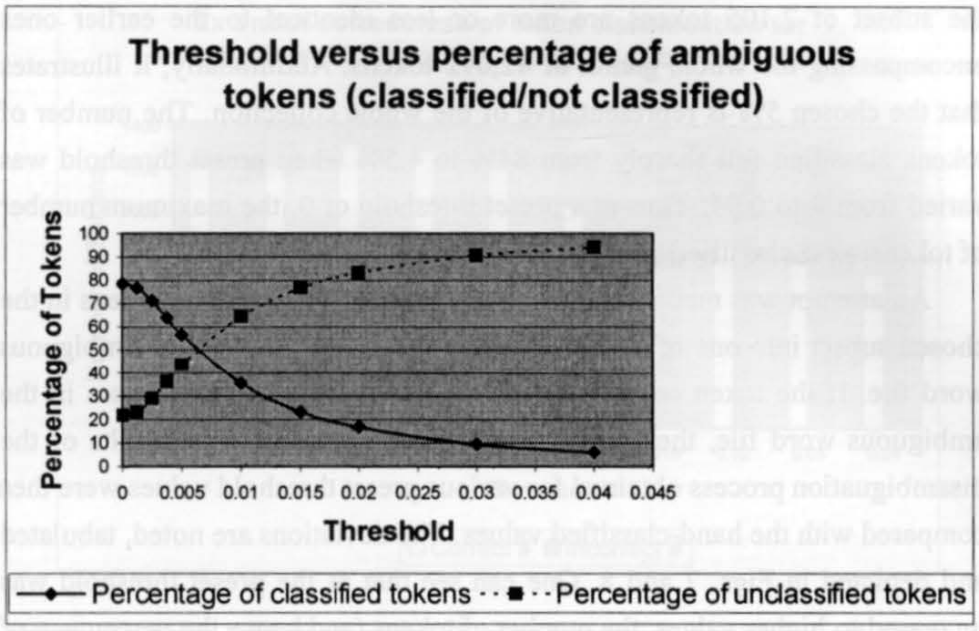


Fig. 5: Threshold versus percentage of classified/unclassified tokens.

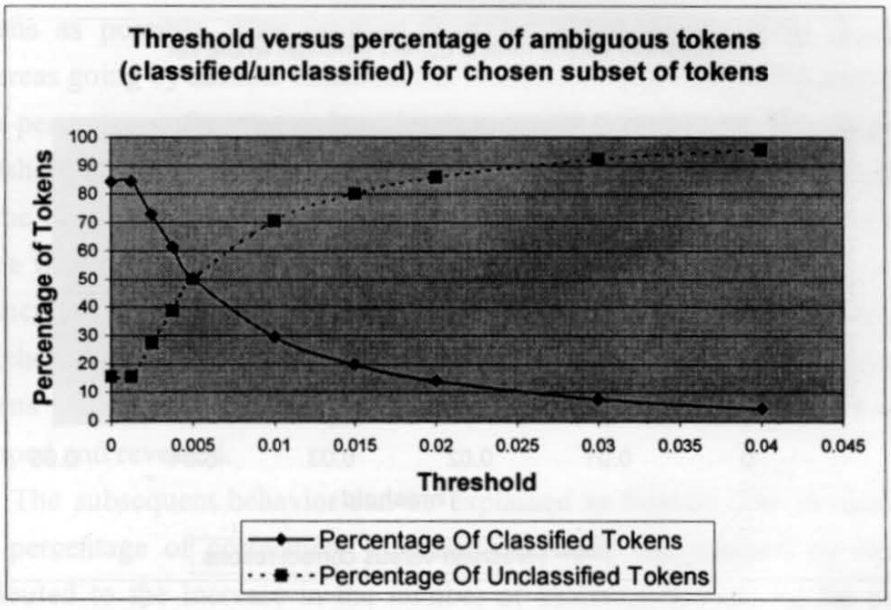
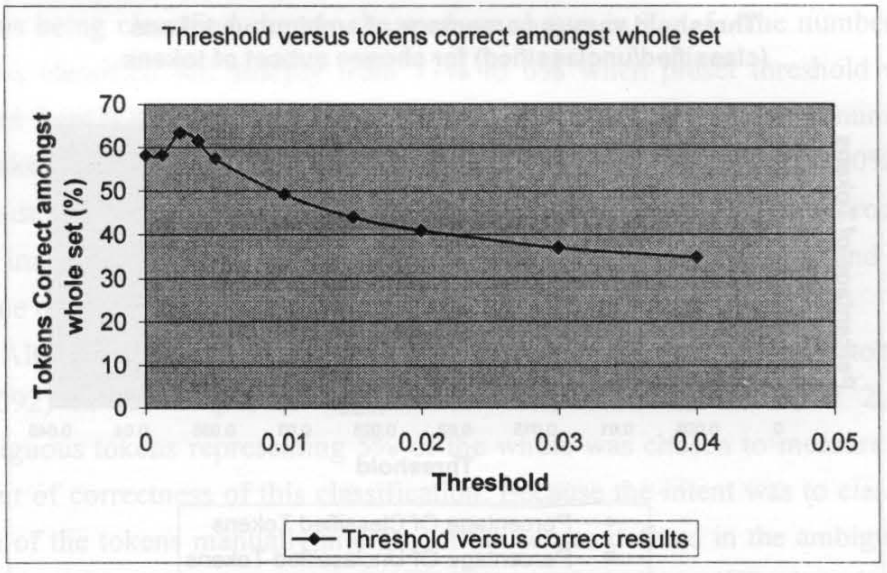


