From Extracting to Abstracting: Generating Quasi-abstractive Summaries

Zhuli Xie*, Barbara Di Eugenio†, Peter C. Nelson†

*Applications and Software Research Center, Motorola
1295 E. Algonquin Road, Schaumburg, Illinois 60173, U.S.A.
Zhuli.Xie@motorola.com

†Department of Computer Science, University of Illinois at Chicago
851 S. Morgan Street, Chicago, Illinois 60606, U.S.A.
{bdieugen,nelson}@cs.uic.edu

Abstract

In this paper, we investigate quasi-abstractive summaries, a new type of machine-generated summaries that do not use whole sentences, but only fragments from the source. Quasi-abstractive summaries aim at bridging the gap between human-written abstracts and extractive summaries. We present an approach that learns how to identify sets of sentences, where each set contains fragments that can be used to produce one sentence in the abstract; and then uses these sets to produce the abstract itself. Our experiments show very promising results. Importantly, we obtain our best results when the summary generation is anchored by the most salient Noun Phrases predicted from the text to be summarized.

1. Introduction[1]

Automatic text summarization has attracted more and more attention in recent years, e.g., see the official website of the Document Understanding Conferences[2] (DUC). Although many summarization systems already show impressive results, state-of-the-art has not progressed beyond summaries composed of whole sentences extracted from the sources, as also discussed in an overview of DUC-2005 (Dang, 2005). Producing human-like summaries would seem to require radical advances in both text comprehension and generation. On the contrary, as others (Barzilay and McKeown, 2005; Soricut and Marcu, 2006), we believe it is possible to produce more human-like summaries via text-to-text generation techniques. In this paper, we present our approach to generating *quasi-abstractive summaries*, a new type of summary produced by text-to-text generation techniques. Quasi-abstractive summaries are composed not of whole sentences, but of fragments that are extracted from the original text. Hence, the sentences that appear in a quasi-abstractive summary are in general different from the sentences that appear in the original document, like the sentences in a human-written abstract.

We claim that such a quasi-abstractive summary is better than an extractive summary. In fact, our quasi-abstractive summaries (anchored by salient topics, see Section 4.2.2.) are significantly better than two different types of extractive summaries: the average cosine similarity between the quasi-abstractive summaries and the human-written abstracts is 29.4% better than the lead-based extractive summaries, and more importantly, 13.9% better than the extractive summaries generated by ADAMS, our adaptive summarizer based on gene expression programming (Xie et al., 2006). The ROUGE scores (Lin, 2004) are higher as well: the ROUGE-1 and ROUGE-2 are 31.5% and 64.3% better than the lead-based summaries, and 2.8% and 19.3% better than ADAMS respectively.

Our approach to generating quasi-abstractive summaries comprises two main steps: 1) train a classifier that can identify *candidate sentence sets* (CSS's). Each CSS contains all the sentences from the document that have one or more fragments in common with a specific abstract sentence; 2) use the learned model to build the CSS's for a new document, and generate the summary by using the sentences created from the CSS's. In the second step, we enhance simple n-gram probabilities by anchoring them via the most salient NPs, as predicted by using several features of the NPs, including centrality (Wasserman and Faust, 1994). The innovative aspects of our approach are: generating candidate sentence sets and using fragments from sentences in those sets to generate the summary; and approximating the unfolding of topics in the abstract by means of NP centrality. In this paper, we present a full fledged end-to-end system, that demonstrates that our approach gives very promising results. In the conclusions, we will discuss alternative methods we are planning to try for each of the two steps.

2. Quasi-abstractive summaries

Extractive summaries are composed of sentences selected from the source documents, and included in the summary in their entirety. Consider the top part of Figure 1. It shows an original abstract and the summary generated by ADAMS, our extractive summarizer. The overlap[3] between the two consists of only two phrases. If counted in words, the overlap comprises only 21% of the abstract. However, in general there are many sentences in the document that more closely correspond to the sentences in the abstract. (Jing, 2002) showed that six operations are often seen in human-written abstracts: reduction, combination, syntactic transformation, lexical paraphrasing, generalization or specification, and reordering. Those operations identify the

[1]The work reported here was completed while the first author attended the University of Illinos at Chicago.

