CCNU at TAC 2008：Proceeding on Using Semantic Method for Automated Summarization Yield

Tingting He, Jinguang Chen, Zhuoming Gui, Fang Li
Huazhong Normal University, 430079, Wuhan, China
tthe@mail.ccnu.edu.cn,cjg2003@hutc.zj.cn, Wenzheng38@sina.com, fang__lf@163.com

Abstract:
The CCNU summarization system, PUSMS (Proceeding to Using Semantic Method for Summarization), join in TAC (formerly DUC) for the first time. For the update summarization tasks, we used syntactic-based anaphora resolution and sentence compression algorithms in our system. Term significance was then obtained by frequency-related topic significance and query-related significance by obtaining co-occurrence information with query terms. For the pilot QA summarization task, a semantic orientation recognition module which used WordNet::Similarity::Vector to obtain all of the main part-of-speech terms' similarity with benchmark words derived from General Inquirer is used in PUSMS pilot system. We also developed a document classifier and a snippets-related content extracting module for the pilot tasks. In all, our initial job can be boiled down to be introducing semantic method into our former statistical summarization system. By analyzing the evaluation results, we found that we were preceding the right target but still have a long way to go.

1 Introduction
Although be fresh for TAC, we have spend many years in automated summarization area. In this report, we focus on how the semantic method involved in our summarization system and the changes it makes. Our system, PUSMS (Proceeding to Using Semantic Method for Summarization) performed not bad in TAC 2008, and we argued that given more experience, our performance can be better. And we did made valuable improvement in several areas, this paper is focusing on introduce the valuable improvement other than every details of building a summarization system.

2 The PUSMS System Architecture
PUSMS consists of the following five steps:
1. Content extracting and sentence splitting.
2. Syntactic-based anaphora resolution and sentence compression
3. Term significance computation for each of the document sets.
4. Sentence scoring.
5. Dynamic sentence choosing and Redundancy removal.

These steps are common for the main and pilot tasks, and details of them can be found in section 2.1-2.5, respectively. For the update summarization task, the historical information is removed in step 4, which is described in section 2.6. And for the pilot QA summarization, a semantic orientation classification step is added between step 2 and 3, which is described in section 2.7.

2.1 Content Extracting and Sentence Splitting
The document preparation step begins with content extracting. For the main task, content extracting is simpler and need little operation because all the test data is uniform in style and have specific tags to mark different functional parts. But for pilot task, since all the experiment data come from BLOG, content extracting became very complex. There are no definite tags to be used and the test data content varied

greatly. In PUSMS, we used the Stanford HTMLParser[1] for deriving text contents from BLOG. The HTMLParser is very powerful but still have some problem in derived all meaningful content from BLOG, in some cases, the HTMLParser derived noting from some source BLOG; and in other, the HTMLParser return meaningful information including some useless information like date line and BLOG introduction. We add some rules to solve this problem.

We develop a sentence splitter in PUSMS. The sentence splitter absorbed a majority of rules from a Perl sentence splitter with the following additional changes made to compensate for erroneous splits:

1. Remove all double quotation marks.
2. Wherever there is a colon, we eliminate the content lead it as well as itself.
3. Wherever there is a semicolon, we treat the content lead and follow it as complete sentences respectively.
4. We eliminate all question sentences and plaint sentences to avoiding their influences for fluency of summaries.

2.2 Anaphora Resolution

We used JavaRAP developed by Long Qiu et al.[1] to build a syntactic-based anaphora resolution module in PUSMS. JavaRAP is an implementation of the classic Resolution of Anaphora Procedure (RAP) given by Lappin and Leass (1994). It resolves third person pronouns, lexical anaphors, and identifies pleonastic pronouns.

The problem we found in using JavaRAP in the summarization area. We proposed that anaphora resolution should be classified into two classes in summarization field: anaphora resolution within a sentence and cross sentences. The former make sentences contain more redundancy information while the later make the sentence more complete and understandable, since most summarization system extracted complete sentences and organized them into summaries. So we made some changes on JavaRAP and used only cross-sentences RAP in PUSMS.

