Opinion Summarization Using Entity Features and Probabilistic Sentence Coherence Optimization: UIUC at TAC 2008 Opinion Summarization Pilot

Hyun Duk Kim, Dae Hoon Park, V.G.Vinod Vydiswaran, ChengXiang Zhai
{hkim277, dpark34, vgvinodv, czhai}@illinois.edu
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801

Abstract

This paper is about the TAC 2008 Opinion Summarization pilot participation by the University of Illinois at Urbana-Champaign (UIUC). We mainly explored two ideas: (1) use of entity recognition and parsing to enhance a standard retrieval method for opinion retrieval, and (2) use of a coherence language model to optimize the ordering of sentences in a summary. Our result showed use of entity recognition showed mixed performance results and coherence language model was not as good as heuristic polarity ordering with guiding phrases.

1 Introduction

As more people post their opinions on the Web, the Internet is becoming a popular and dynamic source for opinions these days. Due to the large volume and wide range of data, there is a growing need to summarize opinionated documents. The opinion summarization pilot task of the 2008 Text Analysis Conference (TAC) offers a good opportunity to study and evaluate methods for summarizing opinionated text documents. We participated in this task and studied two problems in opinion summarization: (1) Can we improve sentence retrieval by assigning more weights to noun phrases or entities in the query? (2) How can we optimize the ordering of sentences to increase the coherence of a summary?

The Pilot Summarization Task in TAC 2008 was to summarize the opinions expressed in the blog documents. The data used was BLOG06 corpus that has been used earlier in 2006 and 2007 in many TREC tracks. There was a set of target topics on which the participants were evaluated. For each target, there was a set of questions, usually dealing with the opinion expressed about the target topic. Also, a set of relevant document ids was released for every target. The final goal was to generate a summary of the opinions expressed in the given document set about the particular aspects of the target topic as referred to in the set of questions associated with each target topic. Additionally, there were some sample answer snippets released for every target that participants can optionally use.

Each summary had a restriction on the length that was based on the number of questions per topic. The submitted results were evaluated on how many relevant opinions were covered, the length of the summary generated, and the coherence of the summary.

Our overall method for solving this problem is to break the task into four steps, viz. data preprocessing, sentence retrieval, sentence filtering, and summary generation. Fig. 1 shows an overview of the system. In the data preprocessing step, we eliminate useless tags from the target documents and identify meaningful keywords in a question query. In the sentence retrieval step, we extract relevant sentences from target documents with the query from the previous step. In the sentence filtering step, we filter the retrieved sentences based on the question and the target sentence polarity. We also remove redundant sentences from the list. In the summary generation step, we order retrieved sentences based on polarity and a statistical coherence language model.

Specifically, to retrieve relevant sentences, we used standard language modeling approach and inference network retrieval frameworks. Also, we explored an idea of putting different weights on words in a query question based on linguistic analysis. Since a large portion of questions was asking about people's opinion on some

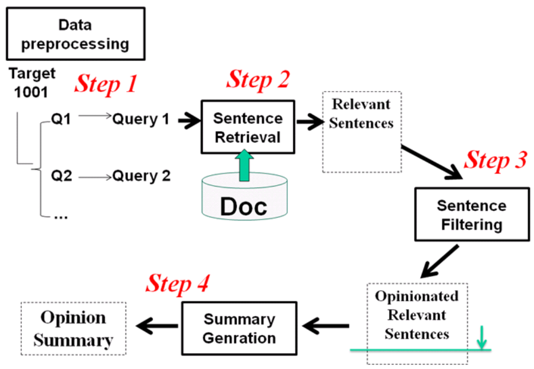


Figure 1: System overview

targets, we hypothesized that named entities and noun phrases should probably have higher weights than other words in the query. Thus we used natural language processing techniques to distinguish named entities and noun phrases, and studied whether putting more weights on them would improve retrieval accuracy.

After retrieving a set of relevant sentences about the target, we used opinion orientation features to further filter out retrieved sentences that do not match the desired opinion polarity. The most important difference between our opinion summarization task and a normal summarization task is that the query question and target documents are opinionated. If the question was asking for positive aspect of one object, the answer also should follow the same orientation. We thus used a heuristic dictionary-based method for predicting opinion polarity, and filtered out sentences which do not match the polarity orientation of the question.

The next step of our method is to select a set of representative sentences by removing redundancy. We used some simple methods to assess sentence similarity and used the maximal marginal relevance (MMR) ranking method [1] to select up to a certain number of sentences that are not redundant with respect to each other.