Fig. 6: Threshold versus percentage of classified/unclassified tokens for a subset.

the subset of 2,105 tokens are more or less identical to the earlier ones encompassing the whole gamut of 42,092 tokens. Additionally, it illustrates that the chosen 5% is representative of the whole collection. The number of tokens classified fell sharply from 84% to 4.5% when preset threshold was varied from 0 to 0.04. Thus at a preset threshold of 0, the maximum number of tokens were classified (84%).

An attempt was made to manually disambiguate each of the tokens in the chosen subset into one of the senses defined for that token in the ambiguous word file. If the token could not be classified into one of the senses in the ambiguous word file, then it was tagged as unclassified. The results of the disambiguation process obtained for various preset threshold values were then compared with the hand-classified values. The deviations are noted, tabulated and depicted in Figs. 7 and 8. One can see that as the preset threshold was increased to higher values, the number of tokens (and hence the percentage of tokens) classified correctly decreases. For the initial values of a preset threshold, however, a behavior quite contrary to the above occurred in that there was an increase in the percentage of tokens classified correctly.



Fig; 7: Threshold versus tokens correct for the token set.

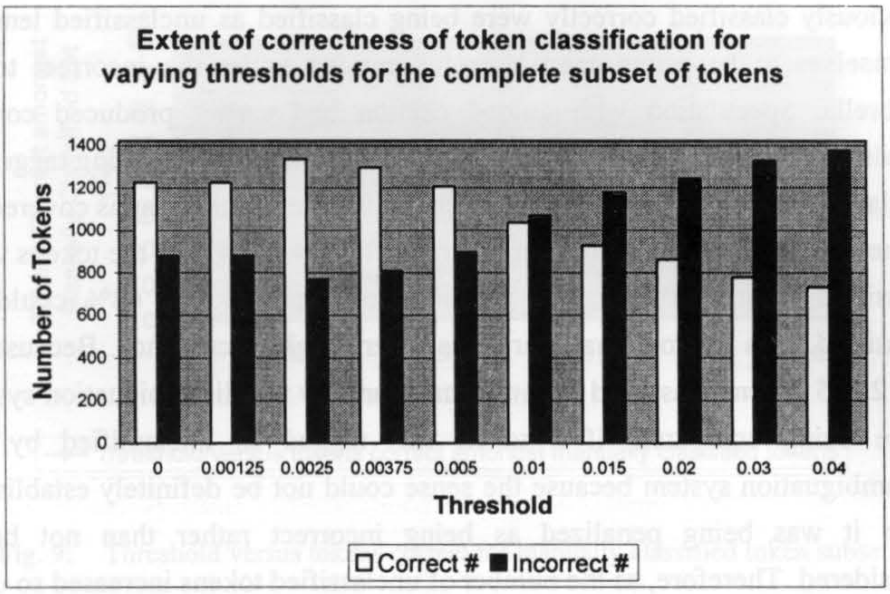


Fig. 8: Extent of correctness of classification for varying thresholds for the token set

The initial behavior was attributed to speculation with ample caution approach of the disambiguation algorithm. The area enclosed within preset threshold values 0-0.00375 more or less encompassed this behavior. At a preset threshold of zero, the algorithm speculated in order to classify as many tokens as possible. This resulted in 86% of the tokens being classified whereas going by manual classification resulted in only 68% being classified. The percentage of correct tokens increased with marginal increase of preset threshold because tokens whose sense was not clearly captured by the tokens in the document and ambiguous word file and were earlier associated with some wrong sense are now rightly being tagged as unclassified. This causes an increase in the number of correct tokens because the manual classification matched with this system-generated classification. As the number of classified tokens started decreasing with an increase in preset threshold, this trend stopped and reversed.

The subsequent behavior can be explained as follows. The decrease in the percentage of correctness as preset threshold was ramped up can be attributed to the increase in the number of incorrect tokens. As the preset threshold increased, the number classified sharply decreased because the disambiguation algorithm was not willing to take chances. Tokens that were

