Autotext Compactor Based on Sentence Tokenization and Extraction

Komal Sarode, Ms. Suvarna Hajare

Department of Computer Science & Engineering Jhulelal Institute of Technology, Nagpur

I. INTRODUCTION

Today's, a great deal of information comes from the Internet in a textual form. The challenge of finding relevant documents on the web is mainly handled by information retrieval techniques utilized in search engines such as Google, Bing, Yahoo, etc. Search engines usually return thousands of pages for a single query, and even the use of sophisticated ranking algorithms can't provide us the exact information we are looking for. A typical user goes through the top-ranked pages and tries to find the relevant pieces of information he or she is interested in, manually. Obviously, a short summary of the retrieved pages would be very helpful in such situations. In general, construction of summaries is an ideal way to cope with the information overload. A summary is a shortened version of a text that contains the main points of the original content.

The automatic construction of abstracts from the texts of documents has gained increasing interest in recent years. Abstracts can help a reader to grasp the central subject matter of a document without looking at the full document. High-quality text summarization would improve document search and browsing in a variety of contexts. For example:

Over the Internet. With such an overwhelming amount of information on the Internet, retrieving and browsing relevant documents in an efficient manner becomes extremely important to Internet users. Abstracts could help users to quickly judge the relevance of documents, and therefore, not to waste time accessing and browsing documents that are not at all interesting to them.

Over Digital Libraries. As documents become increasingly available in electronic form, the effort of storing and indexing them leads to the construction of digital libraries. To provide abstracts for documents in digital libraries so that the documents can be efficiently organized and retrieved, we need to explore automatic techniques since the number of documents that are to be included in digital libraries is enormous and manually constructing abstracts for all documents is impossible.
From hand-held devices. Due to the limited display space on hand-held devices such as Personal Digital Assistants, document condensation is very desirable. Many abstracts are also informative, including material such as main results and conclusions so that a reader can gather the most important information without having to refer to the full document. The indicative function is regarded as essential and the informative function is desirable. In practice, most automatically constructed abstracts contain at least some informative material. Abstracts that are produced manually almost always summarize the important points of the central subject matter of a text; such abstracts are called generic abstracts. Many automatic summarization systems can also construct user-focused or query-based summaries, which do not concentrate on the central topic of a text but contain material that is most important based on a user’s particular interests. A user’s queries to a search engine or to a question-answering system are often regarded as the indication of the user’s particular interests. Much of the reported work has been concerned with producing summaries for newspaper articles or scientific papers. Some summarization systems are domain-dependent, meaning that they can only handle texts from a specific domain. Such systems often rely on the knowledge of texts in that domain (for example, text structures, keywords or key phrases) to find the important information. Some systems are domain-independent, meaning that they do not have strict restrictions on the domain of documents. Abstracts might be constructed from a single document; they might also be constructed from multiple documents. In the latter case, the abstract will summarize the important points in all the source documents (in practice, the source documents are often a cluster of documents that report on the same event). When multi-document summarization is concerned, the source documents can be in a single language or they can be in different languages.

II. PROPOSED WORK

a. Sentence decomposition

The task of summary sentence decomposition is to deduce whether a summary sentence is constructed by reusing the original text and identifying the reused phrases. The benefits of solving the decomposition problem are two-fold. First, large corpora for training and evaluating our cut-and-paste summarizer can be built from the decomposition result. By linking human-written summaries with original texts, we can mark exactly what phrases humans cut from the original document and how the phrases were pasted together to produce the summary. Second, the decomposition result also provides large corpora for extraction-based summarizers. By aligning summary sentences with original document sentences, we can automatically annotate the most important sentences in an input document, therefore constructing large corpora for training and evaluating the extraction-based summarizers.