[2]http://www-nlpir.nist.gov/projects/duc/index.html

[3]In this paper, the "overlap" is defined as a matching sequence of adjacent words between two sentences. One sentence may have multiple overlaps with other sentences.

Two sentences from a human written abstract:

A1 We introduce the bilingual dual-coding theory as a model for bilingual mental representation.

A2 Based on this model, lexical selection neural networks are implemented for a connectionist transfer project in machine translation.

Extractive Summary (by ADAMS):

E1 We have explored an information theoretical neural network that can acquire the verbal associations in the dual-coding theory.

E2 The bilingual dual-coding theory partially answers the above questions.

Candidate sentence set for A1:

S1 The bilingual dual-coding theory partially answers the above questions.

S2 There is a well-known debate in psycholinguistics concerning the bilingual mental representation...

Candidate sentence set for A2:

S3 We have explored an information theoretical neural network that can acquire the verbal associations in the dual-coding theory.

S4 It provides a learnable lexical selection sub-system for a connectionist transfer project in machine translation.

Figure 1: Phrase overlaps with abstract sentences

sources of up to 81% of the abstract sentences. For example, as shown in the bottom part of Figure 1, sentences **S1**, **S2**, **S3**, and **S4** from the text, cover more phrases from sentences **A1**, **A2** in the abstract. In this case the overlap rises to 64% of the abstract. If those overlapping phrases can be identified from the text to be summarized, we will advance our path to generating abstractive summary sentences one big step; the rest is to feed those phrases to a language generation module to produce the final sentence. Doing so will definitely improve cosine similarity, which is a bag-of-words metric, since the more overlap two texts have, the higher their similarity value. The summarization community has also been using ROUGE-n metrics (Lin, 2004). More overlap between abstract and summary will also improve the ROUGE-n metrics, since ROUGE-n are recall measures. Our approach to generating quasi-abstractive summaries consists of the following steps:

Learn model that can identify *Candidate Sentence Set (CSS)*. In turn, this step consists of:

- **1a. Label**: generate patterns of correspondence between sentences in the abstract and sentences in the text to be summarized. For each sentence A_i in the abstract, find sentences from the text that contain its fragments. These sentences comprise A_i's CSS, represented by the sets of points in different shapes on the right side of diagram (I) in Figure 2.

- **1b. Train classifier** to identify the CSS's. The problem is not straightforward, because the classes of the target feature are variable. In fact, the number of sentences in an abstract varies across different documents and it is not clear on what it depends. For instance, the length of the 178 abstracts from CMP-LG ranges from 2 to 14 sentences. Therefore, when given a new text, we do not know how many classes there will be. Moreover, clustering would not work, since clustering algorithms tend to put similar instances together, but sentences in a CSS may not be similar to each other at all. In Figure 1 sentences S3, S4 should be in A2's CSS; however, S3 and S4 have nothing in common except for the function words "a/an" and "the". Hence, we built a decision-tree classifier to recognize whether a pair of sentences belongs to the same CSS (see diagram (II) in Figure 2). This classifier uses 9 shallow statistical features, which include the position of each sentence within the document, their cosine similarity, the sentence pair entropy, etc.

Generate summary for a new document. Also this step comprises two substeps.

- **2a Generate CSS's.** Sentence pairs are built as the cross-product of all the sentences in the text. Then those pairs are run through the classifier. The pairs classified as "1", which means both sentences are in the same CSS, will be merged to form each of the final CSS's. This step is shown in diagram (III) in Figure 2, and results in a set of CSS's ordered by the lowest sequence indices of the sentence pairs in each of the CSS's.

- **2b Realize summary.** Each summary sentence is generated on the basis of one CSS. For each CSS, we compute n-gram probabilities for the words that appear in the document sentences belonging to that CSS. The actual sentences are generated according to two models, *SQa* (Simple Quasi-abstractive) and *QaST* (Quasi-abstractive with Salient Topics).