previously classified correctly were being classified as unclassified lending themselves to be prime candidates for coming under the incorrect token umbrella. Speculation with guided caution had earlier produced correct results. This implied that more and more of the tokens were getting tagged as unclassified. This effect was pronounced in the area of the graphs covered by preset threshold of 0.02-0.04. In this range only 4 to 14% of the tokens were classified whereas the manual classification reported that 68% could be classified. It is obvious that there was a very high discrepancy. Because all the 2,105 tokens classified both manually and by the disambiguation system were being compared, if a token was tagged as unclassified by the disambiguation system because the sense could not be definitely established then it was being penalized as being incorrect rather than not being considered. Therefore, as the number of unclassified tokens increased so does the number of incorrect tokens. Numerically, it was determined that 4-14% were classified, i.e., 86-96% were unclassified. But going by manual classification results, only 32% should have been unclassified. So the major bulk of incorrect tokens resulted from unclassified tokens (54-64% of 2,105 which was very significant) and that explained the sharp surge of incorrect tokens when preset threshold was slowly increased.

Rather than comparing the machine-based classification for all 2,105 tokens with those classified manually, only those tokens that had been tagged to a word sense in the manual classification were considered rather than considering all tokens irrespective of whether or not they could be classified. The comparison done for varying levels of preset thresholds produced the graphs shown in Figs. 9 and 10. Figure 9 is more or less similar to Fig. 7 except that the curve is lowered and closer to the X-axis. As preset threshold increased, the number of tokens tagged unclassified by the machine process also increased sharply. Not even a single token was tagged unclassified in the manually classified subset, however, because just considering only those that had been tagged to some word sense were considered. Hence, all those tokens that remained unclassified by the machine process came under the incorrect umbrella and the lowering of the curve was attributed to this reduction in correct tokens. Noteworthy is that the initial behavior when the percentage of correct tokens increased in the previous case (Figs. 7 and 8) was not clearly observed in this case (Figs. 9 and 10). The reason was that tokens that

