SWING: Exploiting Category-Specific Information for Guided Summarization

Jun-Ping Ng Praveen Bysani Ziheng Lin
Min-Yen Kan Chew-Lim Tan

School of Computing
National University of Singapore
{junping,bpraveen,linzihen,kanmy,tancl}@comp.nus.edu.sg

Abstract

We present our work towards building a robust multiple document summarizer (SWING), with a focus on guided summarization. SWING is an extractive summarizer built upon information retrieval principles. Our key contribution is utilizing category knowledge, collected over all news topics, to calculate *category-specific importance (CSI)* of sentences. We propose two new category-specific features in this work to compute the CSI of a sentence. We also exploit identified named entities to improve the guided summarization process. Evaluation results show that our methods are effective with respect to both the ROUGE and Pyramid scoring metrics.

1 Introduction

The Guided Summarization task at TAC 2011 is designed to promote the development of abstractive and content-aware summarization techniques. It guides the participant systems to do so by explicitly assigning topics to categories and advocating the systems to retrieve content relevant to the aspects of each category. Aside from this main component, the task also has an update component that requires systems to generate summaries that assume the user has read some articles from the same topic before. The task is similar to Update Summarization task at TAC 2009 (Dang and Owczarzak, 2009).

Some interesting approaches have been previously proposed at TAC 2010 for the Guided Summarization task. Conroy et al. (2010) augmented their CLASSY system with a guided query generation component that expands query terms for each aspect of category by performing searches over Google, dictionaries, thesaurii and authored world knowledge. Steinberger et al. (2010) generates guided summaries via information extraction principles. Aspect information extracted from an entity extractor is coupled with a latent semantic analysis model to capture relevant information. They also build lexicons for some category aspects that are not identified by the event extractor. External knowledge such as Wikipedia is also used by many groups (Varma et al., 2010) for this task. An ample set of relevant articles are selected manually from Wikipedia for each category. These articles are used to build domain models and later to extract important sentences containing events mentioned in the template.

We believe that providing category-specific information in a summary is as important as providing relevant topic information. To this effect, we have developed a robust sentence-extractive summarizer, SWING (a guided Summarizer from WING[1]), based on Information Retrieval principles. We extract diverse features from the article text, and utilize them to estimate the importance of information through a regression model. The set of features employed include simple heuristics (length of sentence, positional information of sentence in the article), frequency measures (document frequency) and relative entropy estimates (Kullback-Lieber divergence). While these features help to find the generic importance of individual sentences, a key aspect of our summarizer is that it also makes use of the arti-

[1]The Web Information Retrieval / Natural Language Processing Group (WING).

cle's associated category for each topic to calculate the category-specific importance (CSI) of each sentence. CSI is captured through both category relevance scores and category differential measures.

As answer types for most of the aspects (WHERE, WHO, WHEN) are entities, we also carried out an additional set of experiments to explore how automatically-identified named entities can improve the guided summarization process. We built ranked lists of entities based on their frequency of occurrence at different levels of granularity (i.e. topic and corpus levels) and studied if these have a positive influence on content responsiveness.

In following sections, we explain the system architecture and the features used for computing relevance and CSI. Later, we detail about the submission runs and evaluation results. Finally we present our post-competition analysis and discuss our findings.

2 System Overview

SWING is designed to be an easy-to-use and effective testbed for the evaluation of summarization techniques. Design decisions were made keeping in mind the need for the system to be easily modifiable.

2.1 Architecture

The system consists of several independent Ruby programs linked together with Unix pipes.

Independent modules and programs. Building independent modules makes it easy to add and remove functionalities to and from the system. New ideas and techniques can be incorporated easily without affecting the other parts of the system.

Unix pipes. Unix pipes are a stable, well-proven method of inter-process communication. By standardizing the data format of input and output streams, the independent modules can communicate with each other easily. We have chosen to make use of Javascript Object Notation (JSON) to define the input and output streams because it is simple to construct and easy to process programatically.

Figure 1 illustrates how the different independent modules are chained together. Key functionalities which we have incorporated into this pipeline include:

Input. Converts provided source documents into a JSON formatted string which is compatible with the rest of the system.