ORG	The takeoff went perfectly, Alain Garcia, an Airbus engineering executive, told the LCI television station in Paris. **Its** comparative lack of noise was the result of demands by customers that Airbus make the plane even more quiet than **it** already was, a process that took six months.
Within-sentences And Cross-sentences	The takeoff went perfectly, Alain Garcia, an Airbus engineering executive, told the LCI television station in Paris. **<*The takeoff's*>** comparative lack of noise was the result of demands by customers that Airbus make the plane even more quiet than **<the plane>** already was, a process that took six months.
Cross-sentences Only	The takeoff went perfectly, Alain Garcia, an Airbus engineering executive, told the LCI television station in Paris. **<*The takeoff's*>** comparative lack of noise was the result of demands by customers that Airbus make the plane even more quiet than it already was, a process that took six months.

Table 1: Example of different anaphora resolution results

2.3 Syntactic-based Sentence Compression

We used a syntactic-based sentence compression in PUSMS. We performed all compression by the support of a maximum-entropy-inspired parser developed by Charniak[3][4], which parsed down sentences to Penn tree-bank style parse trees. After that, we built a rule-based sentence compression module with Tsurgeon, a tree-transformation utility built on top of the Tregex tree-matching engine.

We proposed the following five rules as shown in Table 2. These rules is made to eliminate as much as possible redundancy information while keep the sentences grammatical. They are indexed by the decrease

[1] http://sourceforge.net/projects/htmlparser

of the confidence for using them, i.e. using rule n ($1 \leqslant n \leqslant 4$) is less likely to create ungrammatical sentences than rule n+1. Rule 1 eliminates prepositional phrases whose father node is a clause or attached by comma or appear successively. Rule 2 eliminates subordinate clause. Rule 3 deal with the appositive. Rule 4 eliminate gerund phrases or subordinate clause. Rule 5 eliminate noun phrase like ", John said".

Another problem of sentence compression is that important information may be trimmed even if the sentence is grammatical. A dynamic selection algorithm is advanced in PUSMS in the sentence scoring stage to avoid important information be eliminated, details can be found in section 2.5.

RULE	Tregex pattern command	Tsurgeon command
1	@PP|ADVP=pp [>, S | $, /,/ | $, ADVP | $, PP]	delete pp
2	SBAR=sb !<< FRAG [>, S | $, /,/ | > NP | $. /,/]	delete sb
3	@NP|SBAR=nn $. /,/=cr [$, /,/=cl | >>, S]	delete nn cr cl
4	VP [< VBG | < VBD] > (S > (@PP|SBAR=gre > NP))	delete gre
5	NP=np $, /,/=cl $. (VP=vp <: @VBD|VBN)	delete np cl vp

Table 2: Example of different anaphora resolution results

2.4 Sentence Scoring

To evaluate whether a sentence is appropriately included in the summary, two factors are considered. One is the association between a sentence and the query, and the other consideration is the information density of a sentence compared to other sentences in the topic set. Generally, more responsive a sentence is to the query and more information density the sentence contains, more possible the sentence is to be included in the final summary.

Before obtain sentence score, we applied the removal of stop words and word stemming in PUSMS. And sentence scores are obtained from two sentence scoring model.

Query-related Sentence Scoring

In PUSMS, the association between a sentence and the query is obtained with a modified relevance-based language modeling described in Jagadeesh et al., 2005.

In Jagadeesh et al., 2005, It has been shown that relevance-based language modeling (Lavrenko and Croft, 2001) along with semantic representation of words in higher dimensions using HAL spaces (Lund and Burgess, 1996) can be extended to calculate the relevance score of a sentence towards the information need. The relevance-based language modeling assumes both the query and the document as samples from an unknown relevance model R, then it approximates *P(w|R)*, the probability of observing a word *w* in the documents relevant to a particular information need R, using *P(w|Q)*, where $Q = q_1, q_2 \cdots q_k$ is the information need expressed in the form of query words. By definition, the conditional probability can be expressed in terms of the joint probability of observing w with the query words $q_1, q_2 \cdots q_k$.

$$P(w/R) \approx P(w/Q) = P(w/q_1, q_2 \ldots q_k) = \frac{P(w, q_1 \ldots q_k)}{P(q_1 \ldots q_k)}$$

Assumes the query words $q_1, q_2 \cdots q_k$ to be independent of each other while keeping their dependencies on *w* intact. Using this assumption the joint probability can be calculated as:

$$P(w, q_1, \ldots q_k) = P(w) \prod_{i=1}^{k} P(q_i/w)$$

$P(q_i|w)$, can be incorporated into the above expression using the probabilistic interpretation of Hyperspace Analogue to Language (HAL) model. Hyperspace Analogue to Language model constructs the dependencies of a word w on other words based on their occurrence in the context of *w* in a sufficiently large corpus.

