# Oracle at TREC 10: Filtering and Question-Answering

Shamim Alpha, Paul Dixon, Ciya Liao, Changwen Yang
Oracle Corporation
500 Oracle Parkway M/S 4op8
Redwood Shores, CA 94065 USA
trec@us.oracle.com

**Abstract:**
Oracle’s objective in TREC-10 was to study the behavior of Oracle information retrieval in previously unexplored application areas. The software used was Oracle9i Text[1], Oracle’s full-text retrieval engine integrated with the Oracle relational database management system, and the Oracle PL/SQL procedural programming language. Runs were submitted in filtering and Q/A tracks. For the filtering track we submitted three runs, in adaptive filtering, batch filtering and routing. By comparing the TREC results, we found that the concepts (themes) extracted by Oracle Text can be used to aggregate document information content to simplify statistical processing. Oracle's Q/A system integrated information retrieval (IR) and information extraction (IE). The Q/A system relied on a combination of document and sentence ranking in IR, named entity tagging in IE and shallow parsing based classification of questions into pre-defined categories.

## 1. Filtering based on Theme Signature

As a first time filtering track participant, Oracle submitted runs for adaptive filtering, batch filtering and routing this year. Only linear-utility optimized runs were submitted for adaptive filtering and batch filtering. The filtering system is built based on the Oracle 9i database with PL/SQL - an Oracle supported database access language. Since the routing sub-task outputs the top 1000 ranked documents per category, and the training process and similarity score calculation algorithm are the same for batch filtering and routing, we will focus our discussion on batch filtering and adaptive filtering.

The filtering system can be divided into three parts based on functionality:

a. Theme Vector Generation
b. Training
c. Classification

### Theme Vector Generation

Theme vector generation generates a theme vector for each document. It is built-in functionality of Oracle Text, the information retrieval component of the Oracle database[2]. A theme vector containing a list of themes (concepts) and associated weights carries all information of a document used in classification. Themes are normalized words having meanings individually and extracted based on the Oracle Text knowledge base. The knowledge base is built in-house and contains about 425 thousand concepts classified into 2000 major categories. These categories are organized hierarchically under six top terms: *business and economics, science and technology, geography, government and military, social environment, and abstract ideas and concepts*. This knowledge base is built to support concept search and retrieval. For this TREC work, the ConText knowledge base was employed in our filtering system to preprocess documents and generate concept terms. Although the Oracle Text user extensible knowledge base functionality allows users to modify the built-in knowledge base using user specified thesaurus, we used the knowledge base without any modification. We believe augmenting the knowledge base using domain specific information could improve filtering performance. In the theme generation, known phrases are recognized using a greedy algorithm, unknown words and proper name phrases are recognized and treated as themes. Words and phrases are normalized to their canonical forms. Every normalized term is a potential theme for a document.

Theme weights are used to rank the semantic significance of themes to the aggregate document content. Themes are assigned initial weights based on their lexical flags in the knowledge base. Next, several factors derived from the structure of a document and the frequency of the theme in the document are employed to modify the initial weights of the themes. For example, the first few terms inside a sentence have higher weights than the terms at the end of sentences to account for "fronting" and sentence focus.

Generated theme vectors are normalized to have unity length before being sent to the training or classification process. This normalization can be written as :

$$w_j^n = \frac{w_j}{\sqrt{\sum w_j^2}}$$

where $w_j$n and $w_j$ are the j-th component (j-th theme term) weight of theme vector **w** after and before unity normalization respectively.

