SIDE: The Summarization IDE

Elijah Mayfield
Carolyn Penstein Rosé

Fall 2010

SIDE: The Summarization IDE

User Manual

Co-authors can be contacted at the following addresses:

Elijah Mayfield: elijah@cmu.edu

Carolyn Rosé: cprose@cs.cmu.edu

Work related to this project was funded through the Pittsburgh Science of Learning Center, the Office of Naval Research Cognitive and Neural Sciences Division, the National Science Foundation, Carnegie Mellon University, and others. Special thanks to Moonyoung Kang, Sourish Chaudhuri, Yi-Chia Wang, Mahesh Joshi, and Eric Rosé for collaboration and contributions to past versions of SIDE.

Table of Contents

1 SIDE: The Summarization Integrated Development Environment

SIDE, the Summarization IDE, was developed as an infrastructure for facilitating researchers in the task of getting a quick sense of text data. The use of machine learning models can facilitate study of a large set of data by highlighting key features and differentiating the significant factors from noise. However, this work of understanding the data being studied can not and should not be a fully automatic process. It is more desirable to have a human researcher in the loop, receiving filtered information about the data they are working with, and using human judgment to make the final decision about how to utilize this information.

The SIDE framework offers flexibility in the specification of which features to take note of, as well as how to search for these features in data, how to analyze an automated model for errors, and how to deliver this information to a user in a structured way. The combination of all of these tasks in a single suite of applications makes SIDE a valuable research tool as well as a utility for supporting institutional practice.

As we have stated earlier, SIDE is an application that makes use of machine learning. This functionality is provided by the Weka toolkit. It is important to understand what machine learning algorithms do. These algorithms are designed to induce rules based on patterns found in structured data representations. A researcher has two types of options in customizing the behavior of a machine learning algorithm. One is to manipulate the structured representation of the text being studied, and the other is to manipulate the selection of the machine learning algorithm. These two choices are not entirely independent of one another. An insightful machine learning practitioner will think about how the representation of their data will interact with the properties of the algorithm they select. Interested readers are encouraged to read Witten & Frank's (2005) book, which provides a comprehensive introduction to the practical side of the field of machine learning.

Example Applications

As a scenario, consider a course where students are heavily involved in on-line discussions as part of their participation in the course. Instructors likely do not have time to keep up with all of the correspondence contributed on the course discussion board. One such example is an on-line environment for Civics education where students participate in debates over time about bills that they propose in what amounts to a "virtual legislature." A time series displays student posting behavior over the course of an academic year in terms of both the number of posts and the level of argumentation quality automatically detected in those messages. From this visualization, one observation is that the frequency of student posts increases over the course of the semester. It is also clear that the proportion of high quality argumentation, indicated in green, does not consistently increase over the course of the semester. Weeks where the proportion of messages with high quality argumentation is highest seem to be weeks where there is a larger than average frequency of posting, potentially indicative of weeks where there is more intensive debate. Instructors using a visual-

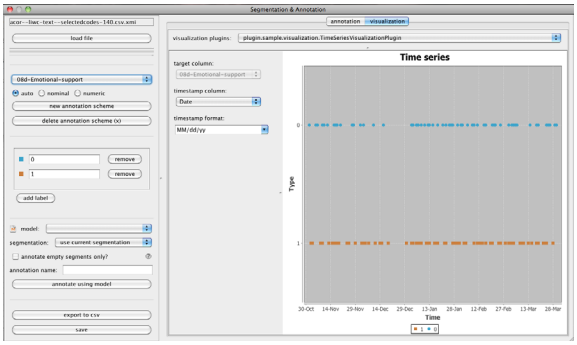

Figure 0: A simple time series visualization in SIDE.

ization like this would be able to determine which weeks students were not particularly engaged in the discussion. It would also help the instructor to identify weeks where the argumentation quality is lower than desired in order to offer suggestions or other support to elevate the level of intensity in the discussion in order to provide students with better opportunities to hone their argumentation skills.

Another similar scenario where reports based on analyses of behavior in a discussion forum can be useful is in project based courses where students do a lot of their group work outside of the direct supervision of the instructor. Insights about the well-being of student groups can be gleaned from some byproducts of group work, such as the messages left in a groupware environment used to coordinate the group work. Prior work has shown that machine learning models can make predictions with reasonable accuracy of how the instructor would rate the extent to which students have contributed productively to their group that week (with a correlation of R=0.63 in comparison with instructor assigned productivity grades). Such an interface could display trends in student participation over time to aid instructors in identifying students who may need more attention and prodding to intensify their contributions to their respective groups.

Visualizations are not the only type of summary that may be useful to instructors. With respect to the example in Figure 1, rather than knowing what proportion of messages exhibited high levels of argumentation quality, the instructor may want to see particularly good examples of argumentation quality to draw attention to as an example for struggling students. Alternatively, an instructor may want to glean a list of questions posed by students during a collaborative learning discussion in order to gauge which topics are confusing for students. Another possibility would be to monitor discussions related to negotiations over issues, such as design discussions where trade-offs such as power output versus environmental friendliness are being made with respect to the design of a power plant. In that case, it might be interesting to classify conversational contributions as indicating a bias towards environmental friendliness, or alternatively high power output, so that it is possible to display how preferences ebb and flow in the course of the discussion.

2 Installation and Setup

Checking your Java VM

In order to use SIDE, your computer must have a Java Virtual Machine installed with support for at least Java 6. As Java is platform independent, this means that almost any system should be capable of running SIDE; however, you must first ensure that you have the appropriate JVM installed. Below you will find instructions for checking your JVM on Windows XP, Mac OS X v10.5, and Fedora Core 11 Linux. Other operating systems should follow a similar general process.

