SIDE: The Summarization Integrated Development Environment

Moonyoung Kang, Sourish Chaudhuri, Mahesh Joshi, Carolyn P. Rosé

Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213 USA
moonyoun,schaudhu,maheshj,cprose@cs.cmu.edu

Abstract

In this type-II demo, we introduce SIDE[1] (the Summarization Integrated Development Environment), an infrastructure that facilitates construction of summaries tailored to the needs of the user. It aims to address the issue that there is no such thing as the perfect summary for all purposes. Rather, the quality of a summary is subjective, task dependent, and possibly specific to a user. The SIDE framework allows users flexibility in determining what they find more useful in a summary, both in terms of structure and content. As an educational tool, it has been successfully user tested by a class of 21 students in a graduate course on Summarization and Personal Information Management.

1 Introduction

A wide range of summarization systems have been developed in the past 40 years, beginning with early work in the Library sciences field. To this day, a great deal of research in summarization focuses on alternative methods for selecting subsets of text segments based on a variety of forms of rhetorical analysis and relevance rankings. Nevertheless, while there is much in common between approaches used for summarization in a variety of contexts, each new summarization project tends to include a new system development effort, because a general purpose, extensible framework for summarization has not been made available. As an example, Teufel and Moens' (2002) argue that the summarization strategy for scientific articles must be different from news articles because the former focus on novelty of information, are much longer and very different in structure.

A large proportion of summarization systems do not allow users to intervene in the summarization process so that the form of the summary could be tailored to the individual user's needs (Mieskes, M., Müller, C., & Strube, M., 2007). From the same document, many summaries can potentially be generated, and the most preferable one for one user will not, in general, be the same as what is preferred by a different user. The fact that users with similar backgrounds can have vastly differing information needs is highlighted by Paice and Jones' (1993) study where an informal sentence selection experiment had to be abandoned because the participants, who were agriculture experts, were too influenced by their research interests to agree with each other. However, summarization systems tend to appear as black boxes from the user's perspective and the users cannot specify what they would want in the summary.

SIDE is motivated by the two scenarios mentioned above - the absence of a common tool for generating summaries from different contexts, as well as the fact that different users might have different information needs from the same document. Bellotti (2005) discusses the problem of information overload in communication media such as e-mail and online discussion boards. The rapid growth of weblogs, wikis and dedicated information sources makes the problem of information overload more acute. It also means that summarization systems have the responsibility of taking into account the kind of information that its user would be interested in.

With SIDE, we attempt to give the user a greater say in deciding what kind of information and how much of it the user wants as part of his summary.

[1] The working system can be downloaded from http://www.cs.cmu.edu/~cprose/SIDE.html, and a video of an example of SIDE use can be found at http://ankara.lti.cs.cmu.edu/side/video.swf.

This project is supported by ONR Cognitive and Neural Sciences Division, Grant number N000140510043

Proceedings of the ACL-08: HLT Demo Session (Companion Volume), pages 24–27,
Columbus, June 2008.

In the following sections, we elaborate on the features of SIDE and its technical details.

2 Functionality

The design of SIDE is aimed at allowing the user as much involvement at every stage of the summary generation process as the user wishes. SIDE allows the user to select a set of documents to train the system upon, and to decide what aspects of input documents should be detected and used for making choices, particularly at the stage of selecting a subset of segments to preserve from the source documents. The other key feature of the development environment is that it allows developers to plug in custom modules using the Plugin Manager in the GUI. In this way, advanced users can extend the capabilities of SIDE for meeting their specific needs while still taking advantage of the existing, general purpose aspects of SIDE.

The subsequent sub-sections discuss individual parts of system behavior in greater detail at a conceptual level. Screen shots and more step by step discussion of how to use the GUI are given with the case study that outlines the demo script.

2.1 Filters

To train the system and create a model, the user has to define a filter. Defining a filter has 4 steps – creating annotated files with user-defined annotations, choosing feature sets to train (unigrams, bigrams etc), choosing evaluation metrics (Word Token Counter, TF-IDF) and choosing a classifier to train the system.