Our prior experience demonstrates that themes are superior to text tokens in representing text documents of medium to large size for classification purposes. Oracle Text first tokenizes documents and then processes these tokens using a greedy maximal match algorithm to generate themes. A brief  description of the process to generate themes from tokens may shed some lights on the reason why themes are superior to tokens in classification. After  finding a token, Oracle Text gets the part of speech information from the knowledge base or finds phrases based on the greedy algorithm and lexical knowledge base. If the token is a noun, a canonical form is used as a normalized form for this token, such as "tastefulness" with canonical form of "tasting" and "dull-headedness"  with canonical form of "stupidity". If the token is a non-noun, a base form is found based on the knowledge base or morphology if the token does not exist in knowledge base. After that, a normalized noun form is used as the theme form for the non-noun base form. For example, "steadied" has a base form of "steady" which corresponds to a normalized form of "steadiness". The following differences between themes and tokens may contribute to the different behaviors in classification:

1. Themes can handle phrases while tokens can not without a lexicon.
2. Themes are represented with normalized forms of concepts, while tokens are forms with or without stemming. Word normalization is mostly based on lexical knowledge, while stemming of a token is mostly based on morphology.
3. The weight of  a theme expresses the lexical information of a term, locations in a document, and term frequency. The weight of a token typically only includes the information of term frequency.

For the classification task no parent themes (broader terms) were used. Whether or not the parent themes improve the learning quality is actually an open question. One side says a specific word should be more important for representing a document and a parent theme may act as a common word. On the other hand, one of the parent themes may tell exactly what a document is about. However, that might depend on the level of parent theme and depend on whether or not the hierarchy of the knowledge base represents the same knowledge hierarchy in the classification application. We intend to investigate this issue thoroughly in the future.

**Training**

The training process calculates the summation of all relevant theme vectors for each category. The summation result serves as the original theme vector for one category. Because of accumulation, the number of themes in the category theme vector can be large. Experiments show that reducing some common themes and less frequent themes for the category theme vector can improve classification accuracy. Theme reduction can also reduce the resource usage and improve classification performance. We adopt a two-step theme reduction. The first step is to choose the top 400 themes with highest theme weights in the category theme vector. As mentioned earlier, the theme weight obtained from Oracle Text combines information about the lexical significance, word position inside one sentence, and occurrence frequency inside the document. Those top 400 themes in the category theme vector are the most frequently occurring and significant words to the category. Another rationale for choosing the theme by weights is that words with little meaning have lower weights and therefore can be removed.

The first step of theme selection based on the theme weight may choose some themes which are common in lot of categories. These common themes are not specific to one category and may produce extra noise to the classification process. The second step of theme reduction is to choose themes which are more specific to one category. We use a chi-square test for theme selection [3]. In specific, we choose a theme if the null hypothesis that this theme is independent of the considered category can be proved not true. The themes will be chosen if:

$$\frac{N(Nr_t - n_t R)^2}{Rn_t(N-R)(N-n_t)} > 3.84$$

where N is the total number of training documents
R is the number of training documents in this category
$n_t$ is the number of training documents containing this word
$r_t$ is the number of training documents in this category and containing this word.
value 3.84 is chosen because the confidence of chi-square test is 0.95.

By chi-square test, the average theme vector size can be reduced to 280. In the original category theme vector, the weight is the summation of each document’s theme weights; those weights help us to choose the top 400 themes for the category. However, during the classification process, we use Robertson-Sparck Jones weights [4] as term weights in category theme vectors. The weights are calculated based on the statistical characteristics of the training set and relevant category:

$$\log\frac{(r_t + 0.5)(N - R - n_t + r_t + 0.5)}{(n_t - r_t + 0.5)(R - r_t + 0.5)}$$

This formula is obtained from the Bayesian statistical model. The Robertson-Sparck Jones weight is the component weight for one term to estimate the log-odds of an given document belonging to the considered category in the assumption that terms are independent [5].

**Classification**

Before classification, category theme vectors are normalized to have unity length. In classification, the similarity scores S between the incoming document and each category are calculated as a dot product between the document theme vector vd and category theme vector vc, that is S = vd.vc. The document is classified to the categories in which the similarity scores are larger than the corresponding category thresholds. The predefined thresholds are determined from the relevance information either from the training set in batch filtering or from feedback in adaptive filtering.