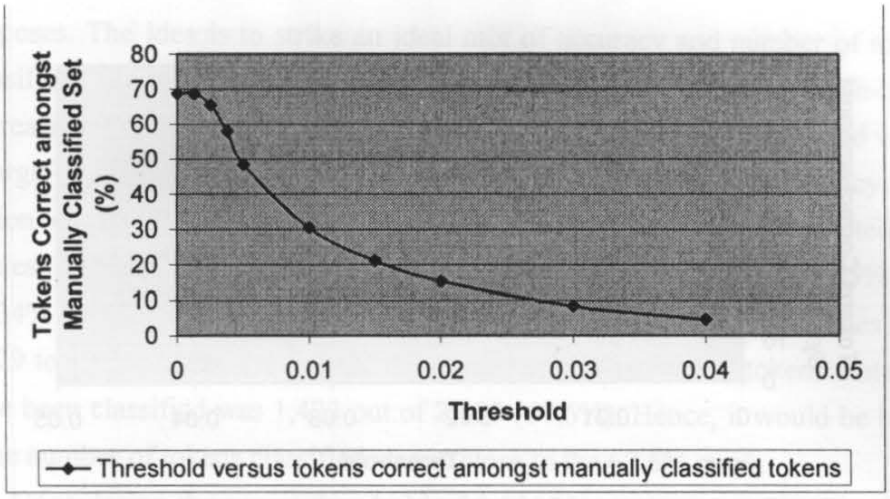


Fig. 9: Threshold versus tokens correct for manually classified token subset.

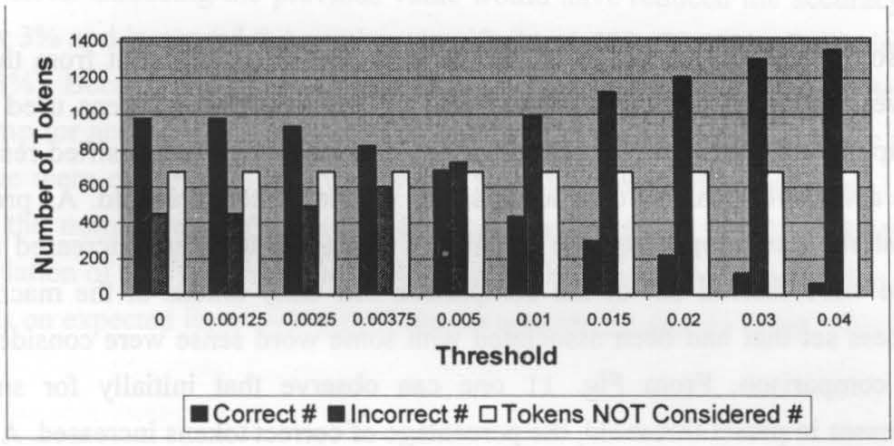


Fig. 10: Extent of correctness of token classification for manually classified token subset.

deserved to be unclassified were rightly termed so, and these matched perfectly the manually classified results. That is beyond the current scope, however, because tokens that were tagged unclassified by the manual classification were not considered. Only 68% of the tokens were considered because they had been associated with at least some word sense. The tokens not considered were also represented in to show how much of the entire space they occupied.

Whereas the previous experiments and analyses penalized unclassified results of the machine process as incorrect, this experiment however did not