The last step is to generate a readable coherent summary based on the selected non-redundant sentences. A main question we study in this step is how to optimize the coherence of a summary. We proposed a statistical coherence language model to order sentences. Specifically, we estimate the probability that words u and v occur in a pair of adjacent sentences using some natural text as training data. We can then use this model to compute a coherence score of a candidate ordering of two sentences to be included in a summary. An optimal order of multiple sentences can be generated using a greedy algorithm by maximizing the pair-wise coherence scores. In addition to this statistical method, we also explored using polarity to separate opinions and heuristically adding additional words to improve readability.

We submitted three runs: a polarity based run (UIUC1), a coherence-based run (UIUC2), and a baseline run (UIUC3). Specific information of each run will be explained in the following sections and summary of details is shown in Table 1. None of them used the provided answer snippets because in a real application of opinion summarization we would not have such information. These runs all follow the general method described above with variations in some of the steps. The results of these runs and some follow-up experiments show that increasing the weights of entities and noun phrases in the query improved performance for some queries but could not make significant improvement for some queries. Also, simple statistical coherence optimization did not perform as well as heuristic polarity ordering method with guiding phrases.

In the rest of the paper, we will first describe each step of our method in more detail and then discuss results with our additional experiments.

2 Data preprocessing

2.1 Document processing

Remove tags, special character: We used the Indri toolkit for our experiments. To build an index with the Indri toolkit, we eliminated UNICODE special character which Indri cannot handle.

Extract main contents: In a blog document, we are mainly interested in the main contents of the blog entry. So, in addition to eliminating the HTML tags, we also delete the contents between two tags, if it is too short and does not belong to the main part of the blog. In addition, we also filter out the sections under a javascript.

Sentence splitting: We use a sentence as the basic unit of retrieval. We believe that a sentence is a natural unit for summarization because a sentence is a basic unit that can deliver a complete message and a coherent summary can be composed of sequence of sentences. We applied a Sentence Segmentation tool[1] to each preprocessed document. The Sentence Segmentation tool handles honorifics and initials within names well, so we were able to obtain clean sentence boundary using the tool.

[1] http://l2r.cs.uiuc.edu/∼cogcomp/atool.php?tkey=SS

2.2 Question processing

Common word elimination: Before we retrieve sentences using questions as queries, we removed common words in questions. Because most given questions have similar formats, there are words which are not related to the topic, such as "people", "opinion", and the stopwords. We applied a simple heuristic on the frequency of words: we counted all the words in given questions and we filtered out words having a frequency count more than a threshold $= 3$.

Find more important words: We considered named entity and noun phrase as more important words because named entity and noun phrase can deliver discriminative information compared to common words, and most of questions were asking opinions about a specific entity. We parsed the question to find all the named entities in the question. We built an NE recognizer using Learning Based Java (LBJ)[2] to identify PERSON, ORGANIZATION, and LOCATION entities. LBJ is a modeling language for the rapid development of learning-based systems, designed for use with the Java programming language. Then we found noun phrases in the question. We parsed each question description with a POS tagger[3]. Then, the noun phrases were identified by identifying consecutive nouns. For example, for the sentence "The new mobile phone was released today at the tech expo.", the noun phrases identified are "mobile phone" and "tech expo".

3 Sentence Retrieval

For sentence retrieval, we mainly used Indri toolkit[4]. We used the basic retrieval model, a combination of the language modeling and inference network retrieval frameworks, with term weighting [7]. We considered each individual sentence as a document, and built an Indri index over all sentence-documents. We then retrieved the sentences using the filtered list of words from the questions as queries.

With IndriQuery, we can assign different weights for individual words / groups of words. We built an Indri query using a weighted combination of the named entities, noun phrases and other words from the query to improve the recall. The retrieved sentences formed the initial ranked candidate list. For the run using different weighting (UIUC1), we used 10:2:1 as weights for named entity: noun phrase: other words. Our approach gave named entities a much higher weight compared to other terms and also gave noun phrases a slightly higher weight than other words. The weighting scheme was decided based on preliminary experiments of relevance judgment results from TREC Blog retrieval task. We checked the MAP score for different weighting schemes, such as 2:1:1, 5:2:1, 10:2:1, etc., and chose 10:2:1 because it had the highest score. Although both experiments used the same BLOG06 data, we cannot be sure that this is the optimal weighting for the current task at hand because the evaluation was done on a different query set. It is possible that another combination of weights may improve the performance for this task. In fact, our UIUC1 run did not perform as well as the other run (UIUC2) which uses uniform weighting.