**Threshold Determination**

**Batch filtering**

Each category has its own threshold to determine if a document can be classified to it based on the similarity score. In order to determine the threshold for one category, we use the classification module to calculate the similarity score between all training documents and the considered category. For any given threshold x, we can get the following contingency table as we know the actual categories of each training document.

| | Relevant | Not Relevant |
|---|---|---|
| Retrieved | $R_+$ | $N_+$ |
| Not Retrieved | $R_-$ | $N_-$ |

We can define a utility (goal) function of the about 4 numbers, say $f(R_+,N_+,R_-,N_-,x)$. x appears explicitly in the function because R+,R-,N+ and N- are all functions of the threshold x. The threshold is chosen to maximize the function f.

$$\text{Threshold} = x: \max_{x} f(R_+,N_+,R_-,N_-,x)$$

In TREC-10, we submit the batch filtering run based on optimization function of linear-utility, which is $f(R_+,N_+)=T10U=2R_+ - N_+$.

In implementation, one can generate a sorted array of training documents ordered by similarity scores to the given category with a decreasing sequence. The relevance information of documents in the sorted array before any given document can determine $R_+$, $N_+$ at the threshold value equal to the similarity score of this document. For each document in the sorted array, one then can calculate the T10U function value at the threshold value equal to the similarity score of this document based on calculated $R_+$, $N_+$. Because the array is sorted such that the similarity scores are decreasing, one therefore can draw a curve of T10U vs threshold. As threshold decreases from the largest value, the T10U values first increase because more relevant documents are located at the positions having larger similarity scores, and decrease after reaching a peak. The peak position corresponds to a similarity score , whose value is the optimized threshold value to maximize T10U function. This calculation makes the assumption that the training set similarity score distribution and T10U quantity is similar to that of the test set.

**Adaptive training**

In adaptive filtering, we first built initial category theme vectors from training process of an initial training set, which contains two relevant documents per category. The training process is the same as we discussed above. The initial category threshold is set to be 40% of the minimum similarity score of the two relevant documents with the considered category. We then classify the test documents in a batch mode with each batch containing 2000 documents coming from the test set stream. After classification of each batch, feedback information including the relevance judgments and the similarity scores is sent to adaptive training, see Fig.1.

Adaptive training includes updating category theme vectors and category thresholds. In order to update the category theme vector, we have to maintain the original category theme vectors which are the theme vectors before any theme selections and has the theme weights from summation of Oracle Text theme weights. To keep the number of themes in the category theme vector from becoming too large, we limit the size of each original category theme vector to a maximum of 2000. The extra feedback training document theme vectors are added to the original category theme vectors using Widrow-Hoff algorithm [6].

$$w_j^n = w_j - 2z(\mathbf{w} \bullet \mathbf{x}_i - y_i)x_{i,j}$$

where $w_j$, $w^n{}_j$ are the weights for j-th component of the category theme vector before and after adaptive training, respectively. $x_i$ is the theme vector of i-th feedback document, $y_i$ the relevance judgment of the i-th feedback document with the considered category with $y_i = 0$ denoting not relevant, $y_i=1$ denoting relevant. $\mathbf{w}.\mathbf{x}_i$ denotes the dot product between the theme vector $\mathbf{w}$ and $\mathbf{x}_i$. $z>0$ is learning rate and is set to 0.2.

The Widrow-Hoff algorithm generates a list of updated themes and weights. We maintain only the top 2000 highest weight themes for each category. The weights here are calculated quantities from Oracle Text theme weights. We apply theme selections and employ Robertson-Sparck Jones weights as category theme vector weights for classification as discussed in the above training section.