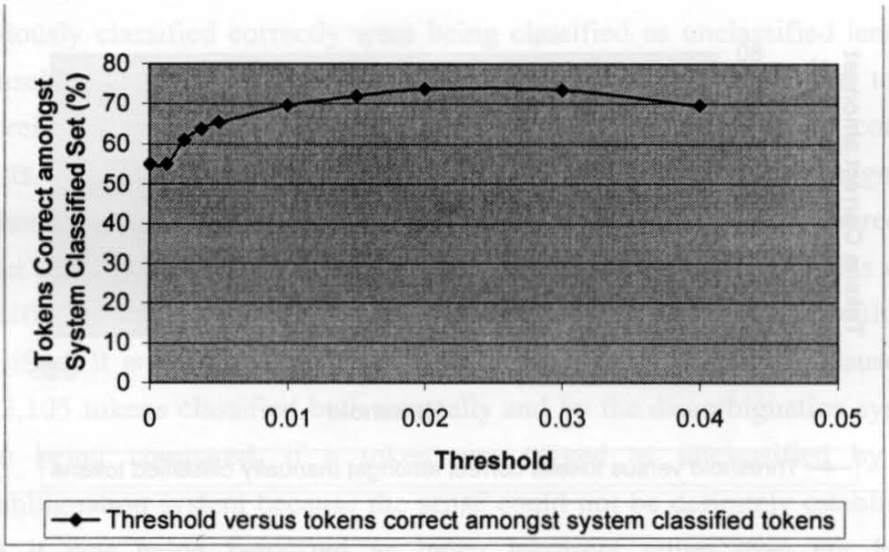


Fig. 11: Threshold versus tokens correct for system classified token subset.

do so. Therefore, the shape of the curve in Fig. 11 is different from those curves discussed earlier in Figs. 7 and 9. The number of tokens used for comparison between manually classified results and system-classified results was a variable quantity depending solely on the preset threshold. As preset threshold was ramped up, the number of unclassified tokens increased and these were filtered out of the comparison set. Only tokens in the machine process set that had been associated with some word sense were considered for comparison. From Fig. 11 one can observe that initially for small increases in preset threshold, the percentage of correct tokens increased. After reaching a maximum, however, the percentage evened out and more or less remained constant. This behavior can be explained as follows. When preset threshold was zero, the disambiguation algorithm tried to classify as many tokens as possible, speculation without adequate caution resulted in tokens being associated with some word sense even when there was a good element of doubt. As the algorithm adopted a speculation with caution approach, not all tokens were associated with a word sense and doubtful ones were flagged as unclassified. This improved the percentage of tokens correct.

Noteworthy is that although the percentages increased, the number of tokens classified fell sharply. From 1,777 tokens at a preset threshold of 0.0125 to 92 tokens at a preset threshold of 0.04, the sway was pronounced. A classification system where less than 5% were classified is useless for practical

purposes. The idea is to strike an ideal mix of accuracy and number of tokens classified. As both are competing and increasing one linearly causes a non-linear decrease of the other, the objective in hand is to choose a preset threshold where a large number of tokens are classified with a high degree of accuracy. The region enclosed by preset threshold values 0.0025 and 0.005 attracted our interest because in this range a reasonable accuracy between 60.95% and 65.34% was achieved. The number of tokens classified in this range varies from 1,529 to 1,056. Using the manual classification, the number of tokens that must have been classified was 1,423 out of 2,105 (67.6%). Hence, it would be better if the number of tokens classified was as close to the 67.6% mark.

Considering the preset threshold of 0.00375, an accuracy of almost 64% was achieved with the number classified at 61.33% mark (deviation from desired 67.6 was 6.27%). This appeared to be the best result for preset threshold. Choosing the previous value would have reduced the accuracy by over 3% and increased the number classified to 1,529 (72.63% - Deviation of 5.03%). Because there was little to choose in the deviation, it was decided to plump for accuracy and arrived at 0.00375 as the preset threshold. If the next value were chosen the accuracy would have increased still further by 1.5% but the number classified would have drastically fallen to 1056 (50.1% - Deviation of 17.5%), which was unacceptable. The histogram depicted in Fig. 12 is on expected lines with the tokens not considered sharply rising. This is in

