Description of the LIPN Systems at TAC2008: Summarizing Information and Opinions

Aurélien Bossard, Michel Généreux and **Thierry Poibeau**
Laboratoire d'Informatique de Paris-Nord
CNRS UMR 7030 and Université Paris 13
93430 Villetaneuse — France
{firstname.lastname}@lipn.univ-paris13.fr

1 Introduction

The Text Analysis Conferences (TAC) offer a unique occasion to show innovative approaches to text summarization. As a first incursion into this new research area, LIPN participated in the following tasks of TAC 2008: the Update Summarization task and the Opinion Summarization task. LIPN systems obtained good results for Opinion Summarization; however, we are confident that there is room for improvement, especially for the systems that did not perform so well on the Update task.

This paper gives a technical description of the systems developed. Algorithms and results are then briefly discussed for the different tasks we participated in.

In the first section, we describe the LIPN1-Opinion system for Opinion Summarization, which is based on an adaptation of the MEAD system (Radev *et al.* 2004) integrating a sentiment analysis component. We then describe a series of other systems based on an original clustering-based summarization algorithm called CBASES, which has been applied to the Update as well as to the Opinion Summarization tasks.

2 LIPN1-Opinion: Summarizing Opinionated Blog Posts

We propose here an approach which combines traditional summarization with sentiment detection. Our system's overall architecture is not without resemblance to MEAD (Radev *et al.* 2004), a technique that is centred on the computation of cluster centroids to evaluate a sentence relevance to a set of clustered documents. Although our approach relies only incidentally on centroids, our system has preserved a few other recognizable aspects of MEAD, such as a certain number of feature computations and re-ranking. Our system departs mainly from MEAD in a number of ways: the inclusion of sentiment analysis to account for opinionated texts, computing of a score for additional relevant features, post-processing (merging) sentences as well as experimenting with similarity measures other than the usual *cosine*.

2.1 How the runs were produced

There are three elements which serve as input to the summarizer: the XML *targets/questions/documents* file associates each *target:question* to a set of relevant blog posts, the posts themselves, and a few optional snippets of answers to each queries drawn from the blog posts and provided by TAC.

Cleaning Each XML formatted blog post goes through boilerplate stripping as well as a basic screening to discard noisy input. Sentences with a ratio:

number of frequent words/total number of words

below a given threshold (0.35) were deemed too noisy and discarded. Frequent words are the one hundred most frequent words in the English language which on average make up approximately half of written texts (Fry, Kress, & Fountoukidis 2000).

Computing Features Once we have clean data, we are in a position to compute a number of relevant features for each sentence. There are nine distinct features: *sentiment*, *isLongestSentence*, *similarityWithTarget*, *similarityWithQuery*, *similarityWithFirstSentence*, *similarityWithSnippets*, *centroid*, *length* and *position*. Details of the computation of each feature can be found in section 2.2.

Clustering This step aims at creating clusters on the basis of the test data. The test data provided a list of blog posts for each target in which answers to queries could be extracted and summarized. Consequently, each cluster gathers blog posts associated with a unique *target:query* combination as provided with the data[1], resulting in a total of forty-eight clusters.

[1] For example, there were ten documents in the cluster 1001.2

Query Classification The clusters were further grouped in two categories according to the sentiment expressed or sought after in the query. Two broad categories were considered: positive and negative. Therefore, a two-class SVM classifier was developed and trained on the manually tagged queries from the training data provided earlier in TAC. The idea behind the grouping of clusters is to improve summaries by selecting sentences having the same opinionated polarity as the query.

Summarizing The selection of relevant sentences for summarization is based on a weighted sum of feature values. To avoid including very short sentences in the summary, sentences of length (in words) below a predefined threshold (10) were given a score of zero[2]. Otherwise, the scoring formula for *n* features is as follows:

$$\sum_{i=0}^{n} w_i * f_i \qquad (1)$$

where w_i represents the weight of feature f_i. For TAC, weights values are shown in table 1.

Feature	Pos. Query	Neg. Query
sim...Snippets	+100	+100
sim...Query	+40	+40
sim...Target	+20	+20
sentiment	+20	-20
sim...FirstSent.	+10	+10
centroid	+10	+10
position	+10	+10
isLongestSent.	-10	-10

Table 1: Feature weights

We can see that an important weight was attributed to sentences with a high degree of similarity with the snippets. Long sentences were marginally penalized. A negative *sentiment* weight for negative queries can be explained by the fact that the value of a negative sentiment feature was -1.

Re-ranking Before being selected for the final summary, highly scored sentences go through a re-ranking phase to avoid repetition. Starting from the highest ranked sentences, the re-ranker only inserts a highly scored sentence into the summary if the summary current compression rate has not yet fall below its compression requirement and the sentence is not too similar (is below a threshold of similarity) to any of the sentences already in the summary. The side effect is that the final compression ratio may end up above requirement. We have set our target compression rate to 25 sentences.