Although International Institute of Information Technology (IIIT) did very well in DUC, we found that there is a technical problem with formulating the task in this way. In fact, given a particular information need $Q = q_1, q_2 \cdots q_k$, if choice has to be made between a word having looser correlation with all words in R and a word having tighter correlation with part of words in R and no correlation with the other words in R, people usually, in general, prefer the later.

Let's see a toy example about the Jaggedness model. Assumes that there are only two q_1, q_2 in Q and two observing words w_1, w_2. $P(w_1) = P(w_2)=0.1$, $P(q_1|w_1)= P(q_2|w_1)=0.1$, $P(q_1|w_2)=0.01$, $P(q_2|w_2)=0.9$. So the joint probability can be calculated as:

$$\mathrm{P}(\mathrm{w}_1, \mathrm{q}_1, q_2) = \mathrm{P}(\mathrm{w}_1) \bullet \mathrm{P}(\mathrm{q}_1|\mathrm{w}_1) \bullet \mathrm{P}(\mathrm{q}_2|\mathrm{w}_1) = 0.1 \times 0.1 \times 0.1 = 10^{-3}$$

$$\mathrm{P}(\mathrm{w}_2, \mathrm{q}_1, q_2) = \mathrm{P}(\mathrm{w}_2) \bullet \mathrm{P}(\mathrm{q}_1|\mathrm{w}_2) \bullet \mathrm{P}(\mathrm{q}_2|\mathrm{w}_2) = 0.1 \times 0.01 \times 0.9 = 0.9 \times 10^{-3}$$

The Jaggedness model prefers w_1, but in intuition, w_2 have high correlations with q_2 and contain more important information than w_1 do. This problem is enlarged and becomes insolvable when Q contains a lot of words, the joint probability of Jaggedness model becomes extreme small and indiscriminate. So we proposed the following evolved formulation to calculate the joint probability:

$$P(w_1, q_1 \ldots q_k) = P(w) \sum_{i=1}^{k} P(q_i \mid w)$$

Let's see the toy example again, in our formulation, the joint probability can be calculated as:

$$P(w_1, q_1, q_2) = P(w_1) \bullet (P(q_1 \mid w_1) + P(q_2 \mid w_1)) = 0.1 \times (0.1 + 0.1) = 0.02$$

$$\mathrm{P}(\mathrm{w}_2, \mathrm{q}_1, q_2) = \mathrm{P}(\mathrm{w}_2) \bullet (\mathrm{P}(\mathrm{q}_1|\mathrm{w}_2) + \mathrm{P}(\mathrm{q}_2|\mathrm{w}_2)) = 0.1 \times (0.01 + 0.9) = 0.091$$

The effectiveness of the evolved formulation is shown in section 4.

For the same reason, assuming that the different words in a sentence are independent and removing the constant terms, the relevance of a sentence S, can be expressed as,

$$P(S \mid R) = \frac{\sum_{W_i \in S} P(w_i \mid R)}{L} = \frac{\sum_{W_i \in S} P(w) \sum_{q_j} P(q_j \mid w)}{L} = \frac{\sum_{W_i \in S} P(w) \sum_{q_j} pHAL(q_j \mid w)}{L}$$

Where L is the length of S and the pHAL, probabilistic HAL, can be interpreted as, given a word w what is the probability of associating a word wo with w in a window of size K.

Query-independent sentence scoring

In our opinion, signature terms are those terms which occur with high frequency in the whole topic set, rather than in the single document it's located. So we proposed a novel inverse-probability-rate model to obtain the information density of a sentence. Probability rate is a language model originally used in information retrieval area. We adopted and improved the method. The following describes the detail of the inverse-probability-rate model.

In our model, a sentence is denoted by a vector

$$S = \{<t_1, w_1>, <t_2, w_2> \cdots <t_n, w_n>\}$$

Where w_i is the weight of feature t_i.

$$w_i = \log \frac{p(t_i \mid Cpos)(1 - p(t_i \mid Cneg))}{p(t_i \mid Cneg)(1 - p(t_i \mid Cpos))}$$

Where *Cpos* is all the documents in the whole topic set, *Cneg* is the document where t_i is located.

So the query-independent sentence score can be obtained as follow:

$$QI(S) = \frac{1}{n} \sum_i w_i$$

Sentence scoring

The final score of sentence is

$$\text{Score}(S_i) = \beta \bullet \frac{QI(S)_i}{\sum_{i=1}^{k} QI(S)_i} + (1-\beta) \bullet \frac{P(S \mid R)_i}{\sum_{i=1}^{k} P(S \mid R)_i}$$

Where k is the number of sentences in the whole document set, β is the factor of mixing the query-related and query-independent score. We trained β in the DUC 2007 data, and the experienced value of β is 0.1.