Annotating Files: The GUI allows the user to create a set of unstructured documents. The user can create folders and import sets of documents or individual documents. The GUI allows the user to view the documents in their original form; alternatively, the user can add it to the filter and segment it by sentence, paragraph, or by own definition. The user can define a set of annotations for each filter, and use those to annotate segments of the file. The system has sentence and paragraph segmenters built into it. The user can also define a segmenter and plug it in.

Feature Sets: The feature set panel allows the user to decide which features the user wants to use in training the model. It is built on top of TagHelper Tools (Donmez et al., 2005) and uses it to extract the features chosen by the user. The system has options for using unigrams, bigrams, Part-Of-Speech bigrams and punctuation built into it, and the user can specify whether they wish to apply stemming and/or stop word removal. Like the segmenters, if the user wants to use a specific feature to train, the user can plug in the feature extractor for the same through the GUI.

Evaluation Metrics: The evaluation metric decides how to order the sentences that are chosen to be part of the summary. In keeping with the plug-in architecture of the system, the user can define own metric and plug it into the system using the Plugin Manager.

Classifier: The user can decide which classifier to train the model with. This functionality is built on top of TagHelper Tools, which uses the Weka toolkit (Witten & Frank, 2005) to give users a set of classifiers to choose from. Once the system has been trained, the user can see the training results in a panel which provides a performance summary - including the kappa scores computed through 10-fold cross validation and the confusion matrix, the sets of features extracted from the text, and the settings that were used for training the model.

The user can choose the model for classifying segments in the target document. The user also can plug-in a machine learning algorithm to the system if necessary.

2.2 Summaries

Summaries are defined by Recipes that specify what types of segments should be included in the resulting summary, and how a subset of the ones that meet those requirements should be selected and then arranged. Earlier we discussed how filters are defined. One or more filters can be applied to a text so that each segment has one or more labels. These labels can then be used to index into a text. For example, a Recipe might specify using a logical expression such that only a subset of segments whose labels meet some specified set of constraints should be selected. The selected subset is then optionally ranked using a specified Evaluation metric. Finally, from this ranked list, some number or some percentage of segments will then finally be selected to be included in the resulting summary. The segments are then optionally re-ordered to the original document order before including them in the summary, which is then displayed to the user.

3 Case Study

The following subsections describe an example where the user starts with some unstructured documents and uses the system to generate a specification for a summary, which can then be applied to other similar documents.

We illustrate a script outline of our demo presentation. The demo shows how simple it is to move through the steps of configuring SIDE for a type of summary that a user would like to be able to generate. In order to demonstrate this, we will lead the user through an annotation task where we assign dialogue acts to turns in some tutoring dialogues. From this annotated data, we can generate summaries that pull out key actions of particular types. For example, perhaps we would like to look at all the instructions that the tutor has given to a student or all the questions the student has asked the tutor. The summarizing process consists of annotating training documents to define filters, deciding which features to use along with what machine learning algorithm to train the filters, training the actual filters, defining a summary in terms of the structured annotation that is accomplished by the defined filters, and finally, summarizing target files using the resulting configuration. The purpose of SIDE is to provide both an easy GUI interface for people who are not familiar with programming, and extensible, plug-and-play code for those who want to program and change SIDE into a more sophisticated and specialized type of summarizer. The demo will provide options for both novice users primarily interested in working with SIDE through its GUI interface and for more experienced users who would like to work with the code.

3.1 Using the GUI

The summarization process begins with loading unstructured training and testing documents. Next, filters are defined by adding training documents, segmenting each by choosing an automatic segmenter, and assigning annotations to the segments.

