IIIT Hyderabad at TAC 2008

Vasudeva Varma **Prasad Pingali** **Rahul Katragadda**
Sai Krishna **Surya Ganesh** **Kiran Sarvabhotla**
Harish Garapati **Hareen Gopisetty** **Vijay Bharath Reddy**
Kranthi Reddy **Praveen Bysani** **Rohit Bharadwaj**

Contact: vv@iiit.ac.in
Search and Information Extraction Lab.
IIIT Hyderabad, India.

Abstract

This paper describes our participation at TAC 2008 in all the three tracks. For the Summarization Track we introduced two major features. First, a feature based on Information Loss if we don't pick a particular sentence. Second, a language modeling extension that boosts novel terms and penalizes stale terms. During our post-TAC analysis we observed that a simple sentence position based summarizer leads to better short summaries than most official runs submitted this year. In the Opinion QA and Summarization Track for the rigid list questions, we have added some additional features to handle opinion expressed in the question. and for the squishy list questions in Opinion QA and Summarization Track, we leveraged on our existing Summarization engine and used a classification based approach to both finding opinionated sentences and also the polarity of the opinions. Finally, for the RTE track we explored a simple graph partition matching based approach.

Part I
Summarization Track

1 Update Summarization

The key to update Summarization is a real world setting where a user needs to keep track of a hot topic, continuously at random intervals of time. There are a lot of hits on the hot topic and lots of documents are generated within a short span. The updates on the topic need to filter out redundant information, while preserving the informativeness of the content. In the current work, we approach sentence extractive Summarization under the assumption that user has already gone through previous document cluster(s). In this paper, we see the update task as an additional redundancy module within the overall Summarization problem.

2 Language Modeling Approaches to Summarization

A statistical language model, or more simply a language model, is a probabilistic mechanism for generating text. Language modeling based approaches have been famous in text Summarization literature since the early work by Luhn (Luhn, 1958). Recently, (Nenkova et al., 2006) has shown reasonable success in multi-document text Summarization using 'just' unigram language models. We use the Probabilistic Hyperspace Analogue to Language (pHAL) as the language modeling mechanism.

2.1 Probabilistic Hyperspace Analogue to Language (pHAL)

From the model of (J et al., 2005) a Hyperspace Analogue to Language model constructs dependencies of a word w on other words based on their occurrence in the context of w in window size k, in a sufficiently large corpus. We use the pHAL model, and use the relevance based Language Modeling approach as followed by (J et al., 2005) for sentence scoring. The pHAL, probabilistic HAL is a natural extension to HAL spaces, as term co-occurrence counts can be used to define conditional probabilities. The pHAL can be interpreted as, "given a word w what is the probability of observing another word w' with w in a window of size K".
$pHAL\,(w'|w) = c \times \frac{HAL(w'|w)}{n(w)\times K}$

The sentence scoring mechanism is based on this model that has been built. Assuming word independence, the relevance of a sentence S can be expressed as,

$$P\,(S|R) = \prod_{w_i \in S} P\,(w_i|R) \approx \prod_{w_i \in S} P\,(w_i|Q)$$

$$P(S|R) = \prod_{w_i \in S} \frac{P(w_i)}{P(Q)} \prod_{q_j} pHAL(q_j|w_i)$$

$$\approx \prod_{w_i \in S} P(w_i) \prod_{q_j} pHAL(q_j|w_i) \quad (1)$$

2.2 Language Modeling Extension

While generating update summaries, whether in the case of *Single Document Summarization* or in *Multi-Document Summarization* the idea is to be able to present information that the reader has not already seen. Looking at it the other way, it could be considered as suppression of information that the user has already seen. In the context of language modeling, if we construct models that represent the document(or document collections) then the corresponding models for the new stream of data should be modified based on the information that has already been seen, and known to be of importance. It is interesting to note that, query relevance doesn't change. However, a considerable topical shift might occur in the new stream. To accommodate this topical shift and to avoid redundancy in the update-summary, we build a background aware language model that penalizes thematic features that occur more frequently in the background than in the new stream.

2.2.1 Signature Terms

Topic signatures as defined by Lin and Hovy (2000) are a set of related terms that describe a topic. Lin and Hovy achieve their purpose by collecting a set of terms that are typically highly correlated with a target concept from a preclassified corpus such as TREC collections. We try to approximate the same using the dataset at hand. For each topic, if we consider cluster A as irrelevant set($\tilde{R}$) and cluster B as relevant set(R) we obtain terms that are relatively more important for cluster B than for cluster A. In this work we report on the utility of this method for term selection to aid context adjustment for Update Summarization.