4 Sentence Filtering

4.1 Polarity filtering

We implemented simple dictionary-based polarity decision module. We first obtained lists of words expressing positive and negative sentiments from the following sources: 1) sentiment words[5] and 2) good and bad (positive and negative) sentiment adjectives[6]. We also added some positive and negative words that commonly occur in questions to generate the final positive/negative word list. The following is the complete list of words we added to the existing lists: Positive words: support, like, great, suggest, approve, want; Negative words: negative, complain, complaint, object, oppose. These word lists are then used by the polarity decision module to decide one of four polarities: viz. positively-opinionated, negatively-opinionated, mixed-opinionated, and non- opinionated. For an input text segment, the polarity decision module counts the number of words that overlap with the positive/negative word list. Depending on the number of positively and negatively inclined words in the target text segment, the polarity module decides whether the text segment is positively, negatively, or mixed opinionated. If the positive words and negative words are quite dissimilar, we will classify the post as either positive or negative. If we find a negation in a sentence, such as "not" or "never", we reverse the polarity. If the numbers of positive and negative words are quite similar, then we classify the document as a mixed-opinionated document. If there are no common words between the text segment and the positive/negative list, the text segment is classi-

[2]http://l2r.cs.uiuc.edu/∼cogcomp/software.php

[3]http://l2r.cs.uiuc.edu/∼cogcomp/software.php

[4]http://www.lemurproject.org/

[5]http://eqi.org/index.htm

[6]http://www.keepandshare.com/doc/view.php?u=12894

fied as non opinionated.

We applied the polarity decision module in two steps. First, based on the assumption that the task wants us to summarize "opinions", we simply filtered out all non-opinionated sentences from our initial candidate list. Second, we applied opinion decision module to the question as well, and then based on the question polarity, we selected sentences which have the same polarity as the question. For example, if the question asked for "people's positive opinion about" a particular target, the relevant sentences should be positively opinionated and also related to the target.

Polarity decision module is also used for paragraph division (See Section 5).

4.2 Remove redundancy

After the initial retrieval and filtering based on polarity, there can still be many similar sentences in the candidate list. In such a case, we need to keep only one representative and eliminate others from the candidate list. We used the Text Similarity[7] tool to find pairwise sentence similarity. The tool provide lesk value [6] based on counting common words and phrases between two strings. Phrasal matching gets more scores than single word overlap. Because longer sentences tend to have more matching, the finals scores are normalized by the length of strings.

Based on this similarity score, we applied MMR [1] technique to eliminate redundant sentences. For each sentence s, we calculated similarity score with each sentence ranked higher than s, and if the sentence similarity score of s with any of them is above a set threshold, we removed s from the candidate list. This way all "duplicate" low-ranked sentences are filtered out. For our submissions, we set similarity threshold = 0.8.

4.3 Cut sentence list up to the limit

Finally, we retained all the sentences in the rank order and pruned the list when the total length of sentences reached the summary size limit (7000 non-white spaced characters per question, each target has one to three questions). Among summaries for 22 targets, 14 summaries reached the limit, and the average length of our final summary results was 11080 non-white spaced characters (accumulated over all sub-questions in one target).

[7] http://www.d.umn.edu/∼tpederse/text-similarity.html

5 Summary Generation

5.1 Result post-processing

Based on the preliminary experiment results, we noticed that a few noise phrases such as date and time of blog entry occur frequently in the blog article collection. Since these did not contribute to a blog opinion, we removed them by matching the following simple regular expression.

```
␣Posted␣by:␣([A-Za-z0-9()]+␣)*?\|
\w\w\w␣\d+,␣\d\d\d\d␣\d\d?:\d\d:\d\d
␣[AP]M␣
```

5.2 Paragraph division

Even though we had a good list of relevant sentences as summary, we needed additional structure to make the summary more coherent. We divided the summary into paragraphs based on the questions. The answer to each question was grouped together in one paragraph. Additionally, we needed to distinguish the positive opinions from the negative opinions on a topic. Hence, we further organized the paragraphs by polarity. We kept all sentences with a positive polarity together at the start of the paragraph, followed by all sentences with negative and mixed polarity. This way, we avoided changing the topic of interest and the sentiment orientation too often in the final summary and hence increased coherence.

To make paragraph division more clear and the change of topic more lucid, we also inserted guiding text snippets. At the beginning of each paragraph, we added guiding sentences like "The first question is ... " or "Following are positive opinions ...".

5.3 Statistical coherence optimization

We also experimented with ordering the sentences so as to optimize the statistical coherence. We considered the following simple statistical approach: Suppose $S = \{s_1, \ldots, s_n\}$ is the set of candidate sentences in the summary. Let $c(s_i, s_j)$ be an asymmetric coherence measure between two sentences s_i and s_j. We can then generate a summary by enumerating all the possible orders of these n sentences and choosing the one that maximizes the overall "pairwise coherence". That is, we seek for an ordering of sentences $s'_1, \ldots, s'_n$ that maximizes the overall sum $c(s'_1, s'_2) + c(s'_2, s'_3) + \ldots + c(s'_{n-1}, s'_n)$.

Because enumerating all the permutations is infeasible, we use a greedy algorithm to approximate it. We try to build a sequence of sentences, one pair at a time.