2.5 Dynamic sentence choosing and Redundancy removal.

In PUSMS, since the syntactic-based sentence compression module produced multi-candidate sentences and inclined to eliminate important information, a novel dynamic sentence choosing method is proposed.

The basic idea of our dynamic choosing method is that a compression is permitted only if the informational density of the compressed sentence should be larger than the original sentence. A compression is done only if the following conditional expression fulfilled:

$$\frac{\text{Score}(S_{\text{original}}) - \text{Score}(S_{\text{cut}})}{\text{Length}(S_{\text{original}}) - \text{Length}(S_{\text{cut}})} \le \alpha \bullet \frac{\text{Score}(S_{\text{original}})}{\text{Length}(S_{\text{original}})}$$

Where $Score_{original}$ and $Score_{cut}$ is the score of the original sentence and the compressed sentence according to section 2.4. α is the compressional intensity control factor, and its experienced value is 0.8 according to the 2007 TAC data.

The dynamic sentence choosing method is iterately performed five times, by using one rule in table 2 one by one.

After the dynamic sentence choosing process, all sentences in the whole topic set can get a score and be sorted degressively.

To avoid including redundancy information in the summary, we used an evolved Maximal Marginal Relevance method in PUSMS. The basic idea of the MMR method is to choose the most significant sentence while minimize the redundancy between a candidate sentence and former selected sentences. The MMR method can be defined as the following:

$$S_i = \arg\max_{S_i \in R-F} [\lambda \bullet Score(S_i) - (1-\lambda) \bullet \max_{S_j \in F} sim(S_i, S_j)]$$

Where F is collection of the selected sentences, R is collection of the sorted sentences in the topic set *Sim*(S_i,S_j) is the cosine-similarity between candidate sentence S_i and former selected sentence S_j. PUSMS choose candidate sentence S_i iterately until given length of the summary is reached.

2.6 Removal of the historical information

In order to remove historical information form topic set B, we used a sentence filtering method in PUSMS. The basic idea of this method is to decide which part of information in the current topic set is new information and to use the new information to generate summaries. Our sentence filtering method can be defined as the following:

$$S_i = \arg\max_{S_i \in R} [\sum_{S_j \in H} sim(S_i, S_j)]$$

Where F is collection of all sentences in topic set B, H is collection of all sentences in topic set A. *Sim*(S_i,S_j) is the same as described in section 2.5. PUSMS iteratively eliminate sentences which obtained highest score in the upper formulation, until the conditional expression as the following is fulfilled:

$$\underset{S \in E}{Count}(S) \ge \mu \bullet \underset{S \in R}{Count}(S)$$

Where E is collection of former eliminated sentences, μ is the filter strength intensity factor. We trained μ in the DUC 2007 data, and the experienced value of μ is 0.1.

2.7 Semantic Orientation Classification

In TAC 2008, most of the questions in pilot QA summarization task are about the semantic orientation problem. A semantic orientation classifier is used in PUSMS to derived positive and negative documents form the topic set.

To build a semantic orientation classifier, we firstly manually find 15 pair of benchmark words from General Inquirer, including 5 pair of noun, verb and adjective, respectively. Then we proposed a novel word semantic similarity calculating method which is based on WordNet::Similarity::Vector to ensure that all part of speech terms can be obtained polarity score. The polarity score of each noun, verb and adjective in sentence S can be obtained as following:

$$PolarityScore(word) = \sum_{i=1}^{k} Sim(word, PositiveWord_i) - \sum_{i=1}^{k} Sim(word, NegativeWord_i)$$

Where *k* is the num of benchmark words $Sim(word, PositiveWord_i)$ is the similarity between a word and the positive benchmark word obtained from WordNet::Similarity::Vector.

Then polarity score of sentence S can be calculated as following:

$$PolarityScore(S) = \sum_{i=1}^{l} PolarityS\,\mathrm{co}\,re(word_i) \bullet (-1)^{Count(NOT)}$$

Where *l* is number of words in sentence S, Count (NOT) is the number of privative word in sentence S.

Benchmark words as well as privative words used in PUSMS are shown in Table 3.