The document set is pre-classified into two sets A and B. Assuming the following two hypotheses:

$$\textbf{Hypothesis 1}\ (H1) : P(B|t_i) = \quad p \quad = \quad P(B|\tilde{t_i}) \quad (2)$$

$$\textbf{Hypothesis 2}\ (H2) : P(B|t_i) = p1 \neq p2 = \quad P(B|\tilde{t_i}) \quad (3)$$

H1 implies that relevancy of a document is independent of t_i, while H2 implies that the presence of t_i indicates strong relevancy assuming $p1 \gg p2$. And the following 2-by-2 contingency table:

	R	$\tilde{R}$
t_i	O_{11}	O_{12}
$\tilde{t_i}$	O_{21}	O_{22}

Where O_{11} is the frequency of term t_i occurring in cluster B, O_{12} is the frequency of term t_i occurring in cluster A, O_{21} is the frequency of term $\tilde{t_i} \neq t_i$ occurring in cluster B, O_{22} is the frequency of term $\tilde{t_i} \neq t_i$ occurring in cluster A.

Assuming a binomial distribution:

$$b(k; n, x) = \binom{n}{r}\ x^k\ (1-x)^{(n-k)}$$

then the likelihood for *H1* is:

$$L(H1) = b(O_{11}; O_{11} + O_{12}, p)\ b(O_{21}; O_{21} + O_{22}, p)$$

and for *H2* is:

$$L(H2) = b(O_{11}; O_{11} + O_{12}, p1)\ b(O_{21}; O_{21} + O_{22}, p2)$$

The $\log \lambda$ value is then computed as follows:

$$\begin{aligned}
&= \log\ \tfrac{L(H1)}{LH2} \\
&= \log\ \tfrac{b(O_{11};O_{11}+O_{12},p)\ b(O_{21};O_{21}+O_{22},p)}{b(O_{11};O_{11}+O_{12},p1)\ b(O_{21};O_{21}+O_{22},p2)} \\
&= ((O_{11}+O_{21})\log p + (O_{12}+O_{22})\log(1-p)) - \\
&\quad (O_{11}\log p1 + O_{12}\log(1-p1) + O_{21}\log p2 \quad + \\
&\quad O_{22}\log(1-p2))) \qquad (4)
\end{aligned}$$

This term λ is called Dunning's likelihood ratio (Dunning, 1993). Dunning suggests that λ is more appropriate for sparse data than χ^2 for hypothesis testing and the quantity $-2\log\lambda$ is asymptotically χ^2 distributed. And hence we can use χ^2 distribution table to look up $-2\log\lambda$ value at specific confidence level.

2.2.2 Approach

Our approach is motivated by the fact that the necessary update may be seen as giving more weight to important sentences. Since, importance is now a factor that is also dependent on the previous utterances on the topic, we should either increase the weight of important terms in this cluster or decrease the weights of the words that are more important in the previous clusters. In any of the approaches we first need to identify signature terms (Lin and Hovy, 2000) of the respective cluster with respect to previous clusters. It may be observed that Summarization for the first cluster is a simple query-focused Summarization problem, and we do not consider it here.

The primary algorithm consists of the following steps:

1. For the first cluster, generate summary using the actual Summarization algorithm, say S(x).

2. For each of the next clusters

   (a) Compute Language model of previous clusters, say LM(A).

   (b) Compute Language model of current cluster, say LM(B).

   (c) Generate Signature terms of previous cluster and current cluster, say T_A and T_B respectively.

   (d) Modify the Language model of current cluster, *LM(B)*, by reducing the importance of co-locations of T_A and/or boosting the importance of co-locations of T_B.

   (e) Generate summary based on the modified Language Model.

2.3 Context Adjustment

Let **T** be a set of Signature terms extracted as described earlier. Signature terms could be utilized in adjusting the language model as follows.
$\forall\ w_i \in \mathbf{T}$,

$$\forall w_j \quad \overline{pHAL}\,(w_j|w_i, B) = pHAL\,(w_j|w_i, B) + pHAL\,(w_j|w_i, A) \quad (5)$$

$$\forall w_j \quad \overline{pHAL}\,(w_j|w_i, B) = pHAL\,(w_j|w_i, B) - pHAL\,(w_j|w_i, A) \quad (6)$$

The update based on Eq. (5) is used to boost co-occurrences of novel terms in Cluster B, while update based on Eq. (6) penalizes co-occurrences of signature terms of previous clusters.

3 Query Independent Scoring

We used two scoring mechanisms for Query Independent Scoring, one QIScore and the other a feature based on Information Loss (explained in 3.2).

3.1 QIScore

We also used the query independent scoring mechanism(QIScore) (Pingali et al., 2007). The query independent scoring is based on a contrastive analysis of the given document set D with a randomly chosen document set.