Among all the pairs, we selected the one that has the highest coherence score. Let it be $\langle s_i, s_j \rangle$. This forms the first unit in the sentence sequence. Then, we find the highest coherence pair among the ones that can be used to extend the sentence sequence in either direction. In other words, if the current sentence sequence is $\langle s_i, \ldots, s_j \rangle$, then among all the pairs of the form $\langle \cdot, s_i \rangle$ or $\langle s_j, \cdot \rangle$, we selected the one with the highest coherence score and added it to the sentence sequence.

The coherence measure $c(s_i, s_j)$ can be defined in many way. In our experiments, we defined it as the average pointwise mutual information of a word in the first sentence, s_i, and a word in the second sentence, $s_j (= s_{i+1})$. For every word pair $\langle u, v \rangle$, we first find $p(u, v)$, the pointwise mutual information of word u in a sentence s_i and word v in the following sentence s_j. Since this is sensitive to the order of sentences, $p(u, v)$ is asymmetric. We used simple additive smoothing.

$$p(u, v) = \frac{adjCount(u, v) + 0.001}{count(u)count(v) + 1.0} \quad (1)$$

where $adjCount(u, v)$ is defined as count of u and v in two adjacent sentences in order. (u in s_i and v in s_j)

Then, we define the asymmetric coherence measure $c(s_i, s_j)$ over two sentences s_i, s_j, as

$$c(s_i, s_j) = \sum_{u \in s_i, v \in s_j} \frac{p(u, v)}{|s_i||s_j|}$$

where $p(u, v)$ is as defined above and estimated using all adjacent sentences in the training data. Computationally, we get all the words in adjacent sentences s_i, s_{i+1} and accumulate the number of times we observe each combination of words (u, v) with u coming from sentence s_i and v from sentence s_{i+1} over the training data. We then normalize the counts to get the pointwise mutual information $p(u, v)$.

This approach is similar to the Mirella Lapata's method [5] with some differences in the definition of the scoring function and the technique used to search for the best order.

6 Evaluation

6.1 Submission

We submitted three runs, labeled UIUC1, UIUC2, and UIUC3. To evaluate the effect of different techniques, we applied different combination of techniques detailed above, for each run. They have been summarized in Table 1.

Techniques	UIUC1 Polarity	UIUC2 Coherence	UIUC3 Baseline
Filter non-opinionated sentences	Off	**On**	**On**
Filter sentences having same polarity as questions	**On**	Off	Off
Paragraphs divided on question	**On**	Off	Off
Paragraphs divided on polarity (pos/neg/mixed)	**On**	Off	**On**
Statistical coherence optimization	Off	**On**	Off

Table 1: Techniques enabled in the submitted runs. On indicates active and Off indicates inactive features.

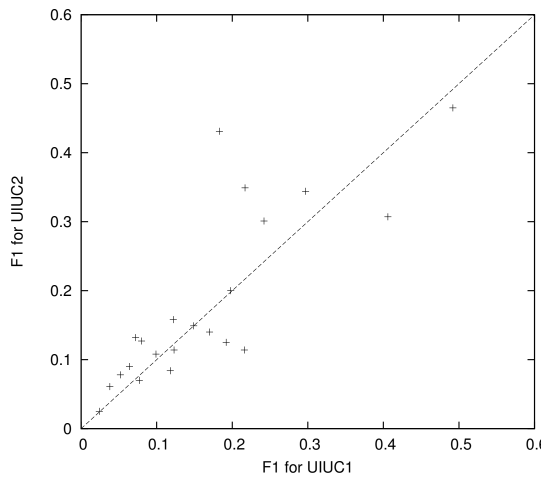


Figure 2: Scatter-plot comparing F1 scores of UIUC1 with UIUC2

6.2 Evaluation results and Discussion

6.2.1 Official evaluation results

Among the submissions, two runs, UIUC1 and UIUC2, were evaluated in the official TAC 2008 evaluation phase. The official results are summarized in Table 2.

	Recall	Precision	F, with $\beta = 1$
UIUC1	0.309181818	0.125090909	0.165045455
UIUC2	0.390136364	0.124818182	0.180545455

Table 2: Recall, Precision, and F with $\beta = 1$

Table 2 shows the average scores over 22 summaries based on the pyramid nugget method. Among the two evaluated runs, UIUC2 showed better recall, and both runs have similar precision. The F-score showed UIUC2 had better performance. Fig. 2 shows a scatter-plot of the F1 score for all queries over the two runs.