	Positive	**Negative**
Benchmark Words (Noun)	achievement respect dependability joy conscience	disadvantage mistake chaos sorrow devil
Benchmark Words (Verb)	admire praise trust love contribute	harm deny doubt dislike deceive
Benchmark Words (Adjective)	good brave famous positive correct	bad incorrect untrue unhappy evil
Privative Words	no never none nothing seldom rarely hardly scarely barely not doesn't isn't aren't wasn't weren't hasn't haven't hadn't don't didn't won't wouldn't shan't shouldn't can't cannot couldn't mustn't	

Table 3: Benchmark Words and Privative words used in PUSMS

System ID	mod Pyramid score - A	numScus - A	linguistic quality - A	overall responsiveness - A	mod Pyramid score - B	numScus - B	linguistic quality - B	overall responsiveness - B	Rouge2 R - A	Rouge2 R - B	RougeSU4 R - A	RougeSU4 R - B	BE R - A	BE R - B
3	20	23	27	23	39	31	35	32	38	51	25	49	37	55
35	21	18	22	21	33	26	27	38	31	45	21	41	38	49
58	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	27	44	18	45	44	51
nb of peers	58	58	58	58	58	58	58	58	72	72	72	72	72	72

Table 4: Performance of PUSMS in TAC 2008 Update Summarization Task

3 Results

3.1 Update Task

For the update summarization task, PUSMS submitted 3 systems, system 3, 35 and 58, respectively. To compare the effectiveness of different module of method, we didn't use anaphora resolution and sentence trimming in system 3, and didn't used anaphora resolution and sentence trimming and removal of removal of the historical information in system 58. The performance of our system can be seen in Table 4.

As seen in Table 4, PUSMS performed not bad for a new participator in TAC and gained best place of 18 out of 72 system. We also found that PUSMS performed rather worse in topic set B than in set A. We think it is because our over-training the filter strength intensity factor μ in DUC 2007 data. While only very little sentences filtered (10%) in set B, and these filtered sentences have little probability selected into summary, our historical information removal module seemed to be of no use. We also found that PUSMS perform better when evaluated by human and pyramid than by ROUGE. That may because of that we pay great emphasis on using semantic method in PUSMS.

System ID	pyramid F-score	Average score for Grammaticality	Average score for Non-redundancy	Average score for Structure/Coherence	Average score for Overall fluency/readability	Average score for Overall responsiveness
7	24	17	14	17	7	33
27	18	14	9	8	13	35
nb of peers	36	36	36	36	36	36

Table 5: Performance of PUSMS in TAC 2008 Pilot QA Summarization Task

3.2 Pilot summarization task

For the pilot QA summarization task, PUSMS submitted 2 systems, system 7, 27, respectively. To compare the effectiveness of different module of method, we didn't use anaphora resolution and sentence trimming in system 7. The performance of our system can be seen in Table 5.

As seen in Table 5, system 27 is performed better than system 7. And system 27 performed very well in average score for non-redundancy, average score for structure/coherence and average score for overall fluency/readability. While both system 7 and system 27 performed very bad in average overall responsiveness. We were confused at first seen this result, since system 27 performed not bad in pyramid F-score. Finally we found that this is mainly because of that we generated too short sentences in PUSMS (≤400 words), while TAC permit to submit longer summaries (≤4000 characters).

4 Conclusion

In this paper we have presented PUSMS, a semantic based summarization system. In all, our initial job can be boiled down to be introducing semantic method into our former statistical summarization system. By analyzing the evaluation results, we found that we were preceding the right target but still have a long way to go.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No. 60773167), the 973 National Basic Research Program (No. 2007CB310804), the Project in the National Science & Technology Pillar Program in the Eleventh Five-year Plan Period (No. 2006BAK11B03), the Program of Introducing Talents of Discipline to Universities (No. B07042), and the Natural Science Foundation of Hubei Province (NO. 2006ABC011).

References

[1] Long Qiu, Min-Yen Kan and Tat-Seng Chua. (2004). A Public Reference Implementation of the RAP Anaphora Resolution Algorithm. In proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004). Vol. I, pp. 291-294.

[2] Lappin, S. Leass, H. 1994. An algorithm for pronominal anaphora resolution, Computational Linguistics, 20(4), 535-561.

[3]E. Charniak. A maximum-entropy-inspired parser. In Proceedings of the Conference on North American Chapter of the Association for Computational Linguistics (ANLP-NAACL), pages 132-139, 2000.

[4]E. Charniak and M. Johnson. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In Proceedings of the Annual Meetings of the Association for Computational Linguistics (ACL), pages 173-180, 2005.