Windows XP

- Click 'Start' then 'Run'.
- In the Run dialog, type 'cmd' and click OK.
- Type 'java -version' and press Enter. If an appropriate version of Java is installed on your computer, you will receive a response which includes, somewhere in the text, "java version 1.6.0" If your computer gives a similar response to this, you may proceed to installing SIDE. Otherwise, skip to the next section, "Installing the Java 6 VM"

Windows 7

- Open the start menu, then search for 'cmd'.
- Click the 'cmd' icon.
- Type 'java -version' and press Enter. If an appropriate version of Java is installed on your computer, you will receive a response which includes, somewhere in the text, "java version 1.6.0" If your computer gives a similar response to this, you may proceed to installing SIDE. Otherwise, skip to the next section, "Installing the Java 6 VM"

Mac OS X v10.6

- Open Finder.
- Click 'Applications' then 'Utilities'.
- Double-click 'Terminal'.
- Type 'java -version' and press Enter. If an appropriate version of Java is installed on your computer, you will receive a response which includes, somewhere in the text, "java version 1.6.0" If your computer gives a similar response to this, you may proceed to installing SIDE. Otherwise, skip to the next section, "Installing the Java 6 VM."

Fedora Core 11 Linux

- Click 'Applications' then 'Administration'.
- Click 'Terminal'.
- Type 'java -version' and press Enter. If an appropriate version of Java is installed on your computer, you will receive a response which includes, somewhere in the text, "java version 1.6.0" If your computer gives a similar response to this, you may proceed to installing SIDE. Otherwise, skip to the next section, "Installing the Java 6 VM."

Installing the Java 6 VM

If you are using a computer running Mac OS X, then you can install the Java 6 VM through the 'Software Update' utility. Open this program by clicking on the Apple icon in the top left corner and running selecting the 'Software Update' option. Install 'jre 6' with the highest update version available for your computer.

If you are using a computer running Windows, Linux, or any other operating system, you will need to download the appropriate file directly from Sun's official website:

http://java.sun.com/javase/downloads

Once you select the appropriate file here, you should open it and follow the instructions it gives.

Installing and running SIDE

Now that Java 6 is installed on your computer, you can start using SIDE. All the files that you will need for basic use are available in a single package located at the following website:

http://www.cs.cmu.edu/~cprose/SIDE.html

Save the file to your desktop for easy access. Now extract the package to a folder, using whatever archive manager you prefer. The resulting folder should be named 'SIDE'.

To run SIDE, open this folder. Depending on the operating system you are using, you will need to follow different steps to run SIDE. Once you have completed these steps, SIDE will be running and you can continue to the next chapter to begin to learn to use the software.

Windows XP

- Open the SIDE folder.
- Double-click the 'run' icon.
- SIDE will start after a short delay.

Windows 7

- Open the start menu and search for 'cmd'.
- Click the 'cmd' icon.
- Type "cd Desktop\SIDE" to navigate from your home folder to the location where SIDE was extracted. If you saved this folder somewhere else, you will have to navigate to it yourself.
- Type "./run.bat"
- SIDE will start after a short delay.

Mac OS X v10.6

- Open Finder.
- Click 'Applications' then 'Utilities'.
- Double-click 'Terminal'.
- Type "cd Desktop/SIDE" to navigate from your home folder to the location where SIDE was extracted. If you saved this folder somewhere else, you will have to navigate to it yourself.
- Type "./run.sh"
- SIDE will start after a short delay.

Fedora Core 11 Linux

- Click 'Applications' then 'Administration'.
- Click 'Terminal'.
- Type "cd Desktop/SIDE" to navigate from your home folder to the location where SIDE was extracted. If you saved this folder somewhere else, you will have to navigate to it yourself.
- Type "./run.sh"
- SIDE will start after a short delay.

3 Using SIDE: Text Annotation

Before stepping through how to use the SIDE GUI, it may be helpful to think at a high level about the process. First note that when using SIDE, we think in terms of producing summaries or reports about documents. Often the term "document" conjures up images of newspaper articles or reports, but even discussions or individual contributions to discussions can be documents. For our purposes, typical documents will be those that come up in instructional contexts such as answers to open response questions, logs from online discussions, email messages from students, posts to course discussion boards, and so forth. Therefore, a document can be a single sentence, a single word, or an entire essay, depending on the nature of your data.

SIDE was designed with the idea that documents, whether they are logs of chat discussions, sets of posts to a discussion board, or notes taken in a course, can be considered relatively unstructured. Nevertheless, when one thinks about their interpretation of a document, or how they would use the information found within a document, then a structure emerges. For example, an argument written in a paper often begins with a thesis statement, followed by supporting points, and finally a conclusion. A reader can identify with this structure even if there is nothing in the layout of the text that indicates that certain sentences within the argument have a different status from the others. Subtle cues in the language can be used to identify those distinct roles that sentences might play. Thus, machine learning models can be used to assign status tags to individual sentences and thus impose a structure on what initially looked at the surface to be unstructured. That structure can then be used in retrieving certain portions of the argument that might be of interest. For example, perhaps only the supporting arguments are of interest. It would not be possible to summarize the argument by pulling out these portions without first imposing the structure on the text that distinguishes between those three types of sentences.

Conceptually, then, the use of SIDE has two main parts. The first part is to construct filters that can impose structure on the texts that are to be summarized, and the second part is constructing specifications of summaries that refer to that structure and extract subsets of text or display visualizations of that structure.

To train the system and create a model, the user first has to define a filter. Filters are trained using machine learning technology. As we have stated previously, two customization options are available to analysts.