	F-Score	Grammaticality	Non-redundancy	Structure/Coherence	Fluency/Readability	Responsiveness
UIUC1	18	31	30	12	17	21
UIUC2	15	20	19	30	33	14

Table 3: Rank among all runs (Total: 36 runs)

	F-Score	Grammaticality	Non-redundancy	Structure/Coherence	Fluency/Readability	Responsiveness
UIUC1	6	15	15	5	6	8
UIUC2	3	9	10	15	16	4

Table 4: Rank among runs that did not use answer-snippet (Total: 19 runs)

Pyramid score based evaluation is mainly evaluated over the contents. On the point of content selection alone, more sophisticated techniques were applied to UIUC1, such as different weighting for named entities and noun phrases, and sentence filtering using polarity matching with the question. However, the performance evaluation result showed that methods of UIUC1 were not effective as those of UIUC2.

We hypothesize that two factors can contribute to degraded retrieval performance in UIUC1: (1) using high weights for named entity/noun phrases might add noise and harm the retrieval performance, or (2) if the polarity decision module made a wrong decision, it could degrade the performance. Since there can be many implicitly opinionated expressions in a blog document, the polarity decision module did not work as well as it worked with other better-formatted text corpora.

Table 3 and Table 4 show the average score rankings for our runs in six evaluation subcategories. The ranks mentioned in Table 3 is on among all the evaluated runs and the ranks in Table 4 is among the submitted runs that did not use the answer-snippets. Since we did not use the provided answer-snippets, the results in Table 4 are more meaningful.

Our F-score ranking among all runs was about average. However, among runs without using answer-snippet, our runs showed pretty good performance especially on the point of content retrieval. As mentioned above, F-score measure from pyramid nuggets explains the performance of content selection. Because the answer-snippet information provides significant clues for relevant information, it is more appropriate to compare content selection performance among runs that did not use answer snippet. The run, UIUC2, was ranked 3^{rd} in F-score among runs that do not use answer-snippets. Therefore, we can say that the content retrieval module in our system has pretty good performance.

In both runs, the non-redundancy scores are low. This is because we applied redundancy removal procedure to sentence list based on each question individually. We extracted sentences for each question separately and did not merge them all for one target until the final summary creation. This led to presence of duplicate sentences in one target topic summary which was generated by merging relevant sentences of all the questions in the target. To remove duplicates which came from different question's relevant sentence list, we need to apply redundancy removal again after merging sentences from each question.

UIUC1 obtained high score in the structure/coherence criterion. Among runs without using answer-snippet, UIUC1 was ranked 5^{th}. UIUC1 adopted both question and polarity paragraph division techniques. Moreover, guiding text snippets for each paragraph are also enabled in UIUC1. Compared to UIUC1, UIUC2 had low structure/coherence score. This suggests that our statistical coherence optimization technique did not work as well as heuristic polarity ordering with guiding phrases. Using a greedy algorithm, which sees only at the next sentence, limits the algorithm from obtaining clues from non-neighboring sentences, and this could harm the coherence score considerably.

Finally, for the overall responsiveness score, UIUC2 had better score than UIUC1. In terms of ranking, UIUC1 showed average performance and UIUC2 showed fairly good performance. Among the runs that did not use answer-snippets, UIUC2 was ranked 4^{th}.

6.2.2 Additional experiments on entity feature weighting

To examine the relationship between content retrieval performance and weighting on entity features such as named entities and noun phrases, we performed some additional experiments. There has been previous work on trying differential weighting for named entities. Hassel's experiment on Swedish text [2] showed the importance of careful usage of named entity feature in automatic summarization. He pointed out that high weighting on named entity can degrade information retrieval

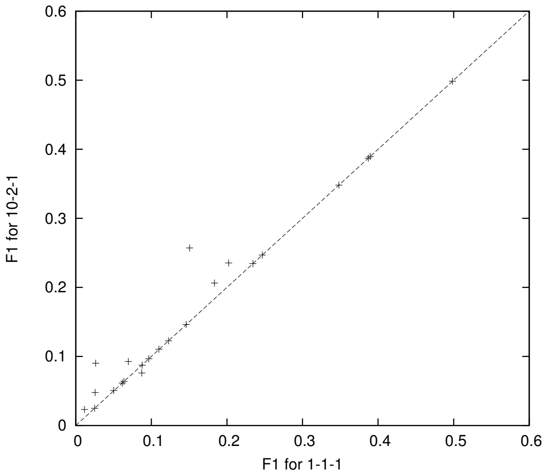


Figure 3: Scatter-plot comparing F1 scores of retrieval runs with 1-1-1 and 10-2-1 weighting schemes

performance because it can cause reference errors and prioritize elaborative sentence over sentence having general information.