Merging At this point we have 48 distinct *target:query* clusters which have been summarized. The final task is to merge clusters with the same *target*. This process involves ranking sentences according to their original position in the summary to eliminate redundant (that is too similar) sentences in the same fashion as we did in reranking. The final result is 25 summaries corresponding to the 25 targets (reduced to 22 by the examiners).

2.2 Feature computation

This section presents the computation methodology for each of the eight sentence features.

sentiment We have attempted to give each sentence a meaningful positive or negative polarity by using a supervised approach where a set of labelled documents were used for training two SVM classifiers, one for categorising queries and one for categorising sentences from blog posts. Queries had rather regular structures with a few repetitive patterns that could be learned rather easily using only a few training queries. The queries provided in an early TAC release of a sample queries served this purpose. The learned classifier was then used to classify each query of the test data. The classification of sentences from the blog posts presented a rather more challenging task, given the variable length, structures and domain targeted. All these obstacles were arguments to adopt the following safer, although not necessarily more accurate, strategy: a document classifier was built to classify each blog posts, in which each sentence was attributed the same polarity as the blog post itself. The document classifier was trained on the self-annotated movie reviews corpus[3] freely available online. The assumption behind this simplification is that blog posts tend to express views which are rather unbalanced, leaning most of the time towards one particular pole of the sentiment scale, which leads to sentences sharing the same polarity as the overall post. Negative sentences were attributed value -1 and positive sentences value +1.

isLongestSentence The longest sentence for each post is attributed a value of one, zero otherwise.

similarityWithTarget The lexical/semantic similarity (see section 2.3) of a sentence and the target is attributed a value between zero and one.

similarityWithQuery The lexical/semantic similarity of a sentence and the query is attributed a value between zero and one.

similarityWithFirstSentence The lexical/semantic similarity of a sentence and the first sentence of a post is attributed a value between zero and one.

similarityWithSnippets A list of snippets were provided for each target. A snippet was a piece of information responding to the information need of queries and the snippets provided by NIST contained all the snippets used in

for the target *Carmax* and the query *What motivated negative opinions regarding purchasing a car from CARMAX?*

[2] After careful examination of the snippets (as defined in section 2.1) size, a more advisable upper limit would have been much less than ten words.

[3] `http://www.cs.cornell.edu/People/pabo/movie-review-data/`

the evaluation. Each sentence was attributed a value of similarity between zero and one corresponding to the similarity with the most similar snippet.

centroid The centroid value measures how significant a sentence is with regards to other sentences in a text, based on the TF*IDF technique. The centroid of sentence *s* is calculated as follows:

$$centroid_s = \sum_{word} tf_{word,s} * idf_{word,s} \tag{2}$$

The final centroid value for a sentence is normalized to a value between zero and one by dividing each centroid by the highest centroid in the text.

position This value reflects how close to the top (sno = 1) of the text sentence *s* is located.

$$position_s = \sqrt{1/sno_s} \tag{3}$$

2.3 Similarity measure

A summarization system based on sentence selection relies heavily on its ability to compute precisely and effectively a large number of similarity values between sentences. One such measure has enjoyed a great deal of popularity, because it is conceptually simple and computationally attractive: the *cosine* measure of similarity represents sentences as weighted vectors of TF*IDF and similar vectors are simply those separated by a small angle. The only extrinsic resource needed is a good lexicon of IDF values for each word. TAC is a good opportunity to look at two slightly more refined alternatives in computing sentence similarity involving word order and the use of a lexical database[4]. As opposed to the *cosine* measure that models sentences as a bag of words, the Levenshtein distance (Ristad & Yianilos 1998) does consider word order by attributing a penalty to operations aiming at transforming one sentence into another: insertion, deletion and replacement of words. In the basic version of the method, each operation is worth a penalty of one. In the more advanced version, the penalty for replacement is commensurate with the continuous value of semantic dissimilarity between two words, based on a measure of relatedness (Pedersen, Patwardhan, & Michelizzi 2004) of two words by comparing the glosses of each in Wordnet. Other approaches that use Wordnet and the Levenshtein distance to classify paraphrases can be found in (Brockett & Dolan 2005).

To provide a comparison as objective as possible, we have computed similarity values for the Microsoft Research Paraphrase Corpus[5] which consists of 4076 paraphrases (1323 fair paraphrases FPs and 2753 good paraphrases GPs). The paraphrases in the corpus were constructed on the basis of news sources from the web, and each paraphrase was annotated by humans indicating whether each pair captures a paraphrase/semantic equivalence relationship. However, unlike blog posts, which consist mostly of opinionated texts, the paraphrases from the corpus are by and large factual in nature, which limits the scope of the conclusion we can make about the use of such similarity measures for our task.