The first is the selection of the machine learning algorithm that will be used. Dozens of options are made available through the Weka toolkit, but some are more commonly used than others. The three options that are most recommended to analysts starting out with machine learning are Naïve Bayes, which is a probabilistic model; SMO, which is Weka's implementation of Support Vector Machines; and J48, which is one of Weka's implementation of a Decision Tree learner. SMO is considered state-of-the-art for text classification, so we expect that analysts will frequently find that to be the best choice.

The remaining customization options affect the design of the attribute space. The standard attribute space is set up with one attribute per unique feature - the value corresponds to the number of time that feature occurs in a text.

SIDE comes packaged by default to search for standard features from Natural Language Processing, integrating an older package called TagHelper Tools. The following types of features can be extracted automatically:

- *Unigrams and bigrams.* A unigram is a single word, and a bigram is a pair of words that appear next to one another. Unigrams are the most typical type of text feature. Bigrams may carry more information. They capture certain lexical distinctions such as the difference in the meaning of the word 'internal' between "internal attribution" and "internal combustion."
- *POS bigrams.* Part-of-speech bigrams are similar to the features discussed above, except that instead of words, they represent grammatical categories. They can be used as proxies for aspects of syntactic structure. Thus, they may be able to capture some stylistic information such as the distinction between "the answer, which is..." and "which is the answer." They may also be used as a primitive proxy for grammaticality - certain patterns of part-of-speech tags will be very rare in grammatical sentences, while others will be very frequent.
- *Treat Features as Binary*. This is a setting which applies to the features above. In some cases, it may be useful to count the number of occurrences of each word in a document; however, in most cases machine learning has been shown to be more effective if a bag-of-words model checks only for the presence of a word (1 or 0). This option toggles between those two settings.
- *Line Length*. This feature simply describes the number of words in a document. This may be useful in conversational data, where short lines should be treated differently from long, extended statements.
- *Contains Non-Stop Word*. This flag can identify whether a statement contains at least some contentful word. It is a boolean value set to either 0 if all words are filtered out from the stop words list (described below) or 1 if a word exists in the document that is not on that list.
- *Stop words*. This flag can weed out whether a contribution is contentful or not, which can be useful when processing chat data rather than newsgroup style data. For example, making a distinction between contributions like "ok sure" and "the attribution is internal and stable." Often the categories that are appropriate for non-contentful contributions are distinct from those that apply to contentful ones, so this can be a useful distinguishing characteristic.
- *Stemming.* Stemming is a technique for removing inflections from words in order to allow some forms of generalization across lexical items, for example the words stable, stability, and stabilization all have the same lexical root.

Once the reader has grasped these concepts, then it is not much of a stretch to consider that defining a filter has four steps: creating annotated files with user-defined annotations, choosing features to use for machine learning, choosing evaluation metrics, and choosing a classifier to train the system. Without a foundation of concepts in machine learning, these notions will likely sound very foreign.

Figure 2 shows the arrangement of buttons on the SIDE main menu, which follows the logical process of using SIDE. The first button allows the user to input files, either pre-annotated or plain text, and convert them to the internal format, referred to as UIMA. The Segmentation & Annotation interface enables users to define coding schemes that can be applied either by hand, using the Annotation interface, to place a structure on a document, or automatically, with a filter that can be defined using the Feature Table & Model Building interface. Structured files that result from annotation, either by hand or automatically, can be summarized. The Summary Panel allows the user

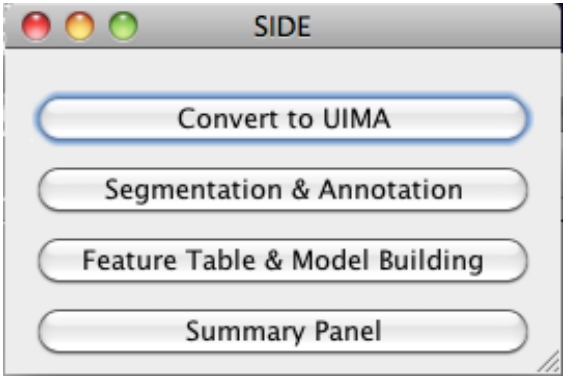

Figure 1: The launch panel and main menu of SIDE

to define how a summary can be built by specifying filters that will be used to apply structure to a document first, and then either specifying how visualizations can display patterns of annotations, or annotations can be used to find desired portions of a document.

For a running example, this tutorial will use example conversations from the AMI Meeting Corpus. A few of these files are available in our distribution of SIDE, including all of those used in this manual. The entirety of the corpus is available online for free. In these conversations, four participants discuss the design of a new TV remote control. The versions we distribute come annotated for dialogue act tagging and extractive summarization.

Lesson 1: Converting your data to SIDE's format for the first time

In order to provide maximum clarity for how to perform a particular task, these lessons will be giving very basic, step-by-step instructions on how to perform a task. While doing this, it is important not to lose sight of the big picture. You should not consider this process as merely a set of steps to be repeated, but an interactive, flexible, adapting understanding of the nature of your data. Before you can do this, however, you need to understand the basics of the user interface.