After a document is segmented, the segments are annotated with labels that classify segments using a user-defined coding scheme (Figure 1). Unannotated segments are later ignored during the training phase. Next, a set of feature types, such as unigrams, bigrams, part of speech bigrams, etc., are selected, which together will be used to build the feature space that will be input to a selected machine learning algorithms, or ensemble of algorithms. In this example, 'Punctuation' Feature Class Extractor, which can distinguish interrogative sentence, is selected and for 'Evaluation Metrics', 'Word Token Counter' is selected. Now, we train this model with an appropriate machine learning algorithm. In this example, J48 which is

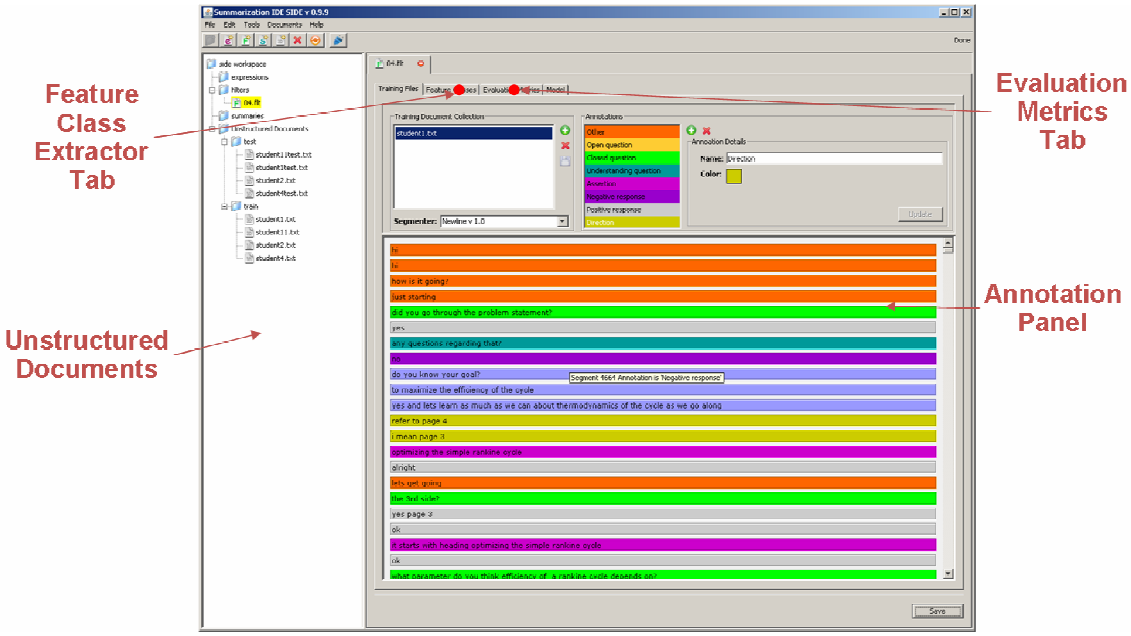

Figure 1: The interface where segments are annotated.