3.2 Sentence Representation Capability

We use information loss as a feature to measure sentence representation capability. We consider sentence and document cluster as different text units and Represent them with probabilistic distribution of terms. We then measure how bad a sentence probability distribution is in modeling the documents distribution using relative entropy. By doing this we are able to capture the amount of information loss in picking a sentence as a representative of whole document cluster.

$$S(D) = \sum_w P(w|S) log \frac{P(w|S)}{P(w|D)}$$

$S(D)$ is amount of information loss when we pick a sentence S as a Representative of document cluster D. The contribution of any term w in sentence information loss can be defined as how much we lose information by assuming the term is drawn from the sentence model instead of the document model.

The sentences with minimum information loss are considered to be more capable in presenting the content of document set.

For generic summary, a sentence importance is measured in terms of its capability to represent the relevant document set D as compare to a random document collection $\hat{D}$.

$$Sentence_{score} = \frac{S(\hat{D})}{S(D)}$$

In case of update summary we consider the sentence important if it is more capable in presenting the content of current document set as compare to a set of previously read documents.
When $D = A$, $\hat{D} = 100$ random news articles.
Otherwise if $D = B$, $\hat{D} = A$.

4 Post Sentence Ranking

Following simple features were used after sentences are obtained in ranked order.

Date Anaphora Once sentences are ranked and the best sentences are identified after eliminating redundancy, we perform a shallow Date Anaphora for relative years and months. Eg., 'last year' would be annotated with 2004, assuming the document is dated 2005. Such simple heuristic seems to be improve readability.

Information Filter and Redundancy Filter A simple information filter that eliminates sentences that contain too many stopwords was implemented. A ratio of number of content words to total number of words needed to be above a predetermined threshold, for the sentence to be include in summary. For summary of cluster B a set of 'n' top ranked sentences from cluster A were used for comparison. Relative Entropy of a sentence with already picked sentences was also used as redundancy measure.

	ROUGE-2	ROUGE-SU4	pyramid score	linguistic responsiveness	overall responsiveness
RUN 1	0.08122	0.11777	0.287	2.396	2.438
RUN 2	0.09520	0.13387	0.331	2.375	2.604
RUN 3	0.07156	0.11295	——	——	——

Table 1: Official TAC Results for cluster A

	ROUGE-2	ROUGE-SU4	pyramid score	linguistic responsiveness	overall responsiveness
RUN 1	0.07219	0.11312	0.246	2.333	1.979
RUN 2	0.79110	0.11999	0.245	2.667	2.167
RUN 3	0.06019	0.10317	——	——	——

Table 2: Official TAC Results for cluster B

Sentence Length Filter Sentences over a threshold number of words were eliminated. This would help remove all the errors due to sentence boundary identification and also eliminate highly descriptive sentences. This could be done in two ways:

1. Thresholding on the total number of words.
2. Thresholding on content words.

5 Experiments

This section describes each of the submitted run and their performance. Results are reported separately for cluster A and cluster B, in tables 1 and 2.

Run 1 Run 1 uses two features, the HAL feature used in (Pingali et al., 2007) and the feature based on Information Loss(Section 3.2). For cluster A, a linear combination of the two features were used while for cluster B only the Information Loss based metric was used in generating the summary. Date Anaphora and Information Filtering was applied along with a sentence length threshold of 30 total words was applied for this run.

Run 2 Run 2 used two features, the HAL feature used in (Pingali et al., 2007) extended as shown in Section 2.3 and the Query Independent Scoring(QIScore) shown in Section 3.1. Following table illustrates the impact of context adjustment on PHAL feature.

	ROUGE-2	ROUGE-SU4
PHAL	0.08190	0.12208
PHAL+ Adjustment	0.08369	0.12395

Table 3: Impact of Context Adjustment on PHAL

Run 3 Run 3 is same as Run 1, except the sentence length threshold being set to 18 words.

6 Evaluation and Results

The results based on the intrinsic and extrinsic evaluations performed at TAC are shown in Tables 1 and 2. Our Run 2 performed well above the median performance based on content evaluations, while performing around the median for linguistic and overall responsiveness. During Post TAC experiments with a little bit of parameter tuning, we obtained a better performance shown as Run 2*[1] in Table 4.

	ROUGE-2	ROUGE-SU4
RUN 2*	0.09310	0.13122

Table 4: Post-TAC RUN 2

7 Post-TAC Experiments

Sentence Position. Sentence Position has been well studied in Summarization research since its inception, early in edmundson's work (Edmundson, 1969). Recently, it has been widely used in learning based approaches (Toutanova et al., 2007; Shen et al., 2007). The University of Ottawa has organized the pyramid annotation data such that for some of the sentences in the original document collection(those that were picked by systems participating in pyramid evaluation), a list of corresponding content units is known (Copeck et al., 2006). We utilized this data to analyze which portions of documents were most sentences being picked-up from, and which of those portions were being most content responsive.