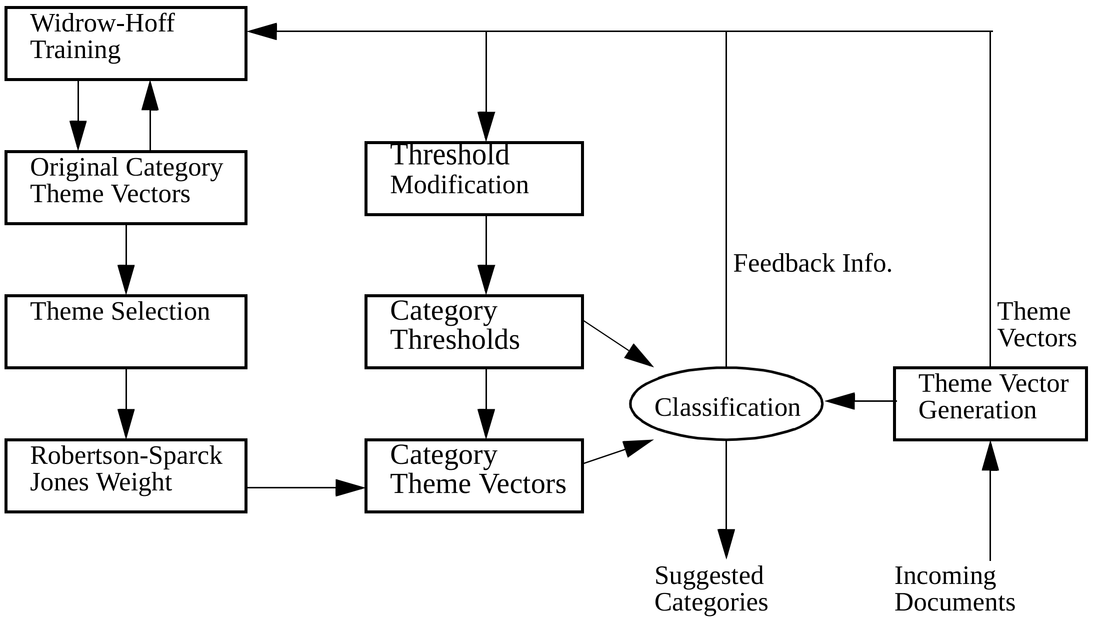

**Figure 1.** Adaptive filtering diagram

Thresholds can be calculated based on the relevance information and similarity scores of all previous feedback documents in the way we discussed in the threshold determination section. However, that calculation may take unacceptably long time. Instead we adopt a simple method to adjust the existing thresholds only based solely on current feedback information.

Thresholds can be adjusted by calculating the optimal threshold for the extra feedback training set as discussed in threshold determination section. We denote the optimal threshold as optimal_threshold_extra_training, then the updated threshold is :

updated_threshold = old_threshold + C (optimal_threshold_extra_training - old_threshold)

where C is a learning parameter and is set to 0.3. We note that the feedback batch size and the learning parameter C are relevant parameters, if the feedback batch size is small, the optimal threshold for the extra feedback documents may vary a lot, one then choose a smaller C. C has to be chosen such that the updated thresholds change with the feedback process in a systematic and stable way.

## Submission Result and Discussions

Oracle submitted three runs. They are listed in the Table 1, and Table 2, with adaptive, batch runs in table 1 and routing in table2, respectively. The numbers in the parenthesis are the median value of all participants. The median values are the (N/2+1)-th value in sorted decreaseing list if the number of participants N is even. Except the precision for batch filter, all numbers in our submitted runs are above median.

We note that the routing behaves better than batch filtering. The fact that batch filtering system has only one more component: thresholding, than routing implies that our threshold determination is not quite good for batch filtering. In batch filtering, the threshold can not be adjusted. Once a threshold is determined, it is used to classify the whole test set without any adjustment. So the initial threshold determination is critical. However, it is interesting to note that the same simple method of determining threshold behaves quite well in adaptive filtering when comparing our adaptive filtering result with others.