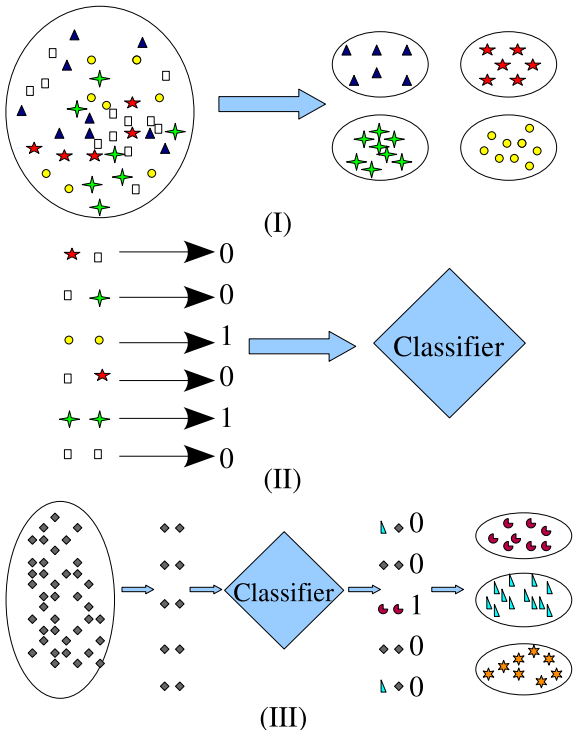

Figure 2: Candidate Sentence Sets Discovery Diagram

3. Learn the CSS model

3.1. Step 1a: Label

To train the CSS generator, we first need to create our training data, namely, the CSS's for our corpus. We build on the detailed study of decomposition of abstract sentences in (Jing, 2002). Jing trained a HMM to identify fragments. We, however, adopt a simpler approach based on string overlap, since our goal is to find phrases which cover a summary sentence as much as possible, not to account for how the decomposition is performed by humans. Our approach is shown in Algorithm 1. For each sentence A_i in the abstract, this algorithm returns a CSS_i that contains all the sentences S_j that overlap with A_i (L_j is the j-th fragment in sentence S_j overlapping with A_i, in the source document D). In our experiments, we use the corpus CMP-LG comprising 178 documents. This corpus contains 863 abstract sentences in total. If we exclude overlaps that consist of one single word, an abstract sentence will correspond to 1 to 15 sentences (overlap length ≥ 2), or 1 to 8 sentences respectively (overlap length ≥ 3). It turns out that a significant portion (70.8%) of abstract sentences are composed of fragments of length ≥ 2, which can be found in the text to be summarized. Since in the generation stage we will use n-gram probabilities with $n \geq 2$, we set the minimal length of overlap to two in Algorithm 1. We now have a training corpus of documents and their CSS's, to be used to train a model that automatically identifies CSS's.

Algorithm 1 Candidate Sentence Set Generation

procedure GetCandidateSentenceSet(A_i)

1: CSS_i:=$\emptyset$
2: **for** each sentence A_i in an abstract **do**
3: T_i:=Tokenize(A_i);
4: T_i:=RemovePunctuation(T_i);
5: **end for**
6: **repeat**
7: (L_j, S_j):=FindLongestCommonToken(A_i, D);
8: T_i:=$T_i.Remove(L_j)$;
9: CSS_i:=$CSS_i \cup S_j$;
10: **until** $L_j = \emptyset || T_i = \emptyset$

3.2. Step 1b: Train the CSS classifier

We reformulate our classification task as: Given a set of documents, where all CSS's have been labelled, transform each document into a sentence pair set where each instance is represented by a feature vector and the target feature is whether the pair belongs to the same CSS. This idea is shown in Figure 2, diagram II.

For a text with a sentences, in total $\frac{a(a-1)}{2}$ pairs will be generated. We encode each pair as a vector of 9 shallow statistical features, which include: *location of paragraph* containing each sentence; *position of each sentence* within the document; *length of each sentence*, measured in number of words $l(s_i)$, and normalized by the Sigmoid function:

$$LN_i = \frac{1 - e^{-\alpha}}{1 + e^{-\alpha}}, \alpha = \frac{l(s_1) - \mu(l(s))}{\sigma(l(s))}$$

where $\mu(l(s))$ is the average length of sentences, and $\sigma(l(s))$ is the standard deviation of the sentence lengths; *cosine similarity* between the two sentences; *length of their longest common n-gram*; *sentence pair entropy*

$$EN = -\sum_{i=1}^{k} \{Freq(w_i) \cdot log[Freq(w_i)]\}.$$

where $w_i \in s_1, s_2$, w_i is a content-word, and $Freq(w_i)$ is the frequency of w_i in that document.