Approach Based on our analysis on the data from DUC 2005, DUC 2006 and DUC 2007 we created a simple heuristic feature. The feature gives most importance to all the lead sentences, more so for first sentence in a document. For large documents(total

[1]RUN 2* : RUN 2 with a little bit of parameter tweaking.

	ROUGE-2	ROUGE-SU4	pyramid score
cluster A	0.08987	0.1213	0.3432
cluster B	0.09319	0.1283	0.3576

Table 5: ROUGE 2, SU4 Recall and modified Pyramid Scores for Sentence Position based feature.

sentences greater than 15), trailing sentences are scored better than the rest in middle. However, trailing sentences for short documents get higher scores than those of larger documents.

As can be observed, only the first sentence of all documents could end up comprising the summary. This is OK until we don't get redundant information in the summary. Hence we also used a simple unigram match based redundancy measure that doesn't allow a sentence to be appended to the summary if the sentence matches any of the already selected sentence in at least 40% of content words in it. We also dis-allow sentences greater than 25 content words.

It could be seen that the algorithm is very simple. It uses a very naïve approach that has been used for decades of Summarization research. There could be datasets that could be generated where this approach fails. However, for newswire data that we currently use for Summarization Test sets, this approach is more than promising. This is especially the case for short summaries and a special case would be update Summarization where it genuinely outperforms most other systems that participated in the task.

Table 5 reports Average ROUGE recall and pyramid scores[2] of the proposed algorithm. Data has been presented separately for each cluster.

8 conclusions

In this part we report on our experiments for the Summarization track's update Summarization task. We used a feature based on Information Loss and a language modeling extension to PHAL and have shown that these features perform reasonably well compared to the state-of-the-art. We also show some results from our preliminary post-TAC analysis, where we find that a simple heuristic feature based on *Sentence Position* is able to generate summaries on par with the state-of-the-art update Summarization systems.

[2]Pyramid Annotations for this experiment were done by one of the authors, who volunteered for pyramid annotations during DUC 2007.

Part II
Opinion Track

1 Introduction

TAC 2008 opinion task focuses on Question Answering (QA) and Summarization tracks. The task aims at mining opinions from blog posts for rigid list and squishy list questions. In opinion QA, each target contains both rigid list and squishy list questions. Our goal here is to provide precise answers from the Blog06 corpus. In opinion summarization task, each target contains squishy list question(s) and a set of relevant blog posts . Our goal here is to extract opinions from this relevant document set and produce summaries for each squishy list question.

In general, rigid list questions (Ex : Which countries would like to build nuclear power plants?) have named entities as answers. Current QA systems typically include four major steps: 1) Question Classifier, 2) Passage Retrieval, 3) Answer Extraction and 4) Answer Ranking. *Question Classifier* analyzes the question to determine the answer type. *Passage Retrieval* searches for paragraphs in the document collection that are likely to contain the answer. *Answer extraction* extracts a list of candidate answers from the retrieved passages. *Answer Ranking* ranks the list of candidate answers to determine the final answer. We have used the same architecture for rigid list questions with some additional features to handle the opinion expressed in the question in each step.

Squishy list questions (Ex : What features do people like in vista?)ask for strings containing the answer. Answering Squishy list in TAC08 QA track is some what similar to descriptive type of question answering, So we have decided to use the summarization system we designed for "DUC 07 multi-document summarization task" which perform well while answering descriptive types of questions on news articles. Our approach to handle these questions involves the following three steps: 1) Question Analysis, 2) Sentence Classification, and 3) Summarization. Question Analysis determines the polarity (positive/negative) of the question. Sentence classification extracts the sentences which are opinions with the same polarity as that of the question. Finally summarization, the system we are using has two features, one being query independent feature and the other being query dependent feature. We added one more feature to handle opinions. For this we made use of Senti Wordnet (Attardi and Simi, 2006), an online lexical source for opinion words.

This part consists of six sections. First we introduce the task, then we describe our work on processing the blog corpus in section 2. Section 3 describes our approach for rigid list questions and section 4 describes our approach for squishy list questions. Section 5 describes the submitted runs for both the tasks. Section 6 discusses the results obtained for each task. We conclude in section 7.

2 Preprocessing the data

In opinion QA track answers must be retrieved from Blog06 corpus. Though we have used only top 50 documents for each target (Blog data in this paper refers to the top 50 document set but not the total Blog06 corpus) there are a significant number of challenges to extract required content from each document such as encoding issues, posts from different blog domains, identifying post author etc.