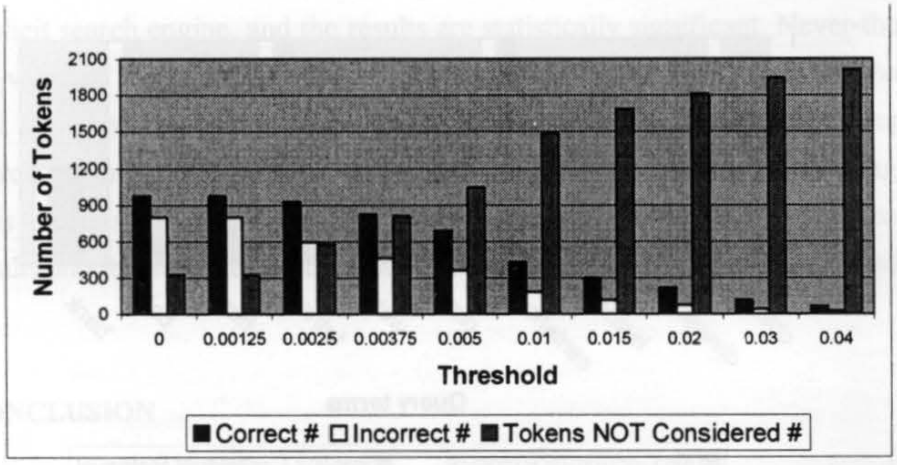


Fig. 12: Extent of correctness of token classification for varying thresholds for token set when only the subset of tokens deemed "classified" by system are considered.

tune with the observation that fewer tokens were classified and formed part of the comparison space. The numbers of correct and incorrect tokens were reduced gradually because fewer tokens were being classified, reducing the classified working set.

4.2 Analysis of Relevance of Retrieved Results

Five users were asked to execute two search queries each on the common interface to the three search engines performing no contextual retrieval, explicit contextual retrieval and implicit contextual retrieval. The results were shuffled and presented to the user. The users ranked the results as either "relevant" or "not relevant". The users did not know which search engine produced what result. The histogram detailing the obtained precision results for various query terms is shown in Fig. 13. From the results, it can be concluded that both explicit and implicit contextual IR produced the highest precision figure eight out of ten times (including the cases when both had the same precision figure). Only on two occasions did the search engine without contextual retrieval come out on top. It follows that both explicit and implicit contextual retrieval exert a positive influence on the relevance of search results. The average calculated precision while using a search engine without contextual retrieval was approximately 0.48, while the search engines with contextual retrieval produced gave precision values of 0.66-0.77, respectively.

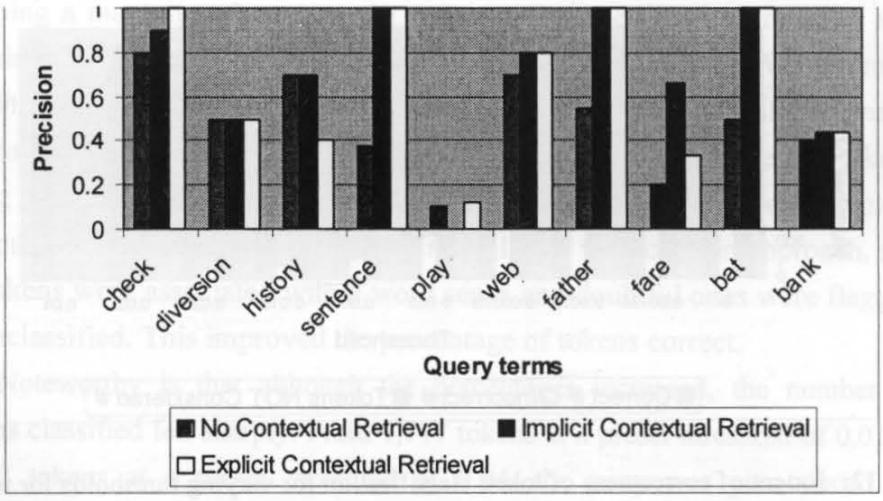


Fig. 13: Performance of retrieval strategies for various query terms.