While decomposition is useful, it is also difficult. The phrases coming from the original document can be at any granularity, from a single word to a complicated verb phrase to a complete sentence. Therefore, identifying the boundary of phrases is a complex issue. We proposed a Hidden Markov Model [Baum, 1972] solution to the decomposition problem. There are three steps in this process. First, we formulate the decomposition problem to an equivalent problem; that is, for each word in a summary sentence, we find a document position that it most likely comes from. This is an important step since only after this transformation are we able to apply the Hidden Markov Model to solve the problem. Second, we build the Hidden Markov Model based on a set of general heuristic rules that we have observed from the text reusing practice of humans. This is actually quite unconventional in applications that use Hidden Markov Models since Hidden Markov Models usually require a training corpus to compute transition probabilities, but we believe it is appropriate in our particular application. The evaluations show that this unconventional Hidden Markov Model is effective for decomposition. In the last step, a dynamic programming technique, the Viterbi algorithm is used to efficiently find the most likely document position for each word in a summary sentence and finally find the best decomposition for a summary sentence.

b. Sentence Reduction

From the decomposition results, we can build a corpus for the sentence reduction module by selecting the summary sentences that were generated by removing phrases from document sentences. The corpus contains summary sentence-document sentence pairs. Each summary sentence is marked with the information of phrase origins that is provided by the decomposition output. In a cut-and-paste summarization system, the role of sentence reduction is to perform the editing operation of removing extraneous phrases from an extracted sentence. It can remove material at any granularity: a word, a prepositional phrase, a noun phrase, a verb phrase, a gerund, a to-infinitive, or a clause. We use the term “phrase” here to refer to any of the above sentence components that can be removed in the reduction process. Reduction improves the conciseness of automatically generated summaries, making it brief and on target. It
can also improve the coherence of generated summaries, since extraneous phrases can potentially introduce incoherence if not removed. We collected 500 sentences and the corresponding reduced sentences written by humans. This indicates that a good sentence reduction system can improve the conciseness of generated summaries significantly. We implemented an automatic sentence reduction system. Input to the reduction system includes extracted sentences, as well as the original document. Output of reduction are reduced forms of the extracted sentences, which can either be used to produce summaries directly, or be merged with other sentences. The reduction system uses multiple sources of knowledge to make reduction decisions: it uses the syntactic knowledge from a large-scale lexicon we have constructed to try to guarantee the syntactic correctness of the reduced sentence; it uses the context in which the sentence appears to determine the phrases that are of local focus so that they will not be deleted during reduction; and it uses statistics computed from a corpus of examples produced by humans to decide how likely a certain phrase is removed by humans.

c. Sentence Compression

The sentence compression problem, which is very similar to our sentence reduction problem. They devised both noisy-channel and decision-tree approaches to the problem. The noisy-channel framework has been used in many applications, including speech recognition, machine translation, and information retrieval. Their system first parses the original sentence into a large parse tree, and then it hypothesizes and ranks various small trees that represent the compressed sentences using the stochastic model. The decision-tree method compresses sentences by learning the decisions for “rewriting” input parse trees into smaller trees, which correspond to the compressed versions of the original sentence.

The main difference between our sentence reduction program and the above compression algorithms include the following: (1) their compression algorithms do not consider the context of the sentences. They treat each sentence as isolated and independent. In contrast, in our reduction system, the context in which a sentence appears plays a very important role in determining how the sentence will be reduced; (2) the compression algorithms can produce ungrammatical sentences, even when an input sentence is grammatical and the parser analyzes the sentence correctly. Both the noisy-channel model and the decision-tree model are stochastic model only. Our reduction program aims to guarantee the grammaticality of the reduced sentences, by relying on syntactic knowledge from a lexicon. It is unlikely to produce an ungrammatical sentence, if the input sentence is grammatical and the parser analyzes the sentence correctly.

d. Sentence Combination

Sentence combination merges the reduced sentences from sentence reduction with other sentences or phrases. For example, the combination operation can be performed on two sentences both of which have been compressed by sentence reduction, or it can be performed on a reduced sentence and a phrase selected from the original text. Combination improves the coherence of generated summaries. A main source of incoherence in extraction-based summaries is dangling pronouns and noun phrases present in extracted sentences. One type of combination operation that is performed by our combination system is to replace the dangling pronouns and phrases with the names of the entities they refer to in the original text. Thus, a main source of incoherence can be eliminated. Moreover, by grouping closely related sentences together and merging them as a single sentence, combination helps readers to understand the relations between sentences. Sentence combination is a technique widely used by expert summarizers.