The features above by no means are complete. We chose them simply because they are easily obtained and domain independent. Other linguistic motivated features may be considered in our future work.

The CMP-LG corpus generated a total of 8,152,713 instances. However, this collection of instances is extremely imbalanced, since the positive instances only amount to 0.398%. Extremely imbalanced data sets cause the dominant class of the target features to be favored. Two whole workshops (Japkowicz, 2000b; Chawla et al., 2003) and a full special issue of the SIGKDD Explorations Newsletter (Chawla et al., 2004) have been devoted to this problem. A common technique to address this problem in machine learning is to replicate instances of the minority class to reduce the unfair preference (Chawla et al., 2002; Japkowicz, 2000a; Akbani et al., 2004). This is the approach we followed as well, hence, we replicated the instances in the minority class to reduce the unfair preference. In the experiments reported below, the positive instances are boosted to about 11% of the total data. The augmented data set was input to three classifiers: Naïve Bayes (NB), Support Vector Machines (SVMs) and decision trees (DTs). We used three freely available packages, Borgelt's NB classifier (Borgelt,

1999), SVM-light (Joachims, 2002) and YaDT (Ruggieri, 2004) respectively. We ran three way cross validation, where in each run the classifier is trained on 2/3 of the data and tested on the remaining third. Table 1 reports the average results across the three runs, obtained via NB, SVMs[4], and decision trees respectively. We can see from Table 1 that DTs performs the best among the three classifiers in terms of F-measure; and the precision is well over 0.6, which give us some confidence that the DTs can accurately classifies positive instances. In Step 1b we tried all three classifiers to predict the CSS's and hence generate summaries for unseen testing documents. It turns out NB doesn't do so well; and SVMs and DTs perform comparably. We finally used DTs because it has better F-measure, and also because SVMs take much longer time to train.

Classifier	NB	SVMs	DTs
Accuracy	0.867	0.901	0.895
Precision	0.385	0.706	0.634
Recall	0.345	0.172	0.296
F-measure	0.364	0.276	0.404

Table 1: CSS Discovery Results

Boosting affects the distribution of instances in the test set as well. Hence, the classifier will identify for us correct positive instances, but also report some false positive instances. However, we are actually only interested in the positive instances it labels, not in the pure classification performance of the classifier itself. Without boosting, the classifier would just not give us any positive instances at all. Also note that the duplicates (of positive instances) are removed when we form the Candidate Sentence Sets.

4. Generate summary for new document

4.1. Step 2a: Generate CSS's

The first substep is to generate CSS's for the new document. Again, here we use the corpus CMP-LG comprising 178 documents. Also in this case, we use 3-way cross-validation: the classifier derived in one of the three runs at the previous step by using $2/3$ of the documents is used to generate the CSS's for documents in the test set ($1/3$ or the rest of the corpus) for that run. We first build the cross-product of all the sentences in it and the feature vectors for these pairs; then those vectors are run through the decision tree classifier just described.

Those pairs which are labelled as "1" will then be merged to form the final CSS's. One pair p will be merged into set s if and only if the cross product, of the sentences from p and the sentences from s, contains only positive sentence pairs. In other words, in the sentence set after merging, any two sentences must form a positive pair.

Since the indices of sentences are sequentially labelled across the document, and so are the pairs, the formation of the CSS's will also exhibit a natural order, i.e., the first CSS will contain a fragment f_1 which appears earlier in the source document than any fragments in the second CSS; and so on for the rest of CSS's.

[4]We experimented with four different kernels. The best results were obtained with the radial basis function kernel.