To convert a CSV file into UIMA format:

1. Open the CSV file you will be converting and examine it in a program such as Microsoft Excel. The files we will be using is located at SIDE/data/ami/
2. Make a note of which column stores the actual text data that you are importing. In this file, it is "text". You will need this information later.
3. Open SIDE, or if SIDE is already running, navigate to the launch panel.
4. Click the 'Convert to UIMA' button, the first button on this list.
5. In the 'document reader plugins' dropdown menu, make sure that the sample plugin 'CSVFileReader'

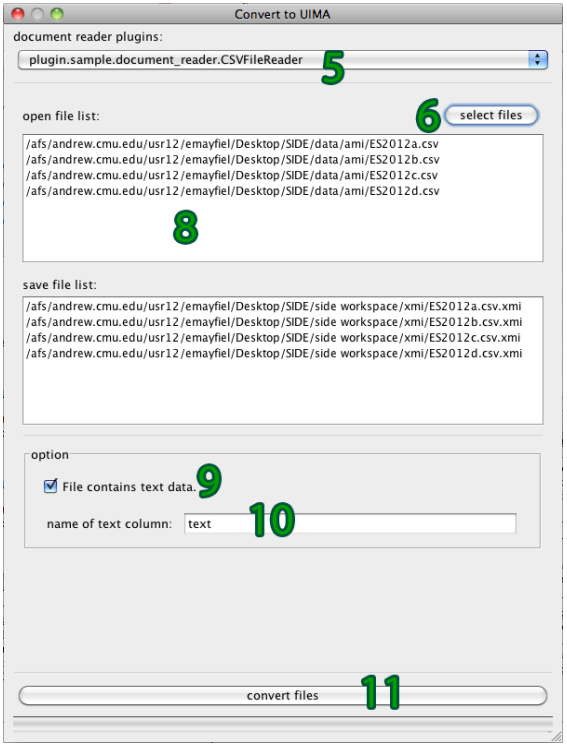

Figure 2: The UIMA conversion window in SIDE.

is selected (see figure 3).

6. Click the 'select files' button (see figure 3).
7. In the file browser that appears, select the files that you want to import.
8. Once a file is chosen, both the 'open file list' and 'save file list' text areas are filled automatically.
9. Ensure that the 'File contains text data' checkbox is checked. This should be the default setting. This informs SIDE where to look when building a feature table representation of your data.
10. Check the 'name of text column' field to make sure that it matches the header for your text column. In this case, the default ("text") is correct (see figure 3).
11. Click the 'convert files' button (see figure 3) at the bottom of the window to produce your UIMA document. This may take some time, especially for CSV files with many different columns (each annotation is processed separately).
12. Once the 'convert succeeded' dialog appears, close the Convert to UIMA window to return to the launch panel.

Why did my file conversion fail?

The most common source of problems in using SIDE is the initial conversion of the file. Don't worry if you've encountered problems! There are three likely sources of errors.

1. UIMA is designed to handle text data in ASCII formatting only. Because of this, any characters encoded in Unicode formats (such as Chinese characters) need to be removed from your document prior to conversion. In the future, we will build a way of handling Unicode charaters into SIDE, but in the current release, conversion of a Unicode file will fail. Many text editors contain an option to "zap gremlins" which will remove these intruding characters for you automatically.
2. SIDE requires that every entry in your data has a value for every column. If there are some cells in your table which are blank, SIDE's conversion will fail. A simple solution to this problem is to simply fill in those spaces with a label "Unknown" or "Blank" before using the file in SIDE.
3. Remember that in .csv files, commas separate columns. However, in text content, commas are often included in the transcription. Ensure that your text column is quoted if it contains commas, or the conversion will fail.

Can I use non-text data sets in SIDE?

Yes. In order to use a non-text dataset with SIDE to make use of its error analysis or defined feature functionality, simply uncheck the 'File contains text data' checkbox. In other windows, the text content of your files will show up as meaningless numbers. This can be safely ignored, and it will not be processed by the machine learning algorithms later on in the process of using SIDE.

Lesson 2: Analyzing annotated data within SIDE's interface

Our example files are meeting transcripts from the AMI Meeting Corpus. They are distributed with SIDE with three annotations intact, though many more exist in the full version of the corpus. This includes the "Speaker" annotation (who contributed each line)as well as two annotation schemes. These annotation schemes are "DialogAct" - which divides statements into one of four high-level categories ("Task," "Elicit," "Minor," and "Other") representing the contribution of the statement to the dialogue as a whole - and "Summary" - which labels individual lines based on whether they are important enough to be included in a conversation summary (with a simple "Yes"/"No" distinction). We can analyze this informa-

tion visually before performing any machine learning. In fact, doing so will be beneficial to our results, in all likelihood, as it will allow us to get a feel for the data that we will be working with, and understand what a meaningful feature space might look like intuitively.

To analyze segmented and annotated text:

1. Open SIDE, or if SIDE is already running, navigate to the launch panel.
2. Click the 'Segmentation & Annotation' button.
3. Click the 'load file' button in the top left corner of the window that appears.
4. Select the file that you want to examine - in our case, ES-2012a.csv.xmi and click 'ok'.
5. The data that we are examining will now appear in the main panel.
6. Open the drop-down box below the 'load file' button. Switch between the different annotation schemes and see that the main panel will adjust to match your selection.
7. Look at the list on the left hand side of the window. These labels are color-coded to match the segments in the main panel.
8. To change the color of an annotation, click the small color box on the left-hand list.
9. Once you have selected a suitable color from this box, click 'ok' and the annotations will change colors immediately.
10. Click 'save' in the bottom left corner to store the new colors you have selected in the UIMA file.
11. Look above the data panel, and see the two tabs labelled 'annotation' and 'visualization'. Click 'visualization' to switch to a different panel.

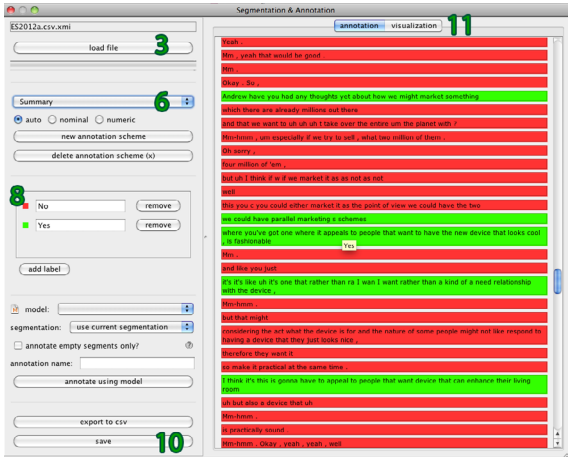

Figure 3: The Segmentation & Annotation window in SIDE

To analyze text using visualization:

12. Open the 'visualization plugins' dropdown and see that there are three available by default.

Note: Two of these visualization plugins rely on your data to have timestamps in one column. As our test data does not, we will only be able to experiment with the third, the pie chart visualization.