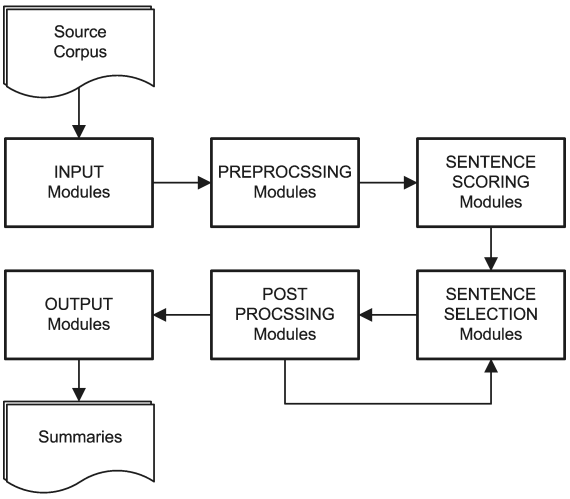

Figure 1: SWING pipeline.

Preprocessing. Processes input text as desired before proceeding with summarization.

Summarization. The main functional component which implements the desired summarization algorithm.

Output. Transforms the output of the pipeline into a desired format for evaluation and validation.

2.2 Methodology

Our summarization system is based fundamentally on a supervised machine learning approach. A set of syntactic features is first derived from the input documents. They are combined together with a set of weights derived from support vector regression (SVR). The model used for SVR is trained on the TAC summarization track data for 2010. After scoring each sentence, the Maximal Marginal Relevance algorithm (Carbonell and Goldstein, 1998) is then used to perform sentence re-ranking and selection.

2.2.1 Features

SWING consists of two classes of features: generic features and category-specific features. Generic features include sentence length (SL), sentence position (SP), and a modified version of document frequency. We also use two category-specific features – Category Relevance score (CRS) and Category KL-Divergence (CKLD) – to compute category-specific importance (CSI) of a sentence.

2.2.1.1 Generic Features

These features capture generic information relevant to all news articles.

Sentence position (Edmundson, 1969) is a simple yet effective feature that is being used in summarization in the news domain. The intuition is that leading sentences in a news article usually contain summarizing information. Since the TAC data consists of news articles, we use the positional information to boost the top sentences of an article. The score of a sentence s is computed as,

$$SP(s) = 1 - \frac{p}{N}$$

where p is the position of sentence s and N is the total number of sentences in that article.

Sentence length is a binary feature that checks if the number of words in a sentence is at least 10. This feature helps in avoiding noisy short text in the summary. The value 10 is empirically determined in our system tuning.

$$SL(s) = \begin{cases} 1 & \text{if } len(s) >= 10 \\ 0 & \text{otherwise} \end{cases}$$

Bigram DFS (BDFS) Document frequency score (DFS) is a generic scoring feature that has been proven successful in past summarization tasks (Bysani et al., 2009; Schilder and Kondadadi, 2008). It computes the importance of a token as the ratio of the number of documents in which it occurred to the total number of documents. We extended the idea of the DFS from unigrams to bigrams. BDFS is the weighted linear combination of the DFS for unigrams and bigrams of a sentence. Since bigrams encompass richer information and unigrams avoid problems with data sparseness, we chose a combination of both. The BDFS of a sentence s, $BDFS(s)$, is computed as

$$\frac{\alpha(\sum_{w_u \in s} DFS(w_u)) + (1-\alpha)(\sum_{w_b \in s} DFS(w_b))}{|s|}$$

where w_u are the unigram and w_b are the bigram tokens in sentence s. α is the weighting factor that is set to 0.3, after tuning on the development set. The DFS of a token w is calculated by the below equation,

$$DFS(w) = \frac{|\{d : w \in d\}|}{|D|}$$

where D is the set of documents in the topic and d is a document in D.

2.2.1.2 Category-Specific Features

The guided summarization task provides additional category and aspect information related to each topic. We devise two features, Category Relevance Score (CRS) and Category KL-Divergence (CKLD), to measure the category-specific importance (CSI) of each sentence. CSI is used along with general relevancy features to boost sentences having target category related information.

CRS computes the categorical relevance of a word, using the frequency of documents (CDFS) and the frequency of topics (TFS) in which the word occurred in a particular category. A linear combination of scores at both topic level and document level is assigned to each word. This feature is devised to utilize category information in guided summarization task and is specific to the task. CRS of a sentence s in category c, $CRS(s)$, is calculated as

$$\frac{\beta(\sum_{w \in s} TFS_c(w)) + (1-\beta)(\sum_{w \in s} CDFS_c(w))}{|s|}$$

where $TFS_c(w)$ and $CDFS_c(w)$ are computed by:

$$TFS_c(w) = \frac{|\{t : w \in t,\ \ \forall t \in c\}|}{|T_c|}$$

$$CDFS_c(w) = \frac{|\{d : w \in d,\ \ \forall d \in t \cap t \in c\}|}{|D_c|}$$

where T_c and D_c are the sets of topics and documents in category c, respectively.