4.2. Step 2b: Realize Summary

As mentioned earlier, once we have obtained the CSS's for a document to be summarized, we use two methods to realize the summary: Simple Quasi-abstractive (SQa) and Quasi-abstractive with Salient Topics (QaST). Both are based on computing n-grams, even if in QaST those n-grams are anchored in salient NPs. We did not adopt a more sophisticated generation approach because, as often happens in a text-to-text generation circumstance, our application does not have access to the kind of information that generic NLG systems require as input (Soricut and Marcu, 2005), e.g., deep subject-verb or verb-object relations as required by Penman (Matthiessen and Bateman, 1991) or FUF (Elhadad, 1991), or the shallow syntactic relations needed by HALogen (Langkilde, 1998).

We present each of our methods in turn.

4.2.1. SQa realization

The SQa method uses n-gram probabilities to generate the actual sentences in the abstract. A simple quasi-abstractive summary is generated as shown in Algorithm 2. To generate a new sentence, first we compute n-gram probabilities based on each CSS predicted by the CSS generator. Then given a word sequence which begins with a start-of-sentence symbol, a new word, which has the highest probability given the previous $N-1$ words, is appended to the sentence until an end-of-sentence symbol is encountered. This process is repeated for each CSS.

Algorithm 2 SQa Summary Realization

```
procedure GenerateSQA()
    /S/: start-of-sentence symbol
 2: /E/: end-of-sentence symbol
    NS_i: the i-th new sentence
 4: for i = 1 to n do
      NGram_i:=BuildNGramModel(CSS_i);
 6:   t:= /S/w_1 ... w_{N-2};
      NS_i:=t;
 8:   w_i:=null;
      while w_i ≠ /E/ do
10:     w_i:=NGram_i.getNext(t);
        t:= t.removeFirstWord();
12:     t:= t.appendWord(w_i);
        NS_i:= NS_i.appendWord(w_i);
14:   end while
    end for
```

4.2.2. QaST realization

The n-gram based language generation model is built solely on statistical information of word sequences from a text. Even if the model is "filtered" by the CSS's so to speak (i.e., only the conditional probabilities of words that appear in that CSS are used), it is very difficult, if not impossible, to generate a good summary for that text without additional knowledge. For a summary to be a summary, it should include the most salient information from the source and exclude the redundant. For a text document to be informative, it should address some topics and regard them as the foci. In our previous work (Xie, 2005), we proposed

a new text feature, *noun phrase centrality*, which represents the prominence of a noun phrase within a certain text. The value of the NP centrality is obtained by applying centrality measures upon a NP network. The NP network is constructed by connecting NPs at the utterance level. We showed in (Xie, 2005) that the salient topics in the forms of NPs from the text can be predicted highly accurately (F-measure = 0.883) by using decision trees that include several features of NPs, including NP centrality.
Here we extend the basic summary generator by anchoring the generation of a new summary sentence in a salient NP predicted in the way described in (Xie, 2005). Our augmented summary generator comprises:

1. Generate salient NPs via topic prediction
2. Sort predicted salient NPs according to their lengths. Longer NPs cause less variation in the sentence generation process.
3. For each sentence S_i to be generated, traverse the lists of NPs and of CSS-based n-gram probabilities in parallel (these sets of probabilities are ordered, since they're derived from the CSS's which are ordered). Generate S_i by using the highest ranked NP which has not been used yet, and the first model where this NP is found. If a certain NP does not exist in any n-gram model, it is discarded.

A sentence is generated by using the highest ranked NP which has not been used and a specific submodel as follows: the NP constitutes the starting sequence $w_i w_{i+1} \dots w_{i+k}$; repeatedly, words are prepended to the sequence at the front according to

$$\arg\max_{i-1} P(w_{i-1}|w_i^{i+N-2})$$

until $/S/$ has been encountered; then repeatedly, words are appended to the sequence at the end according to

$$\arg\max_{i+k+1} P(w_{i+k+1}|w_{i+k-N+2}^{i+k})$$

until $/E/$ has been found.
To estimate the complexity of our model, the first thing to see is that we are actually performing depth-first search here. The complexity of a general depth-first search is $O(b^m)$, where b is the branching factor and m is the maximal depth. In our situation since the CSS's is not that big and hence the n-gram obtained from the CSS's will have $b = 1$ in most cases. The actual running time was very short although we did not record it.