In the blog data different character encodings were used. As the first step we converted all characters to UTF-8 encoding. Posts from different blog domains are present in the blog data. Each domain uses a template to generate blog HTML pages. So, we extracted the required content from these blog posts by reverse engineering the templates used to generate these pages. There are fifty targets in TAC08 QA test data. For each target they provided fifty most relevant documents. So, on the whole there are 2,500 posts which were processed to extract required content. Majority of the posts are from Blogger, WordPress, Typepad, Movable Type domains. Their respective numbers in the blog data are shown in table 6.

Some of the rigid list questions ask for authors of the posts who expressed positive/negative opinion about a target. So, to handle this type of questions we extracted authors of the posts using different regular expressions for different domains listed in table 6. For the remaining 500 blog posts, only content extraction by removing HTML tags and author extraction was performed. For author extraction, we have used a generalized regular expression to handle most of the remaining blog posts. Among the remaining 500 blog posts we were able to extract author from 290 posts.

The test data given for opinion summarization also belongs to Blog06 corpus, so we used the same approach for cleaning the blogs for opinion summarization as we did for opinion QA.

3 Approach for Rigid List Questions

Our system architecture includes the following steps : question classification, post retrieval, answer extraction and answer ranking. Question classifier de-

Domain	No of Documents
Blogger	1160
WordPress	337
TypePad	210
Movable Type	96

Table 6: Distribution of documents in some of the blog domains.

termines the answer type and polarity for the given question. Support Vector Machines (SVM) (Zhang and Lee, 2003) and naive Bayes classifiers (Yang and Liu, 1999) are used to determine the answer type and polarity of the given question respectively. From the sample questions, we have observed that for the questions having answer type as PERSON, some of them are referring to the person names in the posts and the rest are referring to the authors of the posts. To handle both the cases, we mapped the answer type PERSON to a new answer type "AUTHOR-PERSON".

In traditional QA systems where passage is considered as a retrieval unit on which answer extraction techniques are performed. But in the case of blog data, author conveys his opinion through the entire post and his opinion cannot be determined from a single passage within the post. So, we considered post as a unit of retrieval. We used Lucene for both indexing the posts and searching for relevant posts. All the relevant passages returned by Lucene are classified using nave Bayes classifier which classifies the post either relevant or non relevant. A post is said to be relevant if its polarity is same as the polarity of the question and non relevant if its polarity is different from the polarity of the question. Finally the set of relevant passages returned by the classifier were used for answer extraction.

For Answer Extraction we divided the answer types into two super classes.

1. AUTHOR-PERSON
2. Named Entities

In the first case, answer to the given question is the list of authors from all the relevant posts and the names of the persons within the posts. As the author names were previously extracted during Preprocessing, all the authors who are associated with the relevant passages form a list of possible answers to the question. For extracting the person names from the posts, we used Stanford Named Entity Recognizer (SNER) (Finkel et al., 2005) and for the second case also we used SNER to extract possible answer candidates for the given question.

The list of answer candidates obtained from answer extraction is ranked using relevance features. We have two such features:

1. Blog post relevance to the question
2. Blog post relevance to the polarity of the question

The first feature ranks the answer candidates which are extracted from the most relevant posts higher and the second feature ranks the answer candidates based on the polarity of the blog post . We used a weighted linear combination of these two features to compute the final score for each answer candidate. Finally, the ranked list of answers based on candidate scores is provided as answer to the given question.

4 Approach for Squishy List Questions

In both Opinion QA and Opinion Summarization these squishy list questions focuses on the descriptive answers of the target. The approach for these questions will be similar for both the task with some differences in the methods used. The main steps involved in squishy list part of opinion QA and opinion Summarization are:

1. Query analysis
2. Opinion mining and Polarity determination
3. Opinion sentences summarization.

The summarization system we designed for DUC07 has two features to rank sentences for generating summary. These features are 1) Query dependent feature and 2) Query independent feature.

Query dependent feature boosts the sentences which are most relevant to the given query. The query independent feature boosts the most informative sentences in the given text using KL divergence method. Using these two features, we have achieved good results in summarization which is a kind of descriptive QA in DUC 2007. Both Opinion QA and Opinion Summarization are also similar to descriptive QA with minute difference of giving some importance to opinions of the sentences. This factor motivated us to use the above two features along with

our opinion feature for summarization. We did our experiments on two variances of this opinion feature which we will present in detail below.

Query analysis focuses on predicting the polarity class (either positive or negative) for the query. For example a query like What features do people like in vista? focuses on positive aspects of product vista. For predicting the polarity class for the query, we framed a set of rules. We have downloaded a list of words which can be termed as a seed list for query words. This list contains a small set of keywords which are classified as positive or negative. We have classified query based on this list of words as positive or negative. In this process we have taken into account the negated verbs also.