13. Click 'PieChartVisualization' to switch views.
14. Select one of your annotations in the dropdown menu on the far left hand side. This will show you the relative frequency of each label type.
15. Return to the 'annotation' tab. Close the Segmentation & Annotation window to return to the launch panel.

Lesson 3: Creating a new annotation scheme for your data

In this lesson, we'll add a new annotation scheme to represent a new type of information we're interested in predicting or using as evidence.

To add an annotation scheme for a UIMA file:

1. Open SIDE, or if SIDE is already running, navigate to the launch panel.
2. Click the 'Segmentation & Annotation' button.
3. Click the 'load file' button in the top left corner.
4. In the file browser, select the file to edit. We will again be using ES-2012a.csv.xmi.
5. Once the data loads, click the 'new annotation scheme' button. A popup window will appear.
6. Select the segmentation type that you want. It is likely that you will want "native" which preserves the segmentation from your CSV file.
7. Give your new annotation scheme a name. This should represent the feature you are labeling.
8. Click 'segment' to create this annotation scheme. You will notice that there are no label options, and all segments are now unlabeled.
9. Now we need to define labels. Click 'add label' twice to make new label types.
10. Change the text in these labels to match what you want to label.
11. Click the color boxes to the left of the labels to change the color appearing in the main panel.
12. Right click a segment in the Annotation panel and click 'set label.' This will open a popup window.
13. From the scroll menu, select the radio button next to the annotation matching this segment. Click 'ok' to apply the label to this segment.
14. You can annotate multiple segments at a time by holding down shift (for contiguous sections) or control (for dispersed segments) and right clicking once to label all selected segments.
15. You can right click and select 'select unannotated segments' to select all unlabeled segments at once. 'Select all segments' behaves in a similar way.
16. You can right click the incorrectly annotated segment and select 'clear annotation' to remove annotation from that segment.
17. If you choose to remove an annotation type from a scheme altogether, you can click 'Remove' on the list of labels to remove annotations from any segment that were labeled that way.
18. To save the annotations that you have made, click 'save' in the bottom left corner.
19. You may also want to re-export your annotations back to CSV; you can do this by clicking the 'export to CSV' button.
20. Close the 'Segmentation & Annotation' panel once you are finished annotating.

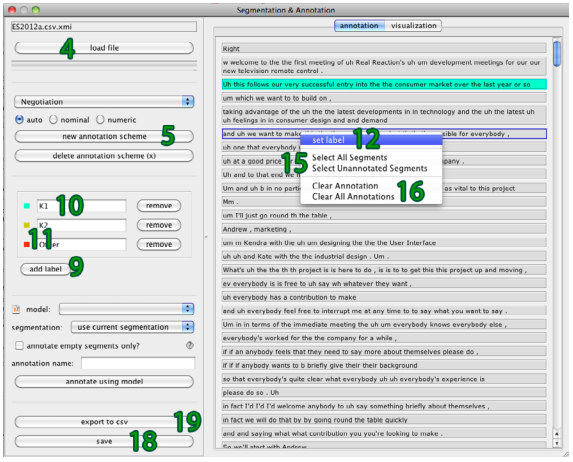

Figure 4: Creating an annotation scheme in SIDE.

4 Using SIDE: Machine Learning

Lesson 4: Building a feature table to represent your data set

To predict an annotation using machine learning, a document must first be converted into a form that is understandable by the algorithms that SIDE uses. This means that the document must be equivalent to a vector of features, each one of which has a single value associated with it.

What this means in a natural language processing scenario is usually that a model will be constructed in the "bag of words" fashion, where each feature in the vector corresponds to the presence of a word or sequence of words. This is the type of feature table that the default TagHelperExtractor we use in this example will create.

In order to improve performance from a baseline, the feature table representation is one of the first places that a research should consider. Later lessons will teach you the basics necessary for error analysis (lesson 6), feature construction (lesson 7), and including your own code in SIDE's plugin framework (Chapter 6). For now, we will use the simpler representation available from the TagHelperExtractor.

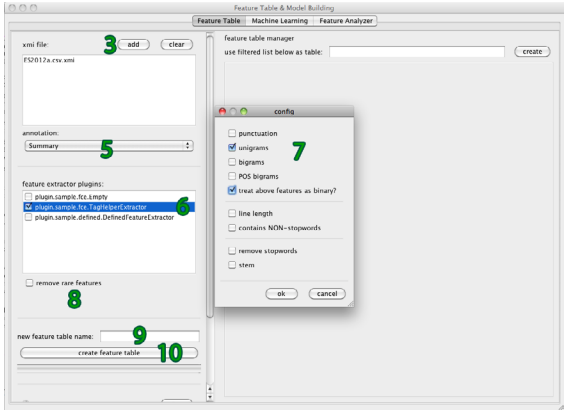

Figure 5: Building a feature table in SIDE.