In our experiments, we evaluated the effect of weighting named entities by computing how well the systems retrieved relevant nuggets. We calculated precision, recall, and F-scores based on the pyramid nuggets on the intermediate sentence extraction results before applying the polarity module so as to exclude the effect of the polarity module. Instead of applying our redundancy check, we eliminated clearly redundant sentences by removing adjacent repeating sentences. For each sentence on the ranked sentence list, we checked presence of any of the nuggets. Since the nugget text given as part of pyramid nuggets' data is a description rather than the exact nugget, it does not exactly match with the string in the extracted sentences. We decided the presence of nuggets by calculating similarity between the sentence and provided nugget text. After stemming and removing stopwords from the target sentence and nugget text, we used Text Similarity tool to count the overlap of words. If the similarity score between the extracted sentence and nuggets in the same target is above the certain threshold, we say that the sentence has the corresponding nugget.

The similarity score threshold was tuned with the evaluation results of our TAC submission, which showed which nuggets were found in our submission result. Based on the preliminary experiment with the first two target results, we tuned the similarity threshold in increments of 0.05 and decided the threshold to be 0.35.

We experimented with 2 sample settings of weights; in Fig. 3 and Fig. 4, "1-1-1" indicates equal weights to all components and "10-2-1" indicates that the named entities were weighted 5 times higher than noun phrases, which were weighted twice as much as other words in the query. Fig. 3 shows the scatter plot comparing the F1 scores of retrieval runs with 1-1-1 and 10-2-1 weighting schemes. We see that for many queries, the F-scores are higher when the named entities and noun phrases are weighted higher. However, the improvement is not uniform across queries and does not result in any significant improvement. Further, for very few queries, the performance degrades when named entities are weighted higher. Fig. 4 shows the F-scores of nuggets found in two sample queries, 1022 and 1027. For these queries, the non-uniform weighting scheme performed better than the uniform scheme. The plots show that more sentences with relevant nuggets are retrieved when the named entities are given higher weights. Typical summary lengths are 4000 characters. The summaries were truncated based on the maximum allowable length in TAC (i.e. $7000\times$ number of questions in a target). Based on these experiments, it seems that weighting named entities higher than other words improves performance to some extent, but the overall improvement is not very significant.

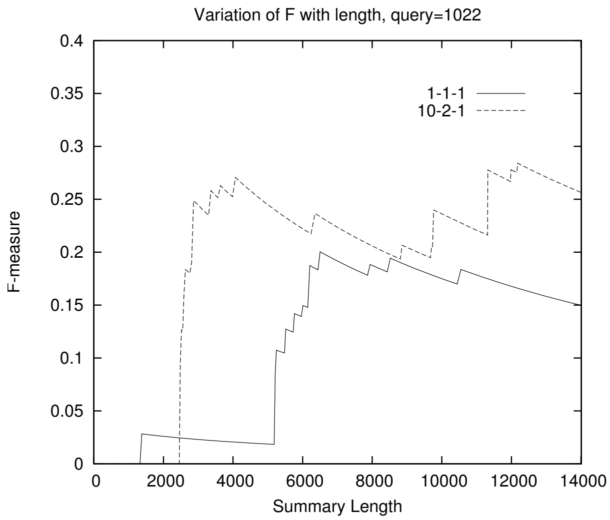
6.2.3 Additional experiments on polarity module

To check the performance of our polarity module, we executed additional experiments with tagged data set which was used Minqing Hu and Bing Liu's previous work [3, 4]. They used 14 product review from Amazon[8], and sentences in data set has manually generated tags indicating features and their sentiment (positive and negative). We collected all the positive and negative sentences, without segregating product and feature-specific reviews, and checked if our polarity module can classify those sentences correctly. Because Amazon reviews are free-formatted text and they are posted by users freely, we can assume that they have similar characteristics as blog posts. Table 5 shows the classification results. The true classification rate is 38% and the exact opposite misclassification rate is 16%. These results show that the performance of polarity module was low and can be a reason for us filtering out some useful sentences in our final summary.

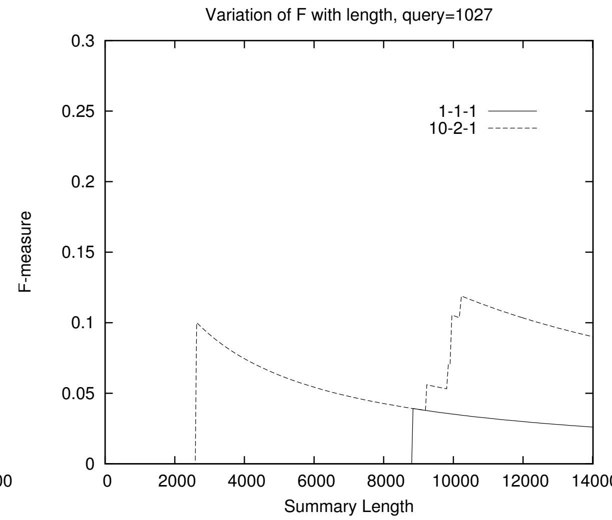
6.2.4 Additional experiments on coherence optimization