5. Experiments and Results

We compare four different models: two baseline models, a lead-based extractive summary (Lead), and ADAMS, the extractive summarizer we developed in previous work (Xie et al., 2006); and two models, *SQa* (Simple Quasi-abstractive) and *QaST* (Quasi-abstractive with Salient Topics), built according to the methodology just described.

Lead: Selects the first sentence from each of the first m paragraphs.

ADAMS: Selects the top m sentences ranked according to a sentence ranking function ADAMS has previously learned.

SQa: Uses n-gram probabilities over the first m discovered candidate sentence sets to generate the sentences in the summary.

QaST: Anchors the choice of the specific set of n-gram probabilities in salient topics. Stops after m sentences have been generated.

Let m be the number of sentences in the summary of the text to be summarized. In real life, the user would set the value of m. In our experiments, in order to compare the machine-generated summary with the human-written abstract, we set the value of m to be the number of sentences in the abstract. Note that, since we limit the number of sentences rather than words, the generated summary can be longer or shorter than the abstract. Moreover, we intend to lift this simplification in future experiments. One of the best sentences generated by QaST is shown in Figure 3. The NP "the plan construction" is the starting word sequence for that new sentence. Using the sentence generation method described above, the resulting sentence is almost a perfect one which combines two original sentences from the text. However, undoubtedly, using the n-gram model only does not guarantee the grammaticality of the generated summary sentences, as shown e.g. by the missing ")" in this example. It remains for us to further explore this problem in our future work.
The summaries generated from those four models are compared with the corresponding abstracts. The evaluation metrics are the mean cosine similarity value, ROUGE-1, and ROUGE-2 scores. The results are listed in Table 2 — we report the best results, obtained with $n = 4$ for computing n-gram probabilities. It turns out that the average cosine similarity and ROUGE-2 score of the SQa summaries to the abstracts is 0.293 and 0.078 respectively, which are even lower than the performance of the Lead system respectively (0.330 and 0.098). Our previous experiments with ADAMS had achieved a much better performance of in all three metrics: 0.375, 0.426, and 0.135 (+13.6%, +27.9%, and +37.8% improvement over Lead respectively). Using simple n-grams over the candidate sentence sets is too simplistic. Our QaST model was given additional knowledge so that its performance is much better than the SQa model. The three evaluation scores for summaries produced by QaST is 0.427, 0.438, and 0.161, which are 29.4%, 31.5%, and 64.3% better than the Lead system; and 13.9%, 2.8%, and 19.3% better than ADAMS. Except the 2.8% improvement in ROUGE-1, these differences are statistically significant by using two sample t-tests on the results.

	Lead	ADAMS	SQa	QaST
Cosine Similarity	0.330	0.375	0.293	**0.427**
ROUGE-1 Score	0.333	0.426	0.334	**0.438**
ROUGE-2 Score	0.098	0.135	0.078	**0.161**

Table 2: Experimental Results

New In collaborative expert-consultation dialogues, two participants (executing agent and the consultant bring to the plan construction task different knowledge about the domain and the desirable characteristics of the resulting domain plan.

Original In collaborative expert-consultation dialogues, two participants (executing agent and consultant) work together to construct a plan for achieving the executing agent's domain goal. The executing agent and the consultant bring to the plan construction task different knowledge about the domain and the desirable characteristics of the resulting domain plan.