The corpus was completed with 4076 pairs of phrases not paraphrase of each other (NPs). Pairs of phrases 4, 5 and 6 are examples of non-paraphrases, fair and good paraphrases respectively. The similarity values computed using the three methods are also provided with each example.

(4) *A 1991 Florida straw poll helped catapult a little-known Bill Clinton to national prominence. / Two Democrats, Sen. Charles Robb of Virginia and Wendell Ford of Kentucky, voted with the 40 Republicans.* (levenshtein = 0.00, levenshtein+wordnet = 0.03, cosine = 0.00)

(5) *Under the settlement, Solutia will pay $50 million in equal installments over a period of 10 years. / Under the agreement, Solutia will pay $50 million and Monsanto will pay $390 million.* (levenshtein = 0.41, levenshtein+wordnet = 0.22, cosine = 0.32)

(6) *EU ministers were invited to the conference but cancelled because the union is closing talks on agricultural reform, said Gerry Kiely, a EU agriculture representative in Washington. / Gerry Kiely, a EU agriculture representative in Washington, said EU ministers were invited but cancelled because the union is closing talks on agricultural reform.* (levenshtein = 0.19, levenshtein+wordnet = 0.21, cosine = 0.96)

We can see that the corpus includes paraphrases using different word order (see 6) and/or semantically related terms (see 5, *settlement* vs *agreement*).

We have measured similarity for each of the three groups (NPs, FPs and GPs) using the three methods described above (cosine, levenshtein, levenshtein+wordnet).

The mean and variance N(μ,σ^2) for all similarity values of the three groups are presented in table 2.

Method	**4076 NPs**	**1323 FPs**	**2753 GPs**
cosine	N(.004,.001)	N(.499,.038)	N(.624,.032)
range	[.000,.423]	[.000,.944]	[.000,.985]
lev	N(.026,.001)	N(.342,.029)	N(.492,.037)
range	[.000,.429]	[.000,.800]	[.000,.900]
lev+wn	N(.100,.001)	N(.399,.026)	N(.530,.033)
range	[.012,.460]	[.035,.857]	[.044,1.0]

Table 2: Similarity methods compared

First, we note that all average values of similarity are consistent with the level of gradation for each group, non-paraphrases having the smallest values while good paraphrases the highest values of similarity. Dispersion around the mean is roughly the same, except for non-paraphrases, that have significantly lower dispersion. This is partly due to the fact that they are bound by zero on the left, but this also reflects the fact that all three methods won't find any similarity when there is none. On the other hand, all methods appear to miss a few similar semantic cues among para-

[4] Wordnet: `http://wordnet.princeton.edu/`

[5] `http://research.microsoft.com/research/downloads/default.aspx`

phrases, which explains a larger dispersion around the mean for paraphrases.

Some experiments (Leite *et al.* 2007) have shown that specific areas of document summarization may benefit from the use of more linguistic knowledge. However, our results show that there does not seem to be any improvement in the measurement of similarity when we consider word order and include a more refined measure of similarity between each word. In fact, the *cosine* method provides a greater range of values for similarity, a characteristic that shows a better ability to separate paraphrases from non-paraphrases. Therefore, we have adopted and integrated the *cosine* measure as part of our summarizer, and set a similarity threshold of 0.7 for detecting paraphrases.

2.4 Discussion of evaluation results

As can be seen in table 3, our run (aforementioned *LIPN1-opinion*, identifier 19) obtained more than competitive results.

Pyramid 0.393 F-measure (best: 0.534, worst: 0.101)	**Grammaticality** 6.636 score (best: 7.545, worst: 3.545)
Non-redundancy 6.818 score (best: 8.045, worst: 4.364)	**Structure/Coherence** 3.045 score (best: 3.591, worst: 2.000)
Fluency/Readability 4.591 score (best: 5.318, worst: 2.636)	**Responsiveness** 4.500 score (best: 5.773, worst: 1.682)

Table 3: Evaluation results

If we group the results by *content*, *fluency/readability* and *overall responsiveness*[6], then we have the following rankings:

Content LIPN1-opinion (0.393 F-measure, best:0.534 worst:0.101) is ranked fifth behind one manual run and three automatic runs

Fluency/readability LIPN1-opinion (4.218 score, best:4.873 worst:2.727) is ranked fourth behind one manual run and two automatic runs

Overall responsiveness LIPN1-opinion (4.500 score, best:5.318 worst:2.636) is ranked eighth behind one manual run and six automatic runs