To check the performance of coherence optimization, we conducted additional experiments on coherence op-

[8] http://www.amazon.com



(a) Query 1022



(b) Query 1027

Figure 4: Variation of F-scores with length

Classification result	Positive	Negative
NonOpinionated	1063	598
Positive	1363	371
Negative	383	412
Mixed	296	210
Total	3105	1591
Exact Match	1363/3105=0.44	412/1591=0.26
(1363+412)/(3105+1591)=0.38		
Exact Opposite	383/3105=0.12	371/1591=0.23
(383+371)/(3105+1591)=0.16		

Table 5: Polarity module performance evaluation. (number of sentences)

timization module. We can measure the performance of coherence module by comparing the order of sentences given by the coherence module with "ideally-ordered" sentences. Since there is no "ideal" order in a summary, we used the individual blog documents for testing. We assumed that the sentence order in original document is ideal. Among the given relevant blog documents, some part was used for training the coherence module and the rest was used for testing. Table 6 shows detailed data set usage.

	Number of sentences
Given data set	52383
Training set	38561 sentences (73.6%)
Test set	13822 sentences (26.4%)

Table 6: Coherence optimization experiment data set.

In addition to the basic optimization which we applied to the submitted run, we tried different coherence functions. Instead of using pointwise mutual information for obtaining $p(u,v)$ (Equation 1), first, we tried to use strict joint probability which is defined as

$$p(u,v) = \frac{adjCount(u,v)+1}{count(allWordPairs) + (size(dictionary))^2} \tag{2}$$

Second, because we suspected that the bias of pointwise mutual information caused the low performance of the original algorithm, we tried to use general mutual information which is defined as

$$p(u,v) = p(u,v)log\frac{p(u,v)}{p(u)p(v)} + p(\neg u,v)log\frac{p(\neg u,v)}{p(\neg u)p(v)}$$

$$+p(u,\neg v)log\frac{p(u,\neg v)}{p(u)p(\neg v)} + p(\neg u,\neg v)log\frac{p(\neg u,\neg v)}{p(\neg u)p(\neg v)} \tag{3}$$

where

$$p(u,v) = \frac{c(u,v)+0.25}{N+1},$$

$$p(\neg u,v) = \frac{c(\neg u,v)+0.25}{N+1},$$

$$p(u,\neg v) = \frac{c(u,\neg v)+0.25}{N+1},$$

$$p(\neg u,\neg v) = \frac{c(\neg u,\neg v)+0.25}{N+1},$$

$$N = c(u,v) + c(\neg u,v) + c(u,\neg v) + c(\neg u,\neg v),$$

For unseen pairs,

$p(u, v) = 0.5 * Min$(seen pairs in training)

$(\neg u, v)$ means "u does not exist in sentence i and v exist in sentence $i + 1$".

We also ran the optimization experiments after first excluding frequent words, then excluding rare words. Words having high frequency tend to be stopwords which are not related to sentence ordering. Some rare signal words such as "first" or "second" may be a significant clue for deciding sentence coherence. However, we decided to ignore such rare clues. Based on this hypothesis, we excluded frequent and rare words when we train out data.

For the evaluation measurement, we used a very-conservative strict pair matching formulation. We set the evaluation score to be (number of correct sentence pair / number of total sentence pair). The range of the used measurement is 0 to 1, and 1 means exact same order to the ideal document.

u	v	p(u,v)
the	the	1.75E-003
to	the	1.22E-003
the	to	1.21E-003
the	of	1.14E-003
the	and	1.14E-003
of	the	1.13E-003
and	the	1.12E-003
a	the	1.06E-003
the	a	1.06E-003
...	...	...

Table 8: Top ranked p(u,v) of strict joint probability.

Table 7 shows the experiment results. For the basic score, we obtained 0.056063 which is 3.5 times larger than that of random ordering, 0.01586. However, it is still low score to expect fine coherent ordering because the range of measure is $[0, 1]$. This result shows that the performance of current probabilistic coherence optimization module is low. This corresponds with the low coherence score results of run UIUC2.

Among different coherence functions, pointwise mutual information showed the highest performance. Excluding some words in sentence ordering also showed positive aspects. By eliminating frequent words, we could obtain a higher score in strict joint probability and mutual information. When eliminating stopwords, we could obtain performance increase in all three functions. Similar to stopwords, eliminating connecting words and transitive words which have common words with stopwords also helped to improve performance generally. However, because connecting words and transitive words contain key words for sentence ordering such as "first", "second" or "next", performance improvement was lower than that when eliminating stopwords. Even performance decreased in strict joint probability when eliminating transitive words.