Figure 3: An Example Sentence Generated by the QaST Model

6. Related Work

(Jing, 2002) is the first study of decomposition of abstract sentences which goes well beyond previous approaches based on sentence alignment (Kupiec et al., 1995; Marcu, 1999; Teufel and Moens, 1997), where an abstract sentence is considered to be re-written from a single sentence from the text. Jing showed that an abstract sentence may consist of fragments found in multiple sentences, and developed an HMM to find those fragments; however, she did not use the patterns found during the decomposition process to predict what fragments shall be used to compose a sentence in the summary, as we do here. Our approach is very similar to *Sentence Fusion* (Barzilay and McKeown, 2005): a new sentence is generated by finding the common theme from a set of similar sentences coming from different documents and computing how the sentences can be fused, based on their dependency tree representations. The similar sentences are found from multiple source articles. However, our approach differs from *sentence fusion* in that our system generates a new sentence from a CSS. Each CSS contains a set of sentences from a single source article, which do not necessarily have a common theme. More recently, (Wan et al., 2007) proposed a *Global Revision* approach to improve the coherence of the summaries. They used a Maximal Spanning (Dependency) Tree to structure a new sentence and combined with four-gram language model for surface realization. This approach is the most similar one to ours that we have seen so far. Our approach differs from theirs in two aspects. First, our goal is to improve the overall similarity of the summaries to human-written abstracts; while their goal is to improve the overall fluency. Second, we provided a complete end-to-end system of constructing a quasi-abstractive summary, from building a CSS model to find the input content, to combining the CSS with a salient NP to finally produce a new sentence; while their approach only concerned about using a set of input words/phrases to build a new sentence without considering how to obtain the input.

Other approaches relevant to ours, since they all aim at deriving summaries that are not extractive, include sentence compression (Knight and Marcu, 2000), and models for headline generation (Banko et al., 2000; Soricut and Marcu, 2006). Our approach differs from theirs in that we try to simulate the process of composing a new sentence by finding the correspondence between fragments of sentences in the abstracts and sentences in the documents, rather than directly manipulating the sentences in the document. (Soricut and Marcu, 2006) also exploits *topic keywords* extracted from the document.

7. Conclusions

In this paper we have proposed a new type of machine generated summary, the quasi-abstractive summary. Quasi-abstractive summaries differ from human-written abstracts in two aspects: on the surface we do not add new words or phrases; and in the core we have not tried to apply text understanding. Despite these limitations, this new type of summary goes well beyond the traditional extractive summaries, as it uses fragments, but not whole sentences, from the source. We showed that the simpler model, that uses N-gram models over the CSS's returned by a decision tree classifier, is not effective. However, when we anchored the N-gram model with the NPs which were predicted to be salient topics in the text to be summarized, we obtained a much better outcome — the average cosine similarity value improved by 29.4% when compared to Lead, and 13.9% when compared to ADAMS. ROUGE-1 score is 31.5% better than Lead and 2.8% better than ADAMS. ROUGE-2 score has the most significant improvement, which is 64.3% higher than Lead and 19.3% higher than ADAMS.

We are currently investigating additional methods for each of the four substeps in our approach. For example, in step 1b, to obtain a better CSS's classifier, we are investigating other ways, such as cost-sensitive classifiers (Domingos, 1999; Ting, 2002; Turney, 2000), to improve the performance of classifier without changing the class distribution. We are also investigating the features used for classification in step 1b, by considering the selection metrics discussed in (Zheng et al., 2004). Improving the performance of the CSS's classifier may engender better results in the summary generation step. And for step 2b, generation, we are looking into models that ensure the grammaticality of the resulting sentences. As a first pass we intend to impose filters on the outcomes of this step, e.g. by parsing each sentence as it is generated to catch major grammar mistakes. The approach presented in (Wan et al., 2007) seems to be a good candidate and we will see what results we can get by combining it with ours. In addition, sentences generated by our summarizer are generally rather long; a simple cutoff may be effective.

8. Acknowledgments

This work was supported by award N000140010640 from Office of Naval Research, and partly awards IIS-0133123 and ALT-0536968 from the National Science Foundation.
We would also like to thank anonymous reviewers for their valuable feedback on this paper.

9. References

Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz. 2004. Applying support vector machines to imbalanced datasets. In *Proceedings of ECML 2004*, pages 39–50.

Michele Banko, Vibhu O. Mittal, and Michael J. Witbrock. 2000. Headline generation based on statistical translation. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 318–325.

Regina Barzilay and Kathleen R. McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.

Christian Borgelt. 1999. A naive bayes classifier plug-in for dataengine. In *Proceedings of the 3rd Data Analysis Symposium*, pages 87–90.

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence and Research*, 16:321–357.

Nitesh V. Chawla, Nathalie Japkowicz, and Aleksander Kotcz, editors. 2003. *Proceedings of the ICML'2003 Workshop on Learning from Imbalanced Data Sets*.