If we give each of the six categories the same weight, then LIPN1-opinion is ranked first with an average score of 0.492 (worst:0.290). It is interesting to examine more in depth the results and look for insights into the summarization of opinionated texts. Our feature-oriented system lends itself naturally to evaluation as weights can be modified to see how discriminative the corresponding feature is. Ideally all the manual evaluation from TAC should be repeated for each relevant combination of features we wish to evaluate, however this is a labour intensive task clearly beyond our means. The alternative is to use off-the-shelves state-of-art automated evaluation software: ROUGE[7] provides a widely accepted approach to evaluate a summary against a set of *gold standard* summaries. For our purpose *gold standard* summaries are built by concatenating all snippets for a specific target. A legitimate concern with this strategy is how well ROUGE evaluation correlates with the PYRAMID evaluation adopted in TAC. Table 4 shows that for all the targets of our run LIPN1-opinion, ROUGE correlates only weakly with PYRAMID F-measures (coefficient of correlation is 0.23)

Target	PYRAMID	ROUGE
1018	0,892	0,304
1030	0,701	0,293
1026	0,541	0,406
1049	0,491	0,332
1024	0,483	0,207
1021	0,476	0,174
1008	0,458	0,457
1010	0,440	0,326
1022	0,434	0,252
1001	0,393	0,179
1009	0,386	0,177
1005	0,385	0,174
1043	0,384	0,242
1044	0,381	0,161
1019	0,351	0,325
1003	0,334	0,310
1027	0,308	0,202
1050	0,271	0,266
1033	0,182	0,506
1047	0,176	0,215
1004	0,127	0,270
1045	0,051	0,041

Table 4: Correlation between PYRAMID and ROUGE

However, ROUGE presents the same basic behaviour as PYRAMID, so that a summary covering only and only a subset of the snippets will have lower recall and unit precision, while a summary covering all snippets and more will have unit recall but lower precision. Other summaries will exhibit behavior between these two extremes.

We wish to evaluate the impact of each feature on the summary evaluation measures by varying the weight associated with each feature. To make comparison possible, we vary weights from the basic configuration used in the submitted run (see section 2.1) and which is represented by the middle row for each of the feature types in table 5.

Sentiment Analysis Let's recall that our basic strategy to match the polarity of each sentence from the blogs to a pair *target:question* was to assume that the sentence

[6] We group the different result categories as suggested in the README file provided by NIST with the results: 1. Content = Pyramid; 2. Fluency/Readability = Grammaticality, Non-redundancy, Structure/Coherence and Fluency/Readability; and 3. Overall responsiveness = Responsiveness

[7] `http://haydn.isi.edu/ROUGE/latest.html`

Feature	Weight	Recall	Prec.	F-sc.
sentiment	±000	0.605	0.200	0.288
sentiment	±020	0.622	0.177	0.265
sentiment	±050	0.610	0.154	0.228
isLongestSent	±050	0.605	0.184	0.270
isLongestSent	±010	0.622	0.177	0.265
isLongestSent	±000	0.627	0.175	0.262
sim. . .Target	±000	0.628	0.178	0.266
sim. . .Target	±020	0.622	0.177	0.265
sim. . .Target	±050	0.615	0.181	0.267
sim. . .Query	±000	0.622	0.181	0.267
sim. . .Query	±040	0.622	0.177	0.265
sim. . .Query	±080	0.623	0,179	0,265
sim. . .1stSent	±000	0,627	0,177	0,265
sim. . .1stSent	±010	0.622	0.177	0.265
sim. . .1stSent	±040	0.614	0.179	0,265
sim. . .snippets	±000	0.509	0.130	0.191
sim. . .snippets	±100	0.622	0.177	0.265
sim. . .snippets	±150	0.632	0.186	0.276
centroid	±000	0.622	0.185	0.272
centroid	±010	0.622	0.177	0.265
centroid	±050	0.631	0.156	0.240
position	±000	0.625	0.180	0.266
position	±010	0.622	0.177	0.265
position	±040	0.623	0.177	0.265

Table 5: Features impact

would share the same polarity as the blog itself. In view of the inverse relationship between sentiment weight and evaluation scores, it seems that in the domain of summarization at least, the assumption may not hold. Given that the assumption can be apparented to an approximation, this does not entail however that sentiment analysis per se is not useful for summarizing opinionated documents. To get a more complete picture of the use of sentiment analysis, a more comprehensive method of labelling sentences for sentiment is necessary.

Longest Sentence The idea here is to avoid populating summaries with long sentences, which we hypothesize could adversely affect readability. This does not seem to be the case.

Similarities with Target and Query The inclusion of similarity scores between the blog sentences and target or query follows from the observation that the main topics in the target (e.g. *Carmax*) and the query (e...g. *What motivated negative opinions regarding purchasing a car from CARMAX?*) tend to be repeated in opinions expressed toward the subject matter. This is probably true for targets, but probably not to the same extent for queries, as is exemplified by the following snippet (e.g. *It costs more to purchase a car from CARMAX than from other used car dealers*).

Similarity with First Sentence Works on summarizing factual documents has shown that the first sentence often captures essential elements of the whole text. However, the contribution of such sentence appears of no use in evaluation measures of summary content for opinionated texts.