We implemented an automatic sentence combination module. Input to the module includes the sentences that have already been reduced by the reduction module, as well as the original document. The output of the combination module is merged sentences, which are used to produce summaries.

III. IMPLEMENTATION

Implementation is the stage of the project when theoretical design turned out into working system; thus it can be considered to be the most critical stage in achieving a successful new system. The implementation stage involves careful planning, investigation of existing system and its constraints on implementation, designing of methods to achieve changeover and evaluation of changeover method.

a. Syntactic Parsing.

We first parse the input sentence using the ESG parser and produce the sentence parse tree. The operations in all other steps are performed based on this parse tree. Each following step annotates each node in the parse tree with additional information, such as syntactic or context importance, which is used later to determine which phrases they are represented as subtrees in a parse tree) can be considered extraneous and thus removed.
In this step, we determine which components of a sentence must not be deleted to keep the sentence grammatical. To do this, we traverse the parse tree produced in the first step in top-down order and mark, for each node in the parse tree, which of its children are grammatically obligatory. We use two sources of knowledge for this purpose. One source includes simple, linguistic-based rules that use the grammatical role structure produced by the ESG parser. For instance, for a sentence, the main verb, the subject, and the object(s) are essential if they exist, but a prepositional phrase is not; for a noun phrase, the head noun is essential, but an adjective modifier of the head noun is not. The other source we rely on is the large-scale lexicon we described earlier.

The final reduction decisions are based on the results from all the earlier steps. To decide which phrases to remove, the system traverses the sentence parse tree, which has been annotated with different types of information from earlier steps, in top-down order and decides which subtrees should be removed, reduced or unchanged. A subtree (i.e., a phrase) is removed only if it is not grammatically obligatory, not the focus of the local context (indicated by a low importance score for each word in the extracted sentence, based on the number of links it has with other words and the types of links.

The final reduction decisions are based on the results from all the earlier steps. To decide which phrases to remove, the system traverses the sentence parse tree, which has been annotated with different types of information from earlier steps, in top-down order and decides which subtrees should be removed, reduced or unchanged. A subtree (i.e., a phrase) is removed only if it is not grammatically obligatory, not the focus of the local context (indicated by a low importance score for each word in the extracted sentence, based on the number of links it has with other words and the types of links.)
importance score), and has a reasonable probability. The program can reduce the sentences into different lengths by adjusting the thresholds it uses for importance scores and phrase removal probabilities.

**Figure 5**: final parsing

**IV. CONCLUSION**

Human-written summary sentence and proposed a Hidden Markov Model solution to the problem. The decomposition program can automatically determine whether a summary sentence is constructed by reusing text from the original document; it can accurately recognize phrases in a sentence despite the wide variety of their granularities; it can also pinpoint the exact origin in the document for a phrase. The algorithm is fast and straightforward. It does not need other tools such as a tagger or parser as preprocessor. It does not have complex processing steps. The evaluations show that the program performs very well for the decomposition task. The results from decomposition are used to build training and testing corpora for sentence reduction and sentence combination. Sentence reduction system that removes extraneous phrases from sentences that are extracted from an article in text summarization. The deleted phrases can be single words, noun phrases, verb phrases, preposition phrases, gerunds, to infinitives, or clauses, and multiple phrases can be removed from a single sentence. The focus of this work is on determining, for a sentence in a particular context, which phrases in the sentence are less important and thus can be removed. Sentence combination program that merges sentences reduced by sentence reduction with other reduced sentences or phrases. We also explored using rule induction machine learning technique to automatically acquire combination rules. The lexicon has been used in many applications, including summarization, traditional natural language generation, word sense disambiguation, and machine translation.

**Figure 6**: final syntax tree

**V. REFERENCES**

3) In Proceedings of the Fifth Conference on Applied Natural Language Processing, Washington D.C.