CKLD is a category differential measure that calculates the importance of a word in the category as its KL divergence value of its distribution in the current category (c) to all the categories in the dataset (C). The greater the divergence from the total set C, the more informative the word is for category c. The probabilities are computed in terms of document frequencies. The CKLD value of a sentence s is given as:

$$CKLD(s) = \sum_{w \in s} p_c(w) log \frac{p_c(w)}{p_C(w)}$$

where p_c is calculated as the $CDFS$ of the word, namely:

$$p_c(w) = CDFS(w)$$

and p_C is calculated as:

$$p_C(w) = \frac{|\{d : w \in d, \ \ \forall d \in (\bigcup_{c \in C} D_c)\}|}{\sum_{c \in C} |D_c|}$$

2.2.2 Role of Named Entities

While CSI takes into account the importance of a word relative to the target category to summarize, it does not explicitly allow us a way to tailor the produced summaries to the *aspects* required for each category. We observed that many aspects of each category seek objective information such as the name of subjects or the location of a described event. These relate largely to WHO, WHERE and WHEN, hinting that we should look closely at sentences containing named entities.

To validate our observations, we experimented with 2 named entity related feature classes:

Top Entities for Topic (Top$_{topic}$) This module gathers the top n named entities (i.e. PERSON, LOCATION, etc.) from documents within a given document set, and assigns a unit score for sentences containing these entities.

Top Entities Corpus-Wide (Top$_{corpus}$) As Top$_{topic}$, but with documents drawn from the whole corpus.

Both feature classes are calculated as:

$$Top_x(s) = \begin{cases} 1 & \text{if } \exists w \in s, w \in W \\ 0 & \text{otherwise} \end{cases}$$

where W is the set of identified top named entities.

2.3 Training and SVR

Each sentence is scored with the features explained above. The features are automatically weighted by support vector regression, following the methodology described in (Bysani et al., 2009). We train the regression model using the ROUGE-2 (R2) similarity of sentences with human models as described in the paper. Data from TAC 2010 is used as the training corpus, and the trained regression model is used to predict the R2 scores of each sentence in the TAC 2011 test set.

2.4 Sentence Reranking

After each sentence has been scored, the maximal marginal relevance (MMR) (Carbonell and Goldstein, 1998) algorithm is used to re-rank and extract the best sentences to make up a 100-word summary. In our implementation, the MMR of a sentence *s* is computed as:

$$MMR(s) = Score(s) - R2(s, S)$$

where $Score(s)$ is the score predicted by the regression model, S is the set of sentences already selected to be in the summary from previous iterations, and $R2$ is the predicted ROUGE-2 score of the sentence under consideration (s) with respect to the selected sentences (S).

We have also experimented with computing the MMR value based on the term frequency/inverse document frequency (TFIDF) of the words in each sentence, but we found that the using the R2 values gives us better extracted sentences when evaluated against ROUGE.

2.5 Post-Processing

There are many text fragments found within the corpus that are not useful in a summary. These include news agency headers and the date of the article. These are removed automatically during post-processing from our generated summaries with the use of regular expressions.

After the post-processing step, the summary may get shortened to less than 100 words. The MMR sentence selection step is then repeated until the final summary reaches the intended length.

2.6 Update Summarization

In the update summarization task, all processing steps are similar to the generic summarization task, except for MMR re-ranking. We follow our previous work in (Lin et al., 2007) and modified the MMR equation so that it further penalizes a sentence in set B if it is similar to some other sentence

in set A. This is useful because an update summary should not contain duplicated content from the set A summary. We find that penalizing sentences that are very similar to those found in documents within set A helps to select sentences which are novel. The similarity is again measured with ROUGE-2. The modified MMR is shown as follows:

$$\begin{aligned} MMR(s) &= Score(s) - \lambda \cdot R2(s, S) \\ &\quad -\delta \cdot \max_{s' \in A} R2(s, s') \end{aligned}$$

The additional, final component of the equation tries to find a sentence in set A that is most similar to s in terms of predicted ROUGE-2. During our tuning phase, we found that $\lambda = 0.2$ and $\delta = 0.2$ give the best performance.

3 Submissions

The test dataset for the TAC 2011 Guided Summarization task consists of 44 topics, categorized into 5 categories. Each topic has 20 relevant documents, equally divided into 2 sets (Set A and Set B) based on the timestamps of the articles. Unlike regular, query-based summarization, each category has a specific template of aspects that defines the information the summarization system has to provide. The task is to generate two 100-word summaries: one for each set of a topic, answering all the aspects in the template. The summary for Set A is an ordinary summary, while the summary for Set B is an update summary that assumes that the user is already familiar with the documents in Set A. Repeated content from Set A should thus be omitted from Set B.