Snippets Similarity Of all the features, we have substantial evidence that comparing our candidate sentences with the snippets by TAC has had a positive influence on the results. Although not surprising, to tap efficiently into this feature requires the use of a decent text similarity method, while the rather modest contribution of the feature suggests that it is by no means the sole responsible for the good results of the system.

Centroid Cluster Centroid clusters are at the very core of the traditional summarizing systems such as MEAD. Results show that although they seem to have improve recall slightly, a weight as high as 50 has not had any positive impact on precision and F-measures. One explanation is that the contribution of centroids is mitigated by the larger weight (and contribution) of snippets similarity, a feature not usually present in summarizing systems built around centroids.

Position This feature is a generalization of the *simWithFirstSentence* feature to all sentences in a document. We observe that there is a weak inverse relationship between the position of the sentence and evaluation measures. This suggests that authors may keep their most striking comments until the very end?

Compression Ratio By augmenting compression ration, i.e. by keeping a smaller number of sentences in the final summary, we tend to lower recall and increase precision, although not at the same rate so that the F-measure steadily increase as we compress the summary. As the relation between compression rate and F-measure does not show a gradual increase followed by a decrease (see table 6), it is not a simple matter of choosing the rate corresponding to the highest F-measure. Instead, the appropriate rate is a matter of trade-off between high enough semantic recall while not impairing on the quality and readability of the results by sustaining an adequate level of precision.

Feature	Nb. sent.	Rec.	Prec.	F-sc.
compression	010	0.441	0.301	0.335
compression	025	0.622	0.177	0.265
compression	050	0.719	0.103	0.173
compression	075	0.767	0.072	0.127

Table 6: Compression rate impact

2.5 Concluding Remarks on the LIPN1-Opinion System

We have presented an architecture to summarize opinionated texts which combines fairly standard features for text analysis with more exploratory features ensuing from sentiment analysis. Given that our system has had very good results for evaluation measures for quality and content, we conclude that this combination appears to be a good starting point to capture key elements that authors wish to express in subjective texts such as blog posts. Although our *post-mortem* analysis showed that *snippet snooping* has had an undeniable positive impact on the results, we have offered some insights on each feature contribution to help understand how the interaction of the parts played out together to produce proper summaries.

3 Around CBASES: A New Algorithm for Automatic Summarization

As presented above, LIPN1-Opinion was based on an adaptation of the MEAD system (Radev *et al.* 2004) integrating a sentiment analysis component. We have performed a series of other experiments and runs using a new algorithm called CBASES, which has been applied to the Opinion Summarization track (LIPN2-Opinion) as well as to the Update Summarization track (LIPN1-Update). We also participated in the Update track with a baseline system (LIPN2-Update) to be able to compare the results obtained by LIPN1-Update with a more basic approach.

We obtained surprising results since CBASES performed quite badly on the Update task but gave interesting results on the Opinion Summarization task, without using the snippets. In what follows, we first give a description of CBASES, our summarization algorithm. We then describe the results obtained for the different tracks.

3.1 CBASES: A Clustering-Based Sentence Extractor for Automatic Summarization

We assume that redundant pieces of information are the most important thing in order to produce a good summary. Therefore, the sentences which carry those pieces of information have to be extracted. Detecting groups of sentences conveying the same information is the first step of our approach. The developed algorithm first establishes the similarities between all sentences of the documents and then applies a clustering algorithm — fast global k-means (Lopez-Escobar, Carrasco-Ochoa, & Martinez Trinidad 2006) — to the similarity matrix in order to create clusters in which sentences convey the same information.

First, our system ranks all the sentences according to their similarity to the documents centroid. We have chosen to build up the documents centroid with the m most important terms, importance being reflected by the tf/idf of each term. We then select, to create a n sentences long summary, the n^2 best ranked sentences. We do so because the clustering algorithm we use to detect sentences conveying the same information, fast global k-means, behaves better when it has to group n^2 elements into n clusters. The similarity with centroid is a weighted sum of terms appearing in both centroid and sentence, normalized by sentence length.

Similarity between sentences is computed using a variant of the "Jaccard" measure. If two terms are not equal (similar lemma), we test their synonymy/hyperonymy using the Wordnet taxonomy (Fellbaum 1998). In case they are synonyms or hyperonym/hyponym, these terms are taken into account in the similarity calculation, but weighted respectively half and quarter in order to reflect that terms equality is more important than terms semantic relation. We do this in order to solve the problem pointed out in (Erkan & Radev 2004), where synonymy was not taken into account for sentence similarity measures, so to enhance sentence similarity measure. It is crucial to our system based on redundancy location as redundancy assumption is dependent on sentence similarities.

```
for all $e_j inE$
    $C_1 \leftarrow e_j$
for i from 1 to k do
    for j from 1 to i
        center($C_j$) $\leftarrow e_m | e_m maximizes \sum_{e_n in C_j} sim(e_m, e_n)$
    for all $e_j$ in E
        $e_j \rightarrow C_l | C_l maximizessim(center(C_l, e_j))$
    add a new cluster: $C_i$. It initially contains only its
    center, the worst represented element in its cluster.
done
```

Figure 1: Fast global k-means algorithm

Once the similarities are computed, we cluster the sentences using fast-global kmeans (description of the algorithm is in figure 1) using the similarity matrix. It works well on a small data set with a small number of dimensions, although it has not yet scaled up as well as we would have expected.