A common test to determine statistical significance is the t-test (Peters, 2002). When the t-test was used to compare precision measures of explicit, implicit and no contextual retrieval two at a time, the obtained results were p of 0.8% between implicit and no contextual retrieval, 3.6% between explicit and no contextual retrieval and 13.5% between implicit and explicit contextual retrieval. The null hypothesis (usually stated negatively) in the three cases was "Precision values of Sample 1 is not less than precision values of Sample 2" where (Sample 1, Sample 2) were (Basic, Implicit), (Basic, Explicit), and (Implicit, Explicit), respectively for the three t-tests. The alternate hypothesis stated, "Precision values of Sample 1 is less than precision values of Sample 2" where the same as previously for the three t-tests. The t-test carried out was a paired one-tailed t-test; paired because individual data points in the 2 data sets were paired because they were results obtained for the same stimulus (query term) and one-tailed because the direction of difference between the two means was specified and not just the fact that they were different. Values of p below 5% suggest that the null hypothesis must be rejected. Otherwise, the null hypothesis can be accepted. The precision figures of the three search engines for ten different query terms were the inputs to the ttest.

The p values of 0.8% and 3.6% for the t-tests conducted on (Basic, Implicit) and (Basic, Explicit) data sets were less than 5%. Therefore, the alternate hypothesis is true. Specifically, the precision values of basic search engine are less than the precision values of implicit search engine, and the precision values of basic search engine are less than the precision values of explicit search engine, and the results are statistically significant. Never-theless, a p value of 13.5% for the t-test on (Implicit, Explicit) data sets implies that the null hypothesis can be accepted. Specifically, the precision values of implicit search engines are not less than the precision values of explicit search engines. This result shows that contextual retrieval improves the relevance of search results and its effectiveness does not depend on the actual approach adopted.

CONCLUSION

The objective of this paper was to implement an automatic word sense based contextual retrieval system and study whether this process could

improve the quality of search results. The inference drawn from the results was that a preset threshold value of 0.00375 was the best possible and that the quality of results depends to a significant extent on the inputs (specifically the ambiguous word file). The preset threshold of 0.00375 was reached after taking into consideration the primary objective of obtaining as many correct results as possible, at the same time keeping the incorrect results to a minimum. A conservative system that classifies very few results with a very high percentage of accuracy would also be useless because a great deal of the ambiguous words would remain unclassified. A mix of speculation with caution, when not absolutely sure, yielded significantly better results than did plain guessing and no guessing.

The results produced by the same search query when executed on search engines performing no contextual retrieval, explicit contextual retrieval, and implicit contextual retrieval were useful in determining the precision. The judgment of the relevancy of the results was made by the users. Their results were used to calculate the precision. We showed that both explicit and implicit contextual retrieval performed better than traditional retrieval. The statistical significance of the results was also established. Both implicit and explicit contextual retrieval produced better precision figures than did the search engine without contextual retrieval. The results obtained indicate that the effectiveness of contextual retrieval does not depend on the approach taken, as both implicit and explicit contextual retrieval worked well. Thus, the utility of word sense based contextual retrieval has been established. This system promises to improve the quality of search results and, consequently, user satisfaction when actually implemented on search engines.

Future work includes using a part of speech tagger to resolve syntactic ambiguity of query terms. The problem of the same word being represented by more than one sense (during indexing) can be overcome if only words in the near proximity (such as 20 words on either side) are used for the weighted related word sum. An approach that would work when more than one word sense is likely for the query term would be a weighted retrieval of results. The sense that is more probable would have more results retrieved and the less likely sense would have fewer retrieved results. The actual numbers would be dependent on how likely each of the senses is (proportional to weighted related word sum). In this way, the probability that at least some of the results

are always correct can be ensured, and the correct sense is captured however ambiguous the query may be. Locating a more suitable public-domain machine-readable thesaurus would have a telling effect on the retrieval process. An ideal thesaurus would provide fewer and more relevant word senses and would accommodate verb senses as well. A mechanism to weigh words by their uniqueness value rather than by their frequency count would be beneficial as well.

ACKNOWLEDGMENT

This work was sponsored in part by the National Science Foundation under grant EIA-9972843.

REFERENCES

Aho, A.V., Kernighan, B.W., and Weinberger, P.J. 1988. *The AWK Programming Language*, Reading, MA, USA, Addison-Wesley.