Opinion mining and Polarity determination focuses on our interest in mining opinion sentences from non-opinions and determining polarity for opinions in the given input data. We built a two class classifier in two phases one for classifying opinions and non-opinions and other for classifying opinions as positive and negative. The classifier used for this classification in Opinion QA task is Rainbow with naive Bayes method for two runs. In Opinion Summarization task we have used SVM classifier (Ku et al., 2006) with unigrams as features in one of runs and Rainbow using prind - probabilistic indexing method in the other run.

Each sentence will have its opinion and polarity class scores obtained from classifier associated with it. In Opinion QA first run we made use of these scores as a feature along with both query dependent and query independent feature to score sentences. In second run of Opinion QA we have used opinion score as a filter and used only query dependent and query independent feature to score sentences. In Opinion Summarization task, we made use of Senti wordnet, which is a lexical source for opinions as the opinion scoring feature. Senti Wordnet will have a list of words which will be the sources for sentence polarity. Each word which contribute to sentence polarity will have both positive and negative score associated with it. So we used this to score sentences along with query dependent and query independent feature in our first run. In second run, we used scores obtained from classifier using probabilistic indexing method as the opinion feature along with query dependent and query independent features.

5 Description of Runs

Our approach to handle opinion sentences uses two phase classification first the sentence is classified as either opinion or non-opinion(fact) then it is classified as positive or negative. The data used to build these models are described below.

For opinion-non opinion classification we have downloaded IMDB movie review data from the web which is annotated. The data contains 10,000 sentences of opinions and non-opinions. We have used this data as a training data for our opinion, non-opinion classification.

We have collected product reviews form Amazon.com. We collected about 1,30,000 reviews on various products. We did use this as a training data for polarity determination. We have classified the reviews as positive or negative based on the ratings given at the end of each review. Each review which has a rating of 4 or 5 has been termed as positive and others as negative. Finally, we got about 98,000 positive reviews and the rest were negative for training.

Models for opinion QA task were built using navie bayes algorithm, where as models for opinion summarization task were built using SVM-HHM techniques.

5.1 Opinion QA run 1

5.1.1 Rigid List

The same approach we described in section 3 was used to extract the answers for rigid list questions. We used two features to rank the candidate answers. Both the features used to rank candidate answers were given equal priority while ranking the candidate answers.

5.1.2 Squishy List

In addition to the two feature used by summarization system, we used a new feature which boost the opinion sentences. The third feature scores the sentences using the weighted linear combination of opinion/non-opinion score and positive/negative score. We pick either positive/negative sentences based on the polarity class predicted in question analysis module. The opinion score of each sentence is calculated as follows,

$$opinionscore = 0.3 * p(sentence, opinion) + 0.7 * p(sentence, polarityclasspredicted)$$

Final score is the weighted linear combination of all three features. The weights assigned to query dependent, query independent and opinion features are 0.275,0.325 and 0.4 respectivly.

5.2 Opinion QA run 2

5.2.1 Rigid List

The rigid list answers submitted for run 2 were same as those submitted for run 1.

Type	Run 1	Run 2	Best Run	Median of Runs
Rigid List	0.131	0.131	0.156	0.063
Squishy List	0.186	0.165	0.186	0.091
Total	0.164	0.154	0.168	0.093

Table 7: Opinion QA results.

Runs	F-Measure	Coherence	Readability	Responsiveness
Run 1	0.101	2.045	3.545	2.364
Run 2	0.102	2.045	3.545	2.500

Table 8: Opinion Summarization results.

5.2.2 Squishy List

In this run the opinion score computed was used as a filter. Sentences are ranked using the two features we had in the summarization system, and a sentence is picked only if its opinion score is greater than 0.4.

5.3 Opinion Summarization run 1

Each sentence which is an opinion will be classified as positive or negative and the polarity class predicted in the query analysis will be used to pick sentences of that particular polarity class.

Then we ranked the sentences of particular class using Senti Wordnet to determine the opinion score. The final score will be weighted linear combination of all the three features query dependent, query independent, and opinion feature.

5.4 Opinion Summarization run 2

The run 2 of opinion summarization task is some what similar to run the run described in section 5.1.2, except that weight distributions are changed. The weights assigned to query dependent, query independent and opinion features are 0.5,0.3 and 0.2 respectivly.

6 Results

In this section we present the results of our participation in opinion QA and opinion summarization tasks.

The opinion QA task in TAC 2008 has questions distributed over 50 targets. Each target has a minimum of one rigid list and squishy list questions and a maximum of 4 questions of both the types. On the whole there were 90 rigid list and 90 squishy list questions. Answers for these questions must be retrieved from the Blog06 corpus. We have used the top 50 documents for each target distributed by NIST to retrieve answers. The evaluation metrics for rigid list questions is F-measure and for squishy list questions nugget pyramid evaluation is used. We present our results in the Table 7.