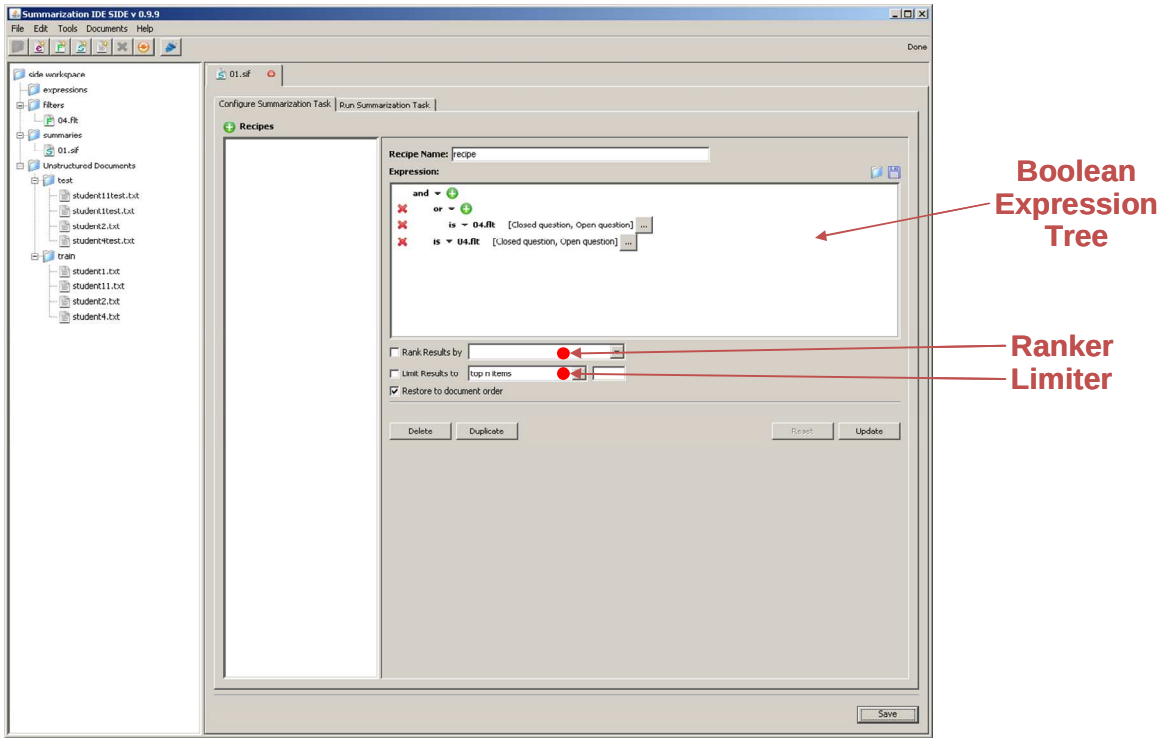

Figure 2: The interface for defining how to build a summary from the annotated data.

one of Weka's (Witten & Frank, 2005) decision tree learners is chosen as the learning algorithm. Users can explore different ensembles of machine learning algorithms, compare performance over the training data using cross-validation, and select the best performing one to use for summarization.

Once one or more filters have been defined, we must define how summaries are built from the structured representation that is built by the filters. Figure 2 shows the main interface for doing this. Recipes consist of four parts, namely 'Selecting', 'Ranking', 'Limiting', 'Sequencing'. Selection is done using a boolean expression tree consisting of 'and', 'or', and 'is' nodes. By doing selection, only those segments with proper annotations will be selected for inclusion in the resulting summary. Ranking is done by the Evaluation Metric selected when defining the Recipe. The size of a summary can be limited by limiting the number of segments you want in your summary. Finally, the summary can be reordered as you wish and displayed.

4 Current Directions

Currently, most of the functionality in SIDE focuses on the content selection problem. We acknowledge that to move beyond extractive forms of summarization, additional functionality at the summary generation stage is necessary. Our current work focuses on addressing these issues.

References

Bellotti, V., Ducheneaut, N., Howard, M., Smith, I., & Grinter, R. (2005). *Quality versus Quantity: E-Mail Centric Task Management and Its Relation with Overload*, Human-Computer Interaction, Volume 20,

Donmez, P., Rosé, C. P., Stegmann, K., Weinberger, A., and Fischer, F. (2005). *Supporting CSCL with Automatic Corpus Analysis Technology* , Proceedings of Computer Supported Collaborative Learning.

Mieskes, M., Müller, C., & Strube, M. (2007) *Improving extractive dialogue summarization by utilizing human feedback,* Proceedings of the 25th IASTED International Multi-Conference: artificial intelligence and applications, p.627-632

Paice, Chris D. & Jones, Paul A. (1993) *The identification of important concepts in highly structured technical papers. In* Proceedings of the 16th ACM-SIGIR Conference, pages 69–78

Teufel, S. & Moens, M. (2002). *Summarizing Scientific Articles: Experiments with Relevance and Rhetorical Status*, Computational Linguistics, Vol 28, No. 1.

Witten, Ian H.; Frank, Eibe (2005). *Data Mining: Practical machine learning tools and techniques, 2nd Edition*. Morgan Kaufmann, San Francisco.