To build a feature table for training a classifier:

1. Open SIDE, or if SIDE is already running, navigate to the launch panel.
2. Click the 'Feature Table & Model Building' button.
3. Click 'add' in the top left corner to open a file chooser popup window.
4. Select files you want to extract features from - you can select multiple files at once by holding control or shift. Click 'ok' once you have chosen all files from which you want to extract features. We will use ES-2012a.csv.xmi.
5. Select which annotation you want to learn to predict in the 'annotation' dropdown menu.
6. Choose which features you would like to extract using the 'feature extractor plugins' list. Check the 'TagHelperExtractor' box.
7. Right click the 'TagHelperExtractor' label to

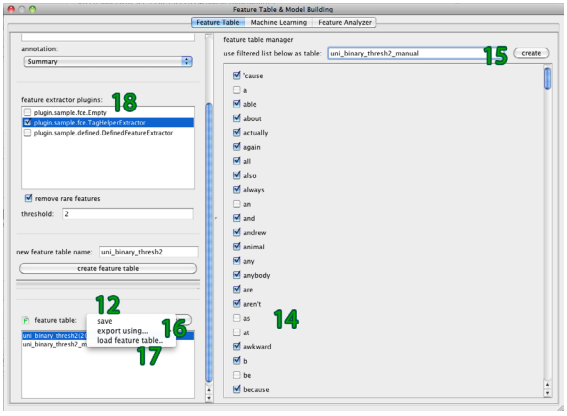

Figure 6: Exporting a feature table for use outside of SIDE.

open a configuration popup window. Here you are able to select which features to extract. We will be extracting a simple unigram model, so select 'unigram' and 'treat features as binary' and click 'ok.'

8. If you wish to filter out features which do not occur frequently, check the 'remove rare features' checkbox. Then, type a number into the textbox that appears. This will require that feature to occur in at least that many documents in order to be included in your feature table.

9. Choose a name for this feature set, and type it into the 'new feature table name' text area.

10. Click the 'create feature table' button and the features you chose will be extracted. Be patient - this may take a few minutes for a large dataset.

11. The 'feature table description' panel will now show a checklist containing the names of the features you extracted.

12. Right click a third time and select 'save' to open a file chooser dialog where you can save the results of this table for future use within SIDE.

13. At this time, leave the window open and proceed to Lesson 5.

To edit a feature table manually once it has been created:

14. If there are features that you believe are hurting performance of your classifier, you can uncheck them from the right-hand window.

15. Then, in the 'use filtered list below as table' text box, give this filtered list a new name and click 'create.' The feature table with the changes you have made will be available to you in the list on the bottom left corner.

Other options associated with feature tables in SIDE:

16. Right click the table name you just created in the 'feature table' list and choose the 'export using' option, then select HTML Feature Table or ARFF Feature Table options. These options will let you view the feature table in a web browser or in the Weka machine learning environment.

17. If you would like to load a table that was previously saved, such as the XML file you just created in step 15, you can right click the 'feature table' list and select 'load feature table.' This will open a dialog for loading tables into SIDE.

18. The 'Empty' feature extractor will allow you to create an empty feature table with no features extracted. This is useful for working with non-text datasets. Then, in the Machine Learning panel (see lesson 5), you can select the 'Use Metafeatures' checkbox to use the other columns in your dataset as features.

19. To remove the feature tables that you've already created, you can click the 'clear' button.

Lesson 5: Training an automatic classifier using machine learning

To train a classifier on your feature table:

1. Before beginning this lesson, complete Lesson 4, or make sure the 'Feature Table & Model Building' window is open with a feature table loaded in the 'feature table' list.
2. At the top of the window, switch to the 'Machine Learning' tab.
3. In the top left corner, click the 'Choose' button to select the machine learning algorithm you would like to use. This will open a tree-structured window.
4. The three algorithms you will most likely want to choose between will be NaiveBayes (in the 'bayes' folder), SMO (in the 'functions' folder), and J48 (in the 'trees' folder).
5. Open the 'feature table' dropdown menu and select the table (see Lesson 4) to use.
6. You can choose whether to perform cross-validation and select the number of folds in the 'cross-validation' field; it is set to do so by default.

Note: If you do not use cross-validation, you will not be able to perform error analysis (lesson 6).

7. You must choose a segmenter from the 'default segmenter' dropdown. However, your choice is not important unless you are performing summarization. For machine learning, this option is ignored.
8. If you wish to use the other annotations on your dataset as features for machine learning, select the 'Use metafeatures' checkbox. This is important if you are using a non-text dataset, for which these extra columns will be your only features.

Note: If you use metafeatures, then all future data that you wish to annotate using this model must have those same exact metafeatures available to the model.

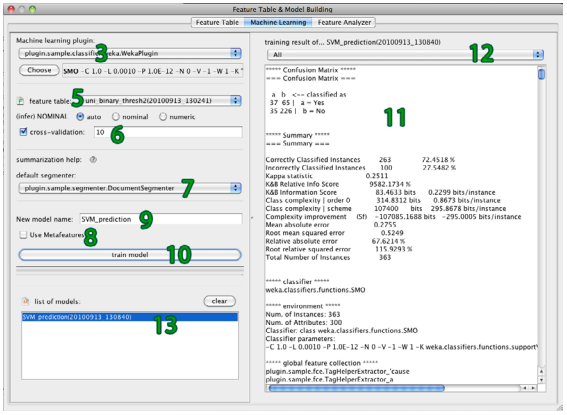

Figure 7: Training a classifier using machine learning in SIDE.

9. Give a name to the model you want to build in the 'New model name' field.
10. Click the 'train model' button to build a model.

Note: Machine learning can be slow! This is especially true of large data sets or feature tables, or complex algorithms.

11. Once your model is built, information about your model will appear on the right window.
12. To focus on specific information about your model, such as the weights themselves (for an SVM-style model) or the confusion matrix (for preliminary error analysis), click the dropdown menu on the top of the right window.

To save/load models for repeated use of a classifier:

13. Right-click the model name in the 'list of models' menu and choose 'save.'
14. To load a classifier that was previously saved, you can right-click in the 'list of models' and select 'load training result.'
15. At this time, leave the window open and proceed to Lesson 6.