This clustering step completed, we select one sentence per cluster in order to produce a summary that contains most of the relevant information/ideas in the original documents. We do so by choosing the central sentence in each cluster. The central sentence is the one which maximizes the sum of similarities with the other sentences of its cluster. It should be the one that characterizes best the cluster in terms of information vehicled.

The overall process of our summarization system is shown in fig. 2.

3.2 LIPN2-Opinion: Summarizing Opinions using CBASES

In this section, we present the results obtained by CBASES when applied to the summarization of opinions from blogs (Opinion Summarization Pilot track). We first give information on how the runs were produced (we re-used some of the components described in section 2) and then detail our results.

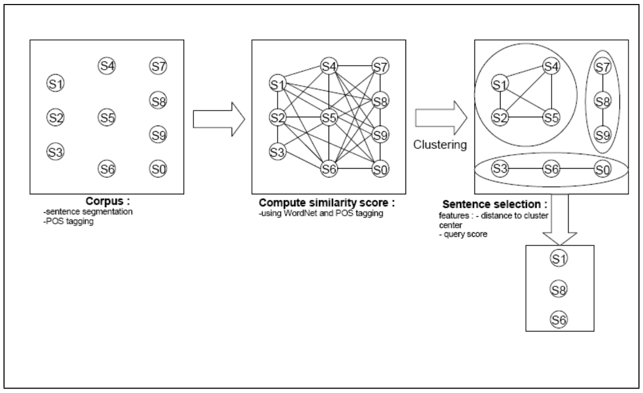

Figure 2: Summarization system

3.2.1 How the runs were produced

Snippets were not used for this system, as we want to keep a rather generic approach[8]. Runs were produced as described below:

Cleaning XML formatted blog posts were cleaned as described in 2.1.

Opinion tagging Using the method presented in section 2.2, sentences are tagged as positive, negative or even neutral.

Summarizing For each target, a summary is created using CBASES, which is detailed in section 3.

Question tagging Each question is tagged as positive or negative, using a small lexicon we created which links a few verbs and adjectives to either a positive or negative trait. Negation also is taken into account. The question is tagged as positive or negative depending on what feature prevails.

Reranking Every summary is reranked: sentences are grouped into three different groups: the first two groups are for positive and negative tagged sentences, the third one for neutral sentences. Groups of sentences appear in the same order as the question: if the first question is tagged as positive, the first sentences will be positive ones.

3.2.2 Results

Evaluation	CBSEAS	Mean	Best	Worst	Rank
Pyramid	0.393	0.151	0.251	0.101	5/20
Grammatic.	5.955	5.136	7.545	3.545	3/20
Non-redund.	6.636	5.878	7.909	4.364	4/20
Structure	3.500	2.677	3.591	2.045	2/20
Fluency	4.455	3.43	5.318	2.636	2/20
Responsiv.	2.636	2.614	5.773	1.682	8/20

Table 7: Opinion task overall results

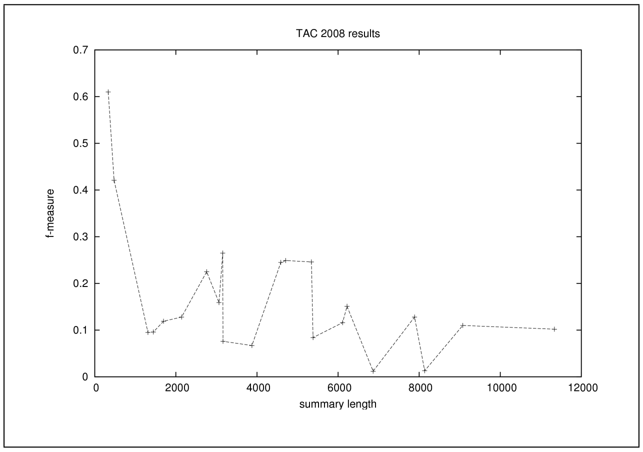

Figure 3: Opinion task LIPN2 results

For this experiment, we chose not to use the snippets (contrary to what has been presented in section 2 for LIPN1-Opinion), as we assume it is a more realistic challenge to look directly at the blogs with no external help. The results we present in table 7 are those of the participants who did not use the snippets. Indeed, systems using snippets obtained much higher scores than the other systems. We cannot compare our system to these.