In the above table rigid list scores are the average F-measure scores over 90 rigid list questions and Squishy list scores are the average pyramid F scores over 87 squishy list questions and total scores are average per-series scores over 50 series.

The opinion Summarization task has questions distributed over 25 targets and a summary for each target. In addition to the blog posts, there are optional answer snippets provided for this task. We did nt use optional answer snippets in our submission. We present results in the Table 8.

The various entries in the table are Average F-measure with beta value 1, Average score for Grammaticality, Average score for Non-redundancy, Average score for Structure/Coherence (including focus and referential clarity), Average score for Overall fluency/readability, Average score for Overall responsiveness respectively.

7 Conclusion

We described our participation in the TAC 2008 opinion QA track and opinion summarization track. In our approach for rigid list questions, we used both the authors of the posts and person names within the posts as a list of possible answers to the PERSON type questions i.e, we are also adding answer candidates of incorrect answer type to the list of possible answer candidates, which definitely decreases the precision. So, using a mechanism to distinguish the fore mentioned two answer types the performance can be increased. For answering squishy list questions, we added a third feature to boost the sentences based on the opinion score of the sentence. Using this approach we obtained the best performance among all the runs submitted by the opinion QA task participants.

In addition to opinion QA experiments, as a part of opinion Summarization task, we conducted exper-

iments using both SVM and Rainbow text classifier. In rainbow we used Word Association as a feature for the classifier rather than giving exact sentence for training. In ranking the sentences, we have made use of Senti Wordnet along with query independent and query dependent features. We described all the approaches used and description of runs in the above sections. The average F-measure scores for each run over 22 summaries are very low. The reasons for this might be 1.not using the optional answer snippets provided, which may be the cause for low precision, 2.We used all the space for summary (total 7000 characters), as a result many sentences out of context appeared in the summaries and 3. weights were not distributed properly.

Part III
Recognizing Textual Entailment(RTE) Track

1 Introduction

The task consists of recognizing the relation between two given text fragments, namely Text and Hypothesis. The track consists of two way and three way tasks. The three way task is to recognize the relationship between text and hypothesis as:

- ENTAILMENT, if Text entails Hypothesis.
- CONTRADICTION, if Text contradicts Hypothesis.
- UNKNOWN, if the truth of Hypothesis cannot be determined on the basis of Text.

The two way task is to recognize the relationship between Text and Hypothesis as

1. ENTAILMENT, if Text entails Hypothesis.
2. NOT ENTAILMENT, if Text does not entail Hypothesis.

Our approach to this task is to construct the syntactic dependency trees for both text and hypothesis sentences and comparing the nodes of the dependency trees by using the semantic similarity between the two nodes.

2 Method

An overview of our system architecture is shown in the figure 1.

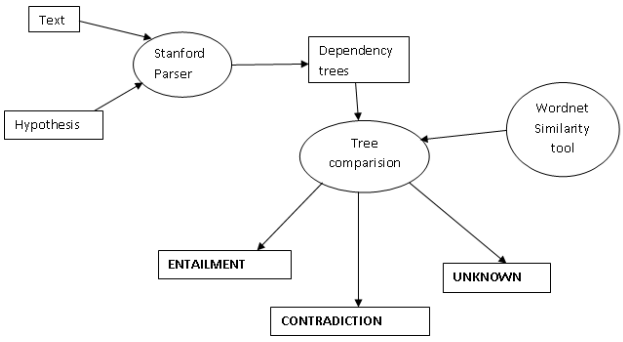

Figure 1: System Architecture for RTE

Our system uses the approach of syntactic tree matching of text and hypothesis syntactic dependency trees (Micol et al., 2007). First we create the syntactic dependency trees for the sentences in both Text and Hypothesis. For constructing dependency trees we used Stanford Parser. We will ignore the irrelevant grammatical categories while constructing the tree. We maintained a list of irrelevant grammatical categories. Some of these categories are Determiners, pre-determiners, post-determiners, clauses, Inflectional phrases, preposition and preposition phrases, auxiliary verbs, complementizers. Now that we have dependency trees for both text and hypothesis, we will check whether the hypothesis tree is embedded in the text. We say that hypothesis tree is embedded in texts tree if all nodes and branches of hypothesis are present in text. For example let 'h' be any node in the hypothesis(we will start with the root node of Hypothesis).