Nitesh V. Chawla, Nathalie Japkowicz, and Aleksander Kotcz. 2004. Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explorations Newsletter*, 6(1):1–6.

Hoa Trang Dang. 2005. Overview of duc 2005. In *Proceedings of the Document Understanding Conference (DUC05)*.

Pedro Domingos. 1999. Metacost: a general method for making classifiers cost-sensitive. In *Proceedings of the Fifth ACM SIGKDD*, pages 155–164, New York, NY, USA. ACM Press.

Michael Elhadad. 1991. FUF user manual - version 5.0. Technical Report CUCS-038-91.

Nathalie Japkowicz. 2000a. The class imbalance problem: Significance and strategies. In *Proceedings of the 2000 International Conference on Artificial Intelligence (IC-AI'2000)*, volume 1, pages 111–117.

Nathalie Japkowicz, editor. 2000b. *Proceedings of the AAAI'2000 Workshop on Learning from Imbalanced Data Sets, AAAI Tech Report WS-00-05*. AAAI.

Hongyan Jing. 2002. Using hidden markov modeling to decompose human-written summaries. *Computational Linguistics*, 28(4):527–543.

Thorsten Joachims. 2002. *Learning to Classify Text using Support Vector Machines*.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization - step one: Sentence compression. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 703–710. AAAI Press / The MIT Press.

Julian Kupiec, Jan O. Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *Proceedings of the 18th ACM-SIGIR Conference*, pages 68–73.

Irene Langkilde. 1998. An empirical verification of coverage and correctness for a general-purpose sentence generator.

Chin-Yew Lin. 2004. Rouge: a package for automatic evaluation of summaries. In *Proceedings of the ACL-04 Workshop on Text Summarization Branches Out (WAS 2004)*, pages 74–81.

Daniel Marcu. 1999. The automatic construction of large-scale corpora for summarization research. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 137–144.

Christian M. I. M. Matthiessen and John A. Bateman. 1991. *Text generation and systemic-functional linguistics: experiences from English and Japanese*.

Salvatore Ruggieri. 2004. Yadt: Yet another decision tree builder. In *Proceedings of 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004)*, pages 260–265.

Radu Soricut and Daniel Marcu. 2005. Towards developing generation algorithms for text-to-text applications. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 66–74, June.

Radu Soricut and Daniel Marcu. 2006. Stochastic language generation using widl-expressions and its application in machine translation and summarization. In *Proceedings of the 44rd Annual Meeting of the Association for Computational Linguistics*, pages 1105–1112, July.

Simone Teufel and Marc Moens. 1997. Sentence extraction as a classification task. In *Proceedings of Workshop on Intelligent Scalable Text Summarization (ACL97/EACL97)*, pages 58–65.

Kai Ming Ting. 2002. An instance-weighting method to induce cost-sensitive trees. *IEEE Transactions on Knowledge and Data Engineering*, 14(3):659–665.

Peter D. Turney. 2000. Types of cost in inductive concept learning. In *Proceedings of the ICML'2000 Workshop on Cost-Sensitive Learning*, pages 15–21.

Stephen Wan, Robert Dale, Mark Dras, and Cécile Paris. 2007. Global revision in summarisation: Generating novel sentences with prim's algorithm. In *Proceedings of PACLING 2007 - 10th Conference of the Pacific Association for Computational Linguistics*, pages 226–235, Melbourne, Australia.

Stanley Wasserman and Katherine Faust. 1994. *Social Network Analysis : Methods and Applications*. Cambridge University Press.

Zhuli Xie, Barbara Di Eugenio, and Peter C. Nelson. 2006. Adaptive learning in machine summarization. In *Proceedings of the Nineteenth International FLAIRS Conference*, pages 180–181, May.

Zhuli Xie. 2005. Centrality measures in text mining: Prediction of noun phrases that appear in abstracts. In *Proceedings of the ACL Student Research Workshop*, pages 103–108, June.

Zhaohui Zheng, Xiaoyun Wu, and Rohini Srihari. 2004. Feature selection for text categorization on imbalanced data. *SIGKDD Explorations Newsletter*, 6(1):80–89.