We submitted two runs for the summarization track this year. The first run (NUS1, ID:43) was created by choosing the best performing configuration on the training data in terms of ROUGE-2 score. The submission consists of five features: SP, SL, BDFS, CRS, and CKLD. The second run (NUS2, ID:17) is obtained by running the NUS1 implementation to retrieve up to 60% of the allowed summary length, and the remainder by incorporating an alternative set of features, consisting of the 2 named entity feature classes described in Section 2.2.2. The purpose of this is to accommodate informative, but non-ROUGE emphatic sentences into the summary for better coverage of the different aspects.

We provide the evaluation results from NIST in tables below. 50 runs were submitted for the task from 22 different teams, including 2 baselines from TAC. Table 1 reports the ROUGE scores and ranking of our submitted runs along with the best ROUGE scores among all participant systems. Similarly, Table 2 reports the manual evaluation results (average pyramid scores and overall responsiveness) of our runs. We also replicate the results of two NIST baselines in the tables for comparison – Baseline1 returns the top sentences in the most recent article untill the summary length (100 words) is reached, and Baseline2 is the output of the MEAD summarizer.[2]

	System	ROUGE-2	ROUGE-SU4
Set A	NUS1	0.13440 (1)	0.16519 (1)
	NUS2	0.12994 (2)	0.15984 (2)
	Baseline1	0.06410	0.09934
	Baseline2	0.08682	0.11749
	Best	0.13440	0.16519
Set B	NUS1	0.09581 (1)	0.13080 (1)
	NUS2	0.08855 (3)	0.12792 (3)
	Baseline1	0.05685	0.09449
	Baseline2	0.05903	0.09132
	Best	0.09581	0.13080

Table 1: Automatic evaluation results.

Results show that our runs have comfortably surpassed both the baselines in terms of ROUGE scores. NUS1 turned in the best ROUGE scores for both Set A and Set B. A drop in ROUGE scores is observed for Set B when compared to Set A across all summarizers, showing that update summarization is a harder task and hints at the need for more sophisticated methods.

A similar trend is observed in the manual evaluation results. Both the runs have scored better than the baselines by a sizable margin. Although our runs are tuned for ROUGE-2 scores in the regression model, the scores are still comparable to the best systems in terms of pyramid scores and overall responsiveness. NUS1 is behind the best systems in terms of average pyramid scores by only 0.003 and 0.006 in Sets A and B, respectively. It shows that ROUGE scores are correlated with content-based manual evaluation metrics. It is interesting to note

[2] http://www.summarization.com/mead/

	System	Avg.Pyramid	Over.Response
Set A	NUS1	0.474 (2)	3.068 (8)
	NUS2	0.468 (3)	3.091 (4)
	Baseline1	0.304	2.500
	Baseline2	0.362	2.841
	Best	0.477	3.159
Set B	NUS1	0.347 (3)	2.455 (12)
	NUS2	0.337 (7)	2.500 (7)
	Baseline1	0.237	2.091
	Baseline2	0.284	2.114
	Best	0.353	2.591

Table 2: Manual evaluation results.

that while NUS1 outperforms NUS2 in most evaluation measures, NUS2 is judged to be consistently better than NUS1 for overall responsiveness that is based on both linguistic quality and informativeness.

4 Post-Competition Experiments

We carried out a set of post-competition experiments to test the efficacy of each of the features used in our submissions. Given the time constraints, it is only feasible to perform the experiments using automatic evaluation measures, thus we only report the ROUGE scores of our experiments in this section.

The probabilities used for CKLD in Section 2.2.1 is computed in terms of the document frequencies. Later, we used term frequencies of the words in calculating its probability. The probabilities for this modified CKLD (*i.e.*, CKLD$_{tf}$) are computed as:

$$p_c(w) = \frac{f_c(w)}{\sum_{w_i \in W_c} f_c(w_i)}$$

$$p_C(w) = \frac{\sum_{c \in C} f_c(w)}{\sum_{c \in C} \sum_{w_i \in W_c} f_c(w_i)}$$

where $f_c(w)$ is the frequency of word w in category c, W_c is set of unique words in category c and C is the set of all categories.

The evaluation results of these post-competition experiments for both Set A and Set B are presented in Table 3. The first row is the original submission run (NUS1). Each row represents a change in configuration of NUS1 by removing one or more features. The final two rows in each set use CKLD$_{tf}$ instead of CKLD.