1. For 'h' we try to find a matching node 'x' in the text tree.
   (a) First we check for the direct match of the words, without the use of semantics.
   (b) If we cannot find any match, then we use the Wordnet::Similarity tool (Pedersen et al., 2006) to find semantic similarity between the words. If the similarity value between the two words is greater than 0.8 we consider that they match.
2. Once we find out the matching node 'x' we try to find matching nodes for the children of 'h' in the children of 'x' as described in the previous step.

The above process starts from the root node of hypothesis tree and continues until all the nodes of hypothesis are matched in the text. If we found a matching node for every node of hypothesis, we judge that text entails hypothesis. Hence the result would be **ENTAILMENT**. If we cannot judge entailment from the above step then we send the dependency trees to separate modules, which recognize CONTRADICTION and UNKNOWN.

For handling **CONTRADICTION**, we check for the negation and antonym features(Snow et al., 2006). If hypothesis is embedded in text, but text contains a negation words like not, no, none etc, then we say that text contradicts hypothesis. We check whether the verbs in text and hypothesis are antonyms to each other

For handling **UNKNOWN** case, the following features based on named entities and verbs have been included.

Checking Named Entities. We check whether all the named entities in the hypothesis are present in the text. If at least one named entity is not present in text we would say the result as **UNKNOWN**.

Checking Verbs. If two or more verbs (other than auxiliary) in the hypothesis have no match in the text, then we say that result as **UNKNOWN**.

3 Evaluation

We submitted two runs to RTE4 challenge. We did not rank the pairs based on entailment confidence. We submitted the runs only for the three way task. The following table shows the official results that our system obtained in RTE4 challenge

	Accuracy	
	2-way	3-way
Run 1	0.531	0.309
Run 2	0.529	0.307

Table 9: 2-way and 3-way classification accuracy

For the first run, we took the threshold for similarity score obtained from Wordnet Similarity tool as 0.8 and for the second run, we took the threshold for similarity score as 0.75.

References

Giuseppe Attardi and Maria Simi. 2006. Blog mining through opinionated words. In *Proceedings of TREC 2006, the Fifteenth Text Retrieval Conference.*

Terry Copeck, D Inkpen, Anna Kazantseva, A Kennedy, D Kipp, Vivi Nastase, and Stan Szpakowicz. 2006. Leveraging duc. In *DUC 2006: In the proceedings of Document Understanding Conference 2006.*

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. volume 19, pages 61–74.

H. P. Edmundson. 1969. New methods in automatic extracting. volume 16, pages 264–285. ACM.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

Jagadeesh J, Prasad Pingali, and Vasudeva Varma. 2005. A relevance-based language modeling approach to duc 2005.

Lun-Wei Ku, Yu-Ting Liang, and Hsin-Hsi Chen. 2006. Opinion extraction, summarization and tracking in news and blog corpora. In *Proceedings of the AAAI Spring Symposium on Computational Approaches to Analyzing Weblogs (AAAI-CAAW-06).*

C. Lin and Eduard Hovy. 2000. The automated acquisition of topic signatures for text summarization.

H.P. Luhn. 1958. The automatic creation of literature abstracts. In *IBM Journal of Research and Development, Vol. 2, No. 2, pp. 159-165, April 1958.*

Daniel Micol, O scar Ferrandez, Rafael Munoz, and Manuel Palomar. 2007. Semantic similarity based on syntactic dependency trees applied to textual entailment.

Ani Nenkova, Lucy Vanderwende, and Kathleen McKeown. 2006. A compositional context sensitive multi-document summarizer: exploring the factors that influence summarization. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 573–580, New York, NY, USA. ACM.

Ted Pedersen, Siddhart Patwardhan, and Jason Michelizzi. 2006. Wordnet::similarity - measuring the relatedness of concepts.

Prasad Pingali, Rahul K, and Vasudeva Varma. 2007. Iiit hyderabad at duc 07. In *DUC'07: Document Understanding Conference, 2007.*

Dou Shen, Jian-Tao Sun, Hua Li, Qiang Yang, and Zheng Chen. 2007. Document summarization using conditional random fields. In *the proceedings of IJCAI '07.*, pages 2862–2867. International Joint Conference on Artificial Intelligence.

Rion Snow, Lucy Vanderwende, and Arul Menezes. 2006. Effectively using syntax for recognizing false entailment.

Kristina Toutanova, Chris Brockett, Michael Gamon, Jagadeesh Jagarlamundi, Hisami Suzuki, and Lucy Vanderwende. 2007. The pythy summarization system: Microsoft research at duc 2007. In *DUC 07: In the proceedings of Document Understanding Conference 2007.*

Yiming Yang and Xin Liu. 1999. A re-examination of text categorization methods. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49. ACM.

Dell Zhang and Wee Sun Lee. 2003. Question classification using support vector machines. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 26–32, New York, NY, USA. ACM.