Our training, classifying, and thresholding methods are all well-known methods, but our system behaves better than medians, especially in adaptive filtering. One explanation for this might be the linguistic suite in Oracle Text and knowledge base we used to process documents. The theme vector we get from Oracle Text contains more information than just text token and occurrence frequency in the document. Theme vector have a list of normalized terms. This term normalization could reduce the size of collection thesaurus, and make it easier to match different terms with the same concept. The weight of the theme contains not only the occurrence frequency information, but lexical information. In conclusion, the combination of these linguistic functionalities and appropriately engineering some well-known learning methods are believed to make our system successful.

Table 1: Adaptive and batch filtering result with T10U optimization. The numbers in the parathesis are the median value for all participants.

| Run label | Run type | Optimi-zation | Precision (median) | Recall (median) | T10SU (median) | F-beta (median) |
|---|---|---|---|---|---|---|
| oraAU082201 | adaptive | T10U | 0.538 (0.462) | 0.495 (0.213) | 0.291 (0.137) | 0.519 (0.273) |
| oraBU082701 | batch | T10U | 0.556 (0.618) | 0.353 (0.293) | 0.249 (0.247) | 0.450 (0.448) |

**Table 2:** Routing result. The number in the parathesis is the median value for all participants.

| Run label | Run type | Mean average precision (median) |
|---|---|---|
| oraRO082801 | Routing | 0.104<br>(0.082) |

## 2. Question Answering based on Information Retrieval and Information Extraction

Questions can be classified into pre-defined categories. Typical categories are: person names, organization names, dates, locations (cities, countries, states, provinces, continents), numbers, times, meaning of acronyms and abbreviations, weights, lengths, temperatures, speed, manner, duration, products, reasons etc.[7][8]

Information extraction (IE) techniques allow us to extract lists of semantic categories from text automatically[9], such as person names, organization names, dates, locations, duration, etc., which are subsets of the whole pre-defined question categories. If a question category is covered by IE, finding the locations of answer candidates becomes easier: the task remains is to rank the list of answer candidates extracted by IE. Otherwise, a number of heuristics are employed to locate the answer candidates and rank them.

**Overview of Oracle Q/A system**:
Our Q/A system consists of three major components shown in figure2: (1) question processor (2) sentence ranking (3) answer extraction.

**Question Processor:**
Its role is to: (a) classify a question into a list of pre-defined semantic categories (b) extract content words from a question and send them to Oracle to retrieve relevant documents.

To classify a question, the first step is to determine its question type. The following wh-words are used to determine the question types: *who, why, where, whom, what, when, how much money, how much, how many, how* (rich, long, big, tall, hot, far, fast, large, old, wide, etc.).

A list of heuristics will help to map the question types to the pre-defined semantic categories:
(1) who is (was) "person name" => occupation
(2) other "who" types => personal name
(3) how rich, how much money, how much + VBD(VBP, VBZ, MD) => money expression
(4) other "how much" types => number
(5) how hot (cold) => temperature
(6) how fast => speed
(7) how old => age
(8) how long => period of time or length
(9) how big => length or square-measure or cubic-measure
(10) how tall (wide, far) => length