Lesson 6: Performing error analysis on an annotation model

In an insightful process of applied machine learning, a practictioner will design an approach that takes into account what is known about the structure of the data that is being modeled. However, typically, that knowledge is incomplete, and thus, there is always a good chance that the decisions that are made along the way are suboptimal. When the approach is evaluated, it is possible to determine based on the proportion and types of errors whether the performance is acceptable for the application or not. And if it s not, then the practitioner should engage in an error analysis process to determine what went wrong and what could be done to better model the structure in the data.

Two common ways to approach an error analysis are top down, starting with the learned model, or bottom up, starting with the confusion matrix. In the first case, the model is examined to find the attributes that are treated as most important in the model. These are the attributes that have the biggest influence on the predictions made by the learned model, and thus these attributes provide a good starting point. In the second case, the bottom-up case, one first examines the confusion matrix to identify large off-diagonal cells, which represent common confusions. Consider the sets in a confusion matrix:

CLASSIFIED AS ->	YES	NO
ACTUALLY YES	(set a)	(set b)
ACTUALLY NO	(set c)	(set d)

The error analysis is then the process of determining how examples in set C could have been associated closely enough with examples in set A to be classified as such by your machine learning model. This can be done by identifying attributes that most strongly differentiate sets C and D (a horizontal comparison), and attributes that are most similar between sets A and C (a vertical comparison). The same process applies for the error in set B or any error cell.

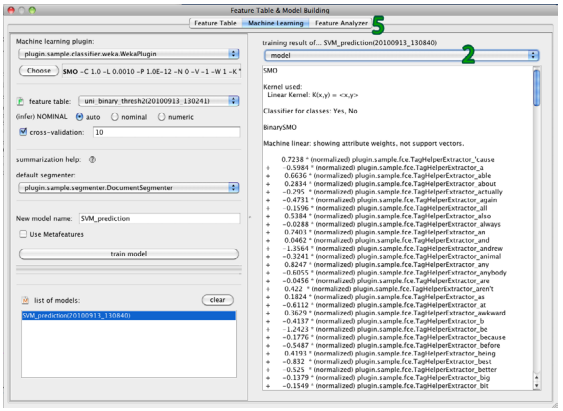

Figure 8: High level classifier information in SIDE

To study a machine learning model at a high level:

1. Before beginning this lesson, complete Lesson 5 or make sure the 'Feature Table & Model Building' window is open with a trained model loaded in the 'Machine Learning' tab.

2. The dropdown menu at the top of the training results page has many options for examination. Open it and click the 'model' option. This will give you the model that your classifier is actually using.

Note: Throughout the process described below, it is important to keep coming back to this model. If you make conclusions based on differences in the data, those conclusions must also be backed up by the actual decision making structure of your model. For instance, a decision tree must actually make use of a feature at some point reasonably high up in the tree in order for that feature to be important. A linear SVM model must be giving weight to a feature in order for that feature to affect the classification of a document. And so on.

3. Now click the 'Summary' option. This page gives a lot of information about our classifier based on cross-validation, including total accuracy, Kappa

accuracy, and other statistics.

4. Finally, click the 'Confusion Matrix' option. This gives us a brief summary of the results and sources of error. We will look at this matrix in much greater detail now.

To explore a confusion matrix in detail:

5. Switch to the 'Feature Analyzer' tab by clicking it at the top of the screen.

6. Select the model you want to analyze in the 'model' dropdown menu in the top left corner.

7. By default, all segments that were evaluated in cross-validation display in the bottom-right scrolling list, and the confusion matrix appears in the middle of the page.

Note: This process requires you to keep track of a lot of information about what categories different sets of data fall into and what these categories mean. As you work, the labels on this window will change to give you some reminders. For instance, the left-hand panel gives the predicted and actual labels of the segments in the cell you have highlighted, while the right-hand panel is labeled with the name of the correctly classified category you are comparing against. Click a cell that is not along the diagonal of correct cells in the confusion matrix at the top of the page. This will fill the bottom left scrolling list with the contents of that cell.

8. Open the 'comparison' dropdown menu and see that you have three options - full, horizontal, and vertical comparison. By default, 'Full Comparison' is selected and shows all segments in the right-hand list. Click 'Horizontal Comparison.'

9. Now that you have selected a comparison type, open the 'highlighted feature' dropdown menu. The contents are sorted by degree of difference between the segments that were incorrectly labeled and the segments that were correct. This means that elements at the top of the list are more likely to be predictive of why a machine learning algorithm is making mistakes.

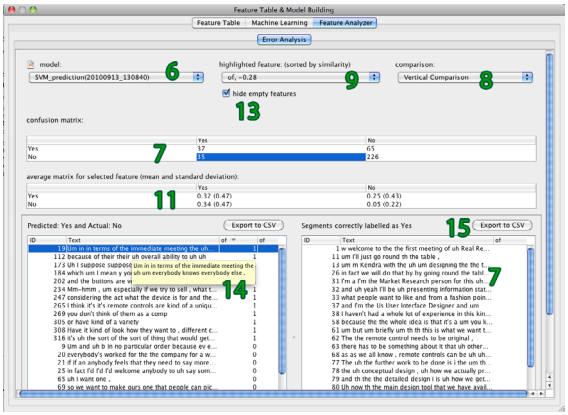

Figure 9: Error analysis interface in SIDE.

10. Select a feature from the 'highlighted feature' list. A second confusion matrix will appear below the menu.

11. Study the second confusion matrix. For each cell, this gives the average value of the highlighted feature among members of that cell, with standard deviation given in parentheses.