	Configuration	R2	RSU4
Set A	NUS1	0.13457	0.16502
	NUS1 – CRS	0.13463	0.16506
	NUS1 – CKLD	0.13702	0.16788
	NUS1 – CRS – CKLD	0.13392	0.16513
	NUS1 – BDFS	0.09419	0.13245
	NUS1 – SL	0.13523	0.16482
	NUS1 – SP	0.1262	0.15586
	NUS1(CKLD$_{tf}$)	**0.13796**	**0.16808**
	NUS1(CKLD$_{tf}$) – CRS	0.13725	0.16749
Set B	NUS1	0.09376	0.12875
	NUS1 – CRS	0.09126	0.12605
	NUS1 – CKLD	0.09359	0.12747
	NUS1 – CRS – CKLD	0.09209	0.1275
	NUS1 – BDFS	0.08388	0.12287
	NUS1 – SL	0.07996	0.11665
	NUS1 – SP	0.09565	0.12978
	NUS1(CKLD$_{tf}$)	0.09354	0.12763
	NUS1(CKLD$_{tf}$) – CRS	0.09209	0.12748

Table 3: Feature ablation tests with ROUGE-2 (R2) and ROUGE-SU4 (RSU4) scores.

A comparison of NUS1 against the configurations without the category-specific features (*i.e.*, CRS and CKLD) is helpful to illustrate the effect achieved of leveraging these category-specific features. The effect is more noticeable in Set B, which shows that both features are needed to achieve the optimal results and they are complementary to each other. Further, CKLD$_{tf}$ achieved higher scores compared to CKLD in Set A. The combination of CRS+CKLD$_{tf}$ helps in improving the scores from 0.13457 to 0.13796 for R2 and 0.16502 to 0.16808 for RSU4, as shown in bold.

Also, it is observed that BDFS is an important generic feature in the pipeline, and a large drop in ROUGE is observed without this feature. The effect of category-specific features is not significant in Set B. This suggests that the update summarization task requires a different approach beyond our use of category-specific information.

5 Conclusion

In this paper we present our SWING summarization system, and briefly described the two runs we submitted for the TAC 2011 Guided Summarization task. There are two unique characteristics for the guided summarization task — categories and aspects. We formulate our methods to address the

acquisition of category-specific information of each topic using the collective knowledge provided in the whole data set. Our initial efforts towards answering aspects make use of named entity information at the topical level. Both automatic and manual evaluation measures validate that our category-specific methods are very effective in producing a guided summary. In future work, we look towards devising more sophisticated features to capture aspect-specific information, as well as examining the integration of aspect-based scores with category-specific scores.

Acknowledgements

We would like to thank Dr Jian Su for her valuable assistance during the preparations for our submissions.

References

Praveen Bysani, Vijay Bharath Reddy, and Vasudeva Varma. 2009. Modeling novelty and feature combination using support vector regression for update summarization. In *Proceedings of ICON-2009: 7th International Conference on Natural Language Processing*.

Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual iInternational ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 335–336.

John M. Conroy, Judith D. Schlesinger, Judith D. Schlesinger, and Dianne P. O'Leary. 2010. CLASSY 2010: Summarization and metrics. In *Proceedings of the Text Analysis Conference*.

Hoa Trang Dang and Karolina Owczarzak. 2009. Overview of TAC 2009 summarization track. In *Proceedings of the Text Analysis Conference*.

H. P. Edmundson. 1969. New methods in automatic extracting. *Journal of ACM*, 16:264–285, April.

Ziheng Lin, Tat-Seng Chua, Min-Yen Kan, Wee Sun Lee, Long Qiu, and Shiren Ye. 2007. NUS at DUC 2007: Using evolutionary models of text. In *Proceedings of the Ducument Understanding Conference*, Rochester, NY, USA, April.

Frank Schilder and Ravikumar Kondadadi. 2008. FastSum: Fast and accurate query-based multi-document summarization. In *Proceedings of ACL-08: HLT, Short Papers*, pages 205–208, Columbus, Ohio, June. Association for Computational Linguistics.

Josef Steinberger, Hristo Tanev, Mijail Kabadjov, and Ralf Steinberger. 2010. JRC's participation in the guided summarization task at TAC 2010. In *Proceedings of the Text Analysis Conference*.

Vasudeva Varma, Praveen Bysani, Kranthi Reddy, Vijay Bharath Reddy, Vijay Bharath Reddy, Sudheer Kovelamudi, Srikanth Reddy Vaddepally, Radheshyam Nanduri, Kiran Kumar N, Santhosh Gsk, and Prasad Pingali. 2010. IIIT Hyderabad in guided summarization and knowledge base population. In *Proceedings of the Text Analysis Conference*.