LIPN2-Opinion obtained quite good results on the "opinion task", when compared to other systems not using the snippets. However, we stated that the quality of our summaries is very erratic. We assume this is due to the length of our summaries, as the longest summaries are the ones which get the worst scores (fig 3).

Solutions to fix this problem could be:

- Define a better score for the correspondence to a user query and remove sentences which are under a threshold;
- Extract sentences from the clusters that contain more than a predefined number of elements only.

This would result in improving the pertinence of the extracted sentences.

3.2.3 Concluding Remarks on LIPN2-Opinion

We have presented above our approach for creating extract summaries. The system we used for the opinion task obtained good results but could be improved by integrating some post-processing methods.

[8] It seems quite unlikely that in real conditions, a system can make use of queries, relevant documents and snippets. In our opinion, it seems more promising to imagine a search engine coupled with a summarization system, even if we have explored an alternative approach making use of the snippets in the system presented in section 2. Of course the snippets help getting more accurate results.

We plan to study other important features that could yield some improvement:

- Taking into account the task for establishing a task-adapted sentence similarity measure. A lot of work still remains to be done in order to better take into account opinion detection for the prodution of summaries.
- Studying Wordnet contribution: as shown in section 2.3, use of Wordnet is not necessarily synonym to quality improvement. However, we think that the quality will increase if we integrate more relevant linguistic resources for improving the quality of paraphrase detection.

3.3 Update task: LIPN systems

The Document Understanding Conference (DUC) 2007 had introduced a new pilot task that fits well with new paradigms on document automatic summary. In TAC 2008, this pilot task, called "Update Task", consists of producing two summaries: the first one about a first group of documents, and the second one for information that represents novelty in a second group of documents.

In this section, we describe how we adapted our summarizing system, CBASES, to the update task and discuss the results and how they can be improved.

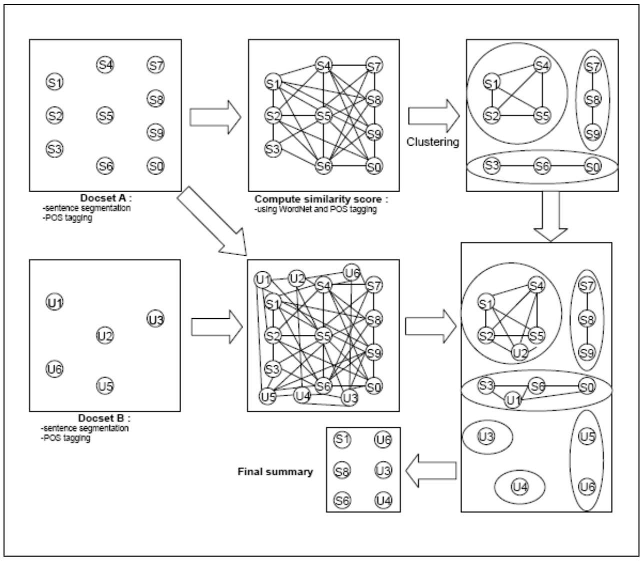

Figure 4: CBASES Update system

3.3.1 Managing Update

We have a summarizing system, CBASES, at our disposal. It is described in section 3.1. This system clusters sentences from documents to summarize in order to create clusters in which sentences convey the same information.

The update task in TAC2008 consists in summarizing a first group of documents, and then summarizing what is new in a second group of documents. The first summary is established using CBASES. Then, summarizing the second group of documents consists in managing updates. This should be done by detecting what is new in the second group of documents, compared to the first group of documents, and summarizing only the part of documents which represents an update.

3.3.2 LIPN1-Update: CBASES adaptation to the update task

Detecting novelty is the first step in managing novelty. Detecting sentences in the second document set which are redundant with sentences in the first document set is the method we have chosen for that purpose. We consider that sentences are not conveying novelty if they are closer to sentences belonging to the first document set than sentences belonging to the second document set.

After having summarized the first document set (see fig.2), we recomputed all sentence similarities after including the new sentences from the second document set. We then marked all sentences in the clustering first obtained as immobile. Using fast global kmeans, we continued the clustering adding new clusters, with the following constraints:

- the documents from the first document set must not be moved to another cluster;
- the cluster centers from the first clustering must not be recalculated.

Doing so, sentences from the second document set which are close to sentences from the first one are added to old clusters, whereas sentences which appear to bring novelty are added to new clusters. These new clusters are the ones we will select sentences from for the second document set. Fig 4 gives a view of our update system.

3.3.3 LIPN2-Update: A Baseline system

We wanted to compare our system to a basic system for sentence selection. The feature on which is based this system is sentence specialty: it selects sentences whose terms are close to the bottom of the Wordnet hierarchy.

3.3.4 Results Analysis

Results were surprising: CBASES (v0.1) obtained worse scores than our baseline. We identified two major problems:

- document set B summaries were inferior to document set A summaries in terms of precision and recall;
- recall for all summaries was two times worse than precision.