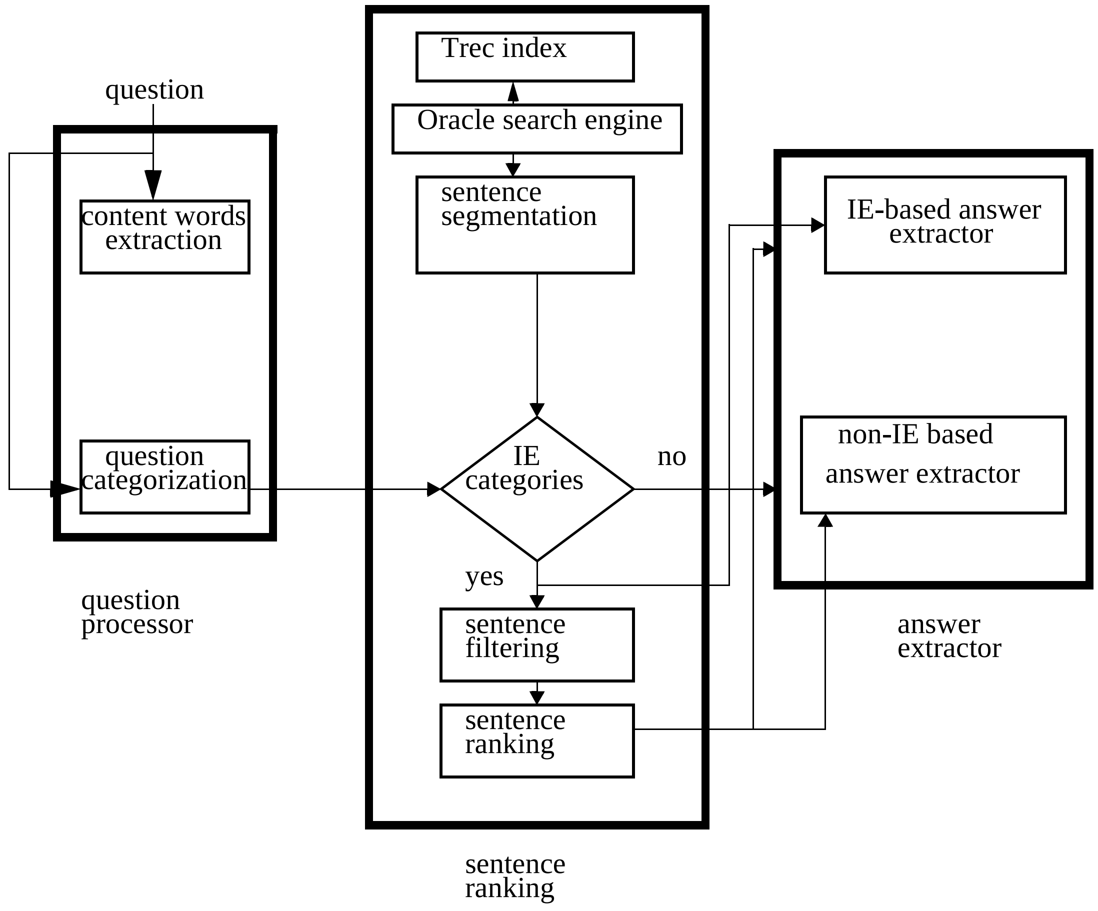

Figure 2: Architecture of the Oracle Q/A System

A complicated problem is to map the question type "what" to its semantic category. Here, a part-of-speech (POS) tagger is used to assign the most appropriate part-of-speech for each word in a question based on the contextual information [10]. The head noun of the first noun phrase in a question is used to decide its semantic category. For example, "What costume designer decided that Michael Jackson should only wear one glove?" The head noun of the first noun phrase is "designer". Using WordNet's lexicon [11], one finds that "designer" is a person, so, the semantic category of this question is "person name". If the head noun of the first noun phrase in a question is a stop word, then, the head noun of the second noun phrase is used to decide the semantic category. For example, "What was the name of the first Russian astronaut to do a spacewalk?" The head noun of the first noun phrase is "name" (a stop word), so, the head noun of the second noun phrase "astronaut" is used to decide the semantic category. Similarly, WordNet's API can tell that its semantic category is "person name".

When extracting a list of keywords from a question, our principle is to extract all content words, but ignore all non-content words. The distinction between these two types of words is that content words should appear in the relevant documents, but non-content words should not appear in the relevant documents. At lease, stop words and stop phrase (such as: how much, what time, what country) belong to non-content words. Furthermore, a list of heuristics is helpful to distinguish content words from non-content words. For example, "What is the length of coastline of the state of Alaska?", and "What is the Illinois state flower?" Word "state" is a non-content word in the first question, but a content word in the second question. Removing non-content words as many as possible makes retrieved documents more focusing on the subject topic of the question and is very helpful for extracting right answers from retrieved documents.