We can infer the superiority of pointwise mutual information came from its effective handling on frequent words. When u and v are frequent, because the denominator of pointwise mutual information increases ($count(u)$ $count(v)$ +1.0, Equation 1), it penalizes $p(u, v)$. However, the denominator of strict joint probability ($count$ $(allWordPairs) + (size(dictionary))^2$, Equation 2) is fixed regardless of frequency of u and v. In the same way, because mutual information add up all the cases that u and v exist or not, N is fixed regardless of frequency of u and v (Equation 3). Therefore, when eliminating frequent words, we could see performance improvement in strict joint probability and mutual information, while we could not see it in pointwise mutual information which already penalized frequent word by itself. When eliminating stopwords, performance improved in all functions including pointwise mutual information. This is because stopwords list contains not-meaningful words which were not frequent in the training set. The performance improvements of strict joint probability and mutual information when eliminating stopwords are still larger than that of pointwise mutual information. We can confirm our guess in Table 8 showing many stopwords which should not be used significantly to determine sentence order are highly ranked in strict joint probability.

7 Conclusion

Our submission to the opinion summarization pilot task at TAC 2008 gave us a great opportunity to try our various techniques and ideas to tackle the task. On the contrary to our initial expectation, the run have higher weighting on named entity and noun phrases, UIUC1, showed lower retrieval performance. Our experiments showed that differential weighting of named entities and noun phrases helped retrieve more relevant sentences and improve the nugget score in some cases. By evaluating our polarity module, we showed the possibility that using imperfect polarity decision model is the reason that UIUC1 had a degraded performance in content selection. The run adopted statistics coherence optimization module showed low coherence evaluation. The experiment results on the performance of statistical optimization also showed that sentence ordering based on only

Selection of training words	Coherence score		
	Strict Joint Probability	Mutual Information	Pointwise Mutual Information
No omission	0.022259	0.041651	$\mathbf{0.056063}$
Omitted 5% most frequent words (counts > 33)	0.031389	0.049498	0.051460
Omitted 10% most frequent words (counts > 11)	0.027013	0.046103	0.045725
Omitted 20% most frequent words (counts > 6)	0.020448	0.034785	0.032219
Omitted 40% most frequent words (counts > 2)	0.019769	0.021882	0.022259
Omitted 50% most frequent words (counts > 1)	0.018109	0.020599	0.019694
Omitted 95% least frequent words (counts < 33)	0.022259	0.032898	0.044065
Omitted 90% least frequent words (counts < 14)	0.022033	0.032823	0.049045
Omitted 80% least frequent words (counts < 6)	0.021203	0.037048	0.054931
Omitted 60% least frequent words (counts < 2)	0.022259	0.041802	0.056591
Omitted stopwords[9]	0.036445	0.054554	$\mathbf{0.057119}$
Omitted non-stopwords	0.018185	0.022863	0.020750
Omitted connecting words[10]	0.022335	0.043235	0.057421
Omitted non-connecting words	0.018788	0.019241	0.017958
Omitted transitive words[11]	0.020825	0.042405	0.056968
Omitted non-transitive words	0.019090	0.017279	0.019694

Table 7: Coherence optimization result. For the baseline, the coherence score of a random ordering was 0.01586.

pointwise mutual information between words is not effective. However, by comparing coherence ordering results using strict joint probability and mutual information, we could learn using pointwise mutual information was the better choice than others. Moreover, additional experiments with selected words based on frequency of words showed the possibility of better performance by tuning characteristics of training set.

As part of future analysis, we can consider use less strict coherence measure and different ordering methods. We also need to examine more factors affecting retrieval and sentence ordering performance. It will also be instructive to analyze what weighting scheme would perform well based on the query.

Based on the result analysis, we can make better solution by using summary structuring methods and content selection methods from UIUC2 with additional redundancy removal after merging sentences from each question. We also need to improve the precision of the polarity decision module by preparing better-selected positive/negative word lists or applying different decision methods.

References

[1] J. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336, 1998.

[2] M. Hassel. Exploitation of named entities in automatic text summarization for swedish. In *Proceedings of NODALIDA'03- 14th Nordic Conference on Computational Linguistics*, 2003.

[3] M. Hu and B. Liu. Mining and summarizing customer reviews. In *Proceedings of International Conference on Knowledge Discovery and Data Mining*, 2004.

[4] M. Hu and B. Liu. Mining opinion features in customer reviews. In *Proceedings of National Conference on Artificial Intellgience*, 2004.

[5] M. Lapata. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the 41st Meeting of the Association of Computational Linguistics*, 2003.

[6] M. Lesk. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from a ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, 1986.

[7] H. T. Trevor Strohman, Donald Metzler and W. B. Croft. Indri: A language-model based search engine for complex queries. In *Proceedings of the International Conference on Intelligent Analysis*, 2005.