This was due to bad choices we made about building summaries: we chose to extract only sentences which are not alone in their cluster; this had the effect of reducing the size of document set B summaries and to lower recall for these

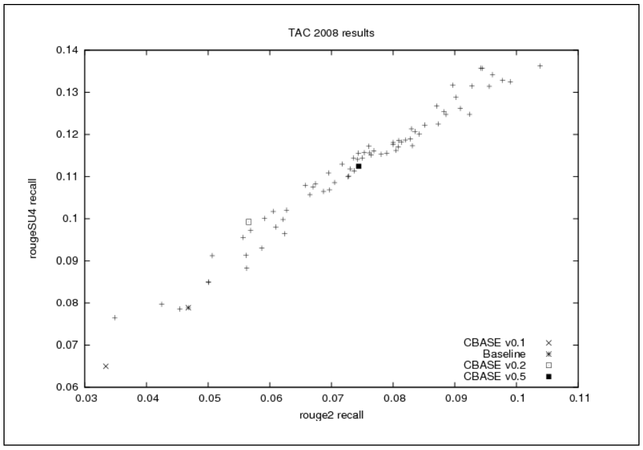

Figure 5: Results of our system on the "Update" task

summaries. We also chose to prefer summaries which do not exceed 100 words. For that reason, our system often preferred 60 words long to more than 100 words long summaries. That resulted in artificially lowering recall for document set A and document set B summaries.

We decided not to remove lone sentences, and to avoid getting sentences which do not fit the topic, we filter sentences according to their similarity to topic.

We then adopted another strategy to produce summaries that are always 100 words long: we arbitrary chose to select seven sentences from each document set. That is enough to always get 100 words, and not too much as to add noise to extracted sentences when increasing the number of clusters (CBASES v0.2).

The last change we made to our system was to extract from a cluster not the sentence closest to the centre but the sentence which maximizes similarity with a query (CBASES v0.5).

The results of the first version of our system and of the new ones are shown in fig.5.

What is important to notice is that our system also obtained poor results in manual evaluation, for the same reasons as for automatic evaluation.

3.4 Concluding Remarks on LIPN1-Update

We presented in this paper our automatic summarization system. If the results obtained during the evaluation were disappointing, the corrected system behaves well. There are three ways CBASES can be improved further:

- Improving query scores by introducing sentence position and features such as numerical expressions and query extensions;
- Sentence compression: deleting parts of the sentence which are not relevant;
- Introducing request extensions.

4 Conclusion

We have presented in this paper the system developed by LIPN for TAC 2008. We have obtained encouraging results, especially for the Opinion Summarization track. We are currently in the process of evaluating the different components of our systems, their relative performance and their contribution to the task. Since it was our first experiments in summarization and our first participation in TAC, we know that a lot of work remains to be done to improve the quality and the accuracy of our systems. We have presented in this document some proposals to enhance the overall quality of the different components and algorithms that remain to be tested.

Acknowledgement

This work has been conducted under the project *Infom@gic* as part of the French Business Cluster *Cap Digital*.

References

Brockett, C., and Dolan, W. B. 2005. Support vector machines for paraphrase identification and corpus construction. In *The 3rd International Workshop on Paraphrasing (IWP2005)*.

Erkan, G., and Radev, D. R. 2004. Lexrank: Graph-based centrality as salience in text summarization. *Journal of Artificial Intelligence Research (JAIR)* 22:457–479.

Fellbaum, C., ed. 1998. *WordNet, an Electronic Lexical Database*. Cambridge: MIT Press.

Fry, E. B.; Kress, J. E.; and Fountoukidis, D. L. 2000. *The Reading Teachers Book of Lists*. Hoboken: Jossey-Bass, 4th edition.

Leite, D.; Rino, L.; Pardo, T.; and Nunes, M. 2007. Extractive automatic summarization: Does more linguistic knowledge make a difference? In *HLT/NAACL Workshop on TextGraphs-2: Graph-Based Algorithms for Natural Language Processing*, 17–24.

Lopez-Escobar, S.; Carrasco-Ochoa, J. A.; and Martinez Trinidad, J. F. 2006. Fast global k-means with similarity functions algorithm. In Corchado, E.; Yin, H.; Botti, V. J.; and Fyfe, C., eds., *IDEAL*, volume 4224 of *Lecture Notes in Computer Science*, 512–521. Springer.

Pedersen, T.; Patwardhan, S.; and Michelizzi, J. 2004. Wordnet: : Similarity - measuring the relatedness of concepts. In *Proceedings of the Conference of the Association for the Advancement of Artificial Intelligence (AAAI 2004)*, 1024–1025.

Radev, D.; Jing, H.; Styś, M.; and Tam, D. 2004. Centroid-based summarization of multiple documents. *Information Processing and Management* 40:919–938.

Ristad, E. S., and Yianilos, P. N. 1998. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(5):522–532.