**Sentence Ranking:**
After the query processor extracts a number of content words from a question, two queries are formulated: one uses proximity operator “near" with maximum span size 25 to connect these words, the other uses “accum" operator to connect them. Near opearator find all query terms within specified span. Documents are ranked based on the frequencies and proximity of query terms in the document. Accum (accumulate) operator finds documents matching one or more query terms. Documents are ranked based on the sum of weights of the terms matched and frequency of the terms in the document. The first query has higher priority than the second one, because “near” operator always retrieves more relevant documents, but usually, the number of documents retrieved by “near” is not big enough, so, “accum” query is used to supplement it. Oracle Text retrieves a list of relevant documents (60 documents in trec10) based on the two queries. Then, the relevant documents are broken into paragraphs, the paragraphs are segmented into sentences. According to our experiments, it is suitable to extract long answers (250 bytes) from ranked paragraphs, but to extract short answers (50 bytes), the paragraphs must be further segmented into sentences.

Ranking the segmented sentences is based on the following information: (1) the number of unique content words in a sentence (2) tf and idf of each content word (3) total number of content words in a query (4) the smallest window size which contains all the unique content words in the sentence.

Our information extractor (IE) has two modules: one used for sentence filtering, the other used for answer extraction (IE-based answer extractor). If the semantic category of a question is covered by the IE, the IE is used for sentence filtering. Only selected sentences which satisfy the IE, are the candidates of the sentence ranking. For example, if the semantic category of a question is "person name", only the sentences which include at least one person name will participate the sentence ranking, all the rest of sentences are filtered out from answer extraction, because they do not include answers of the question. The IE was also integrated with sentence segmentation algorithm. The standard sentence delimiters are "?!.", followed by one or more spaces, then followed by a word whose first letter is a capital letter. There are many exceptional cases, such as Mr. Steve, St. Louis. The IE could recognize these exceptional cases, and guarantee the success of the sentence segmentation.

**Answer Extraction:**
After the sentences are ranked, top five of them are used to extract the answers. From previous description, our IE only covers a subset of the whole semantic categories. If the answer type of a question belongs to the subset, it is easy to extract answers using the IE. Otherwise, we concluded a number of heuristics, which help to extract answers. The sentence ranking algorithm can find the smallest window in a sentence, which contains all the content words in the sentence. This window divides the sentence into three parts: (1) the words in front of the window, (2) the words after the window and (3) the words inside of the window. According to our observation, the priorities of the three parts are (1) (3) (2). We further observed that in (1) and (3), the words closer to the windows have higher priority than others. Based on these observations, we picked up certain percent of words from each part of the sentence according to their priorities to form the final answers.

**Other Linguistic Processing:**
(1) acronyms and abbreviations: like other advanced search engines, our system also does limited automatic query expansion, mainly for queries with acronyms, abbreviations, etc. It expanded (a) acronyms of geographical terms, such as "U.S. = United States", "N.C. = North Carolina" (b) abbreviations of organization names, such as "YMCA = young mens christian association", "NBS = national bureau of standards"

(2) stemming: Oracle's search engine does not use Porter's stemmer. Our stemmer is more conservative, which obtains good precision, may hurt recall a little bit. To remedy this problem, extra stemming was

added in rare situations. For example, "When did Hawaii become a state?", the main verb was stemmed as "$become".

(3) Information Extractor (IE): an information extractor was created over the last few months to recognize (a) person names (b) organization names (c) dates (d) number (e) locations (f) money expression (g) time (h) temperature (I) speed (j) weight (k) length (l) square measure (m) cubic measure (n) age, etc.

**Performance Evaluation:**
A question answering system was created based on information retrieval and information extraction. Our study shows that traditional IR technique are not only useful to rank documents, but also to rank paragraphs and sentences. Finding the smallest window from a sentence which contains all the content words in it, is very helpful to extract answers when its semantic category is not covered by the IE, the window size is also an important factor to decide the sentence rank.