Armstrong, R., Freitag, D., Joachims, T. and Mitchell, T. 1995. Webwatcher: A learning apprentice for the World Wide Web, in: *Proceedings of the AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments*, Stanford University, Stanford, CA, USA, March, 6-12.

Baeza-Yates, R. and Ribeiro-Neto, B. 1999. *Modern Information Retrieval,* New York, NY, USA, Addison-Wesley/ACM Press.

Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. 1984. *Classification and Regression Trees.* Monterey, CA, USA, Wadsworth & Brooks/Cole Advanced Books and Software.

Brown, P., Della Pietra, S., Della Pietra, V., and Mercer, R. 1991. Word sense disambiguation using statistical methods, in: *Proceedings of the 29th Meeting of the Association for Computational Linguistics (ACL-91)*, Berkeley, CA, USA, June, 264-270.

Cheevers, S., and Tsai, J. 2002. *Oracle 9i Database Release 2 Product Family*, CA, USA, Redwood Shores, May, http://technet.oracle.com/products/oracle9i/pdf/9idb_rel2_prod_fam.pdf.

Daelemans, W., Zavrel, J., Berck, P. and Gillis, S. 1996. MBT: A memory-based part of speech tagger-generator, in : *Proceedings of the Fourth Workshop on Very Large Corpora (WVLC)*, Copenhagen, Denmark, 14-27.

Fisher, R.A., and Yates, F. 1938. *Statistical Tables*, London, Example 12.

Lexicon Interest Group. 1998. Preliminary recommendations on lexical semantic encoding. The *Expert Advisory Group on Language Engineering Standards (EAGLES) Interim Report*, Pisa, Italy.

Lieberman, H. 1995. Letizia: An agent that assists web browsing, in: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI95)*, Montreal, Canada, August, 924-929.

Longman. 1988. *Longman Dictionary of Contemporary English*, New Edition, Essex, England, Harlow.

Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D., and Miller, K. 1993. Introduction to WordNet: An on-line lexical database. *Cognitive Science Laboratory at Princeton University*, www.cogsci.princeton.edu/~wn/.

Open Directory Project. 2002. http://dmoz.org.

Pedersen, T., and Bruce, R. 1997. Distinguishing word senses in untagged text, in: *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, Providence, RI, USA, August, 197-207.

Peters, J.S. 2002. How to do a t-test with MS-Excel 2000? www.cofc. edu/~petersj/SMFT639_MSExcel_T-test.html.

Resnik, P., and Yarowsky, D. 1997. A perspective on word sense disambiguation methods and their evaluation, in: *Proceedings of the ACL SIGLEX '97 Workshop on Tagging Text with Lexical Semantics: Why, What and How?*, Washington DC, April, 79-86.

Roget, P. 1991. Electronic thesaurus derived from the *Roget's Thesaurus*. MICRA, Inc., May.

Roget, P. 1977. *Roget's International Thesaurus*, Fourth Edition, New York, NY, USA, Harper and Row Publishers Inc.

Sanderson, M. 1996. Word sense disambiguation and information retrieval. Ph.D. Dissertation, *University of Glasgow Department of Computing Science*, September.

Voorhees, E.M. 1993. Using WordNet® to disambiguate word senses for text retrieval, in *Proceedings of the ACM conference on SIGIR-93*, Pittsburgh, PA, USA, June, 171-180.

Weiss, S. 1997. *IR Vocabulary*. http://www.cs.jhu.edu/~weiss/glossary.html.

Yarowsky, D. 1992. Word-sense disambiguation using statistical models of Roget's categories trained on large corpora, in: *Proceedings of the 14th International Conference on Computational Linguistics*, Nantes, France, July, 454-460.

Yarowsky, D. 1993. One sense per collocation, in: *Proceedings of ARPA Human language technology workshop*, Princeton, NJ. USA, March, 266-271.