The following table shows the evaluation result provided by NIST for our system

| | strict | lenient |
|---|---|---|
| NIST score | 0.477 | 0.491 |
| % of correct answers | 60.77% | 62.60% |
| % of correct first answers | 40.04% | 40.85% |

The current (Oracle 9i) knowledge base is designed for information retrieval; for Q/A track, we found it necessary to expand the lexicon to cover wh-focus ontological facets.

## 3. Web Track
As preparation, we investigated the TREC-10 web task using TREC-9 web track documents and queries. We also attempted to productize lessons learnt from our participation in Trec8 adhoc manual task. A set of different collections including TREC Web and Adhoc collections helped us in our effort to formulate generic techniques applicable across domain. Due to resource constraints, we were unable to work on Trec10 web track. Here we summarize our findings based on older collections.
Our experiments in link analysis using Oracle intranet data indicate that link analysis adds little value to intranet search. Link analysis is a technique that helps bring order to an unorganized collection lacking central authority (such as web) by using popularity measure. A organized intranet will have clearly defined authorities for different subject matters.

IDF weighting used in tf-idf scoring is not very effective when the collection is pretty large (a couple of million documents) and number of terms in the queries is pretty high. If the queries are free-text queries, IDF weighting fails to distinguish between important and unimportant terms. Weighting techniques which weight terms inversely proportional to a factor of the frequency ratios (x times as rare terms get y times as much weight) seem to perform better in this situation. We saw significant improvement in R-precision by adopting this technique.

As the number of documents increases, the number of distinct score values supported by a system becomes important. Until recently Oracle Text used 100 distinct integers in the range of 1 to 100 for scoring. We found that allowing a million distinct values improves system IR quality computed in average precision by improving tie splitting. Even though number of relevant documents retrieved did not increase very significantly (about 3-4%), average precision increased by 10-15% (for example, Trec9 web track average precision improved from 0.11 to 0.125).

Using Trec8 and Trec9 collections, we identified a few simple flaws in our system which have been removed. On average, recall has increased by about 25% and precision at 10 has improved by more than 50%. We ran this out-of-box automatic system against TREC-8 adhoc task. Oracle TREC-8 manual task submission received an average precision score of 0.42. Out of 50 benchmark queries, performance (number of relevant retrieved) is tied for 10 queries, 20 won by manual and 20 won by automatic.

## References

[1] Oracle Technet Text Homepage (*http://technet.oracle.com/products/text*)

[2] K.Mahesh, J. Kud, and P. Dixon, Oracle at Trec8: A Lexical Approach, *in Proceedings of the Eighth Text Retrieval Conference (TREC-8)*, 1999.

[3] C.D. Manning and H. Schutze, Foundations of Statistical Natural Language Processing, the MIT press, 2000.

[4] S. E. Robertson and K. Sparch Jones, Relevance weighting of search terms, *Journal of the American Society for Information Science,* **27**, 129-146, 1976.

[5] K. Sparck Jones, S. Walker and S.E.Robertson, A probabilistic model of information retrieval: development and status, *Technical Report TR446 Cambridge University Computer Laboratory,* 1998.

[6] D. D. Lewis, R. E. Schapire, J.P. Callan and R. Papka, Training algorithms for linear text classifications, in SIGIR'96.

[7] Dan Moldovan, Sanda Harabagiu. Lasso: A tool for surfing the Answer Net. In the *Proceedings of the Text Retrieval Conference (TREC-8),* 1999.

[8] Sanda Harabagiu, Dan Moldovan. Falcon: Boosting Knowledge for Answer Engines. In the *Proceedings of the Text Retrieval Conference (TREC-9),* 2000.

[9] Rohini Srihari and Wei Li. Information Extraction Supported Question Answering. In the *Proceedings of the Text Retrieval Conference (TREC-8)*, 1999

[10] Eric Brill, Some Advances in Transformation-Based Part of Speech Tagger. In the *Proceedings of AAAI*, 1994

[11] G.A. Miller, WordNet: A Lexical Database. *Communication of the ACM*, **38**, 39-41, November 1995