12. Now switch to 'Vertical comparison' and reopen the 'highlighted feature' dropdown menu. You will see that the features have now been automatically resorted to show similarity between the two cells. In this case, because the cells represent different actual labels, this will again show a possible reason why the classifier made a mistake.

13. You may have noticed that the 'hide empty features' box is checked by default. This option can be unchecked if you are interested in features which do not appear in the documents in the cells you are comparing.

14. To see the entirety of a document's text, hover over the 'Text' column and it will appear as mouseover text.

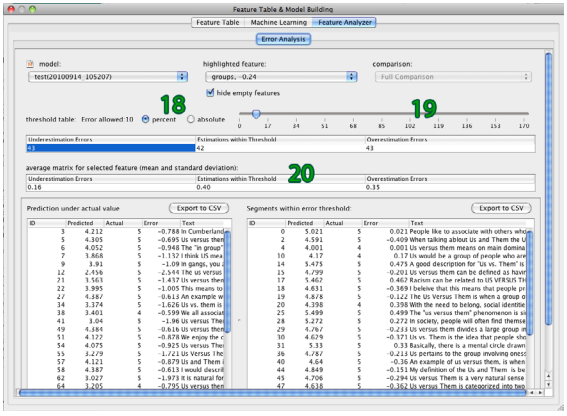

Figure 10: Numeric error analysis in SIDE.

15. If you are interested in experimenting with your data outside of SIDE, you can click the 'Export to CSV' buttons to export the tables in their respective windows to an external CSV format.

16. Continue to experiment with different cells and different comparisons to get a better understanding of the ways in which your model failed. Once you are done, you can close the 'Feature Table & Model Building' window.

To study a numeric prediction result:

SIDE does offer support for numeric prediction, such as linear regression tasks. The error analysis interface for these problems is different, because there is no concept of a "confusion matrix." Instead, we can consider two different types of errors - overestimation and underestimation, where the predicted value is higher or lower than the actual result, respectively. In fact, that is exactly what the numeric error analysis interface does.

17. When using the numeric interface, there is a slider visible to the user. This slider allows you to customize the margin of "correct" answers.

18. Select "percent" or "absolute" to determine whether you want the error tolerance to be measured as an absolute value (for instance, a prediction within 2.0 of your actual value is acceptable) or as a percentage (for instance, a tolerance of up to 10% error).

19. Moving the slider from side to side will change the accepted tolerance. Note that this can be slow to respond as it is recalculating categorization for all instances in real time.

20. The confusion matrices underneath the slider are now comprised of only three cells, rather than a matrix. Thus, there is no longer an option of either horizontal or vertical comparison. You can still select one category of error, but it will always be compared based on its differences from the correctly classified instances.

Lesson 7: Defining more complex features of your data by hand

Once you have identified different types of errors that have occurred in your data, it is likely that you will want to attempt to improve your performance based on these insights. There are a few different ways of doing this. The most straightforward is to change the algorithm that you are using, either moving to an entirely new algorithm or altering parameters of the current algorithm. However, there is a great deal that can be done in the feature table representation while keeping the same algorithm for learning. The feature table can be modified by removing items which are giving incorrect evidence, filtering down the features in a table, as discussed in lesson 4. You can also write entirely new plugins which offer additional functionality over what SIDE already gives you, as will be discussed in Chapter 6. However, the DefinedFeatureExtractor gives another option for users. This interface can be daunting at first, but it gives a great deal of flexibility in terms of what information you would like to include in a new feature.

The features that we will be constructing can be comprised of a variety of structures. The first ones that we will discuss are boolean features. These are boolean trees that evaluate to 1 as a feature if the statement that is described by the tree is true, and 0 if it is not true.

These features can contain important contextual information. For instance, consider a feature such as:

(XOR diminished ambitious)

This feature may make a distinction between subtleties in a movie review, for instance, in the following extract:

> *this is a promising premise, and mr. taylor's film could have gone any number of ambitious ways from this point [...however] this is not a film which has the wherewithal to kill off its leading star in the opening ten minutes. the entire sequence is, then, clearly an exercise for character exposition, with attempts at humour terribly diminished by utter predictability*

To define a boolean feature based on word n-grams:

1. Ensure that SIDE is running and that you are in the 'Feature Table & Model Building' window.
2. Click 'add' to choose the file or files to build features from.
3. Select which annotation you want to learn to predict in the 'annotation' dropdown menu.
4. Check the 'DefinedFeatureExtractor' option in the 'feature extractor plugins' menu. Then, right click on it to open the feature definition window.
5. The top segment of the window will be open to the 'Word N-Gram' tab by default. This gives you three columns of n-grams which appear in your document.
6. To change the length of n-grams in one of these lists, change the number in the 'NGrams of length:' text box and click 'Load' to refresh.

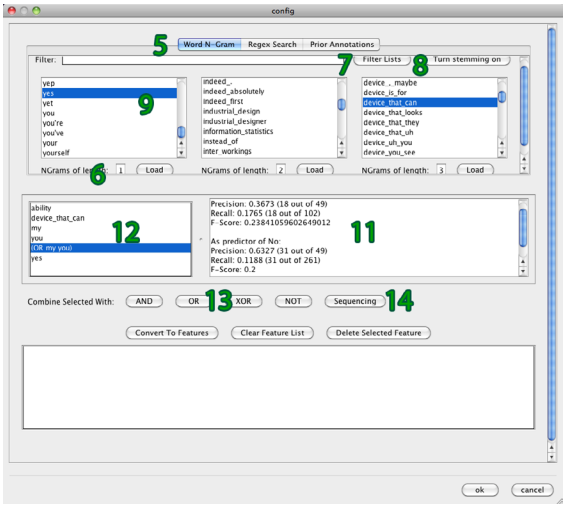

Figure 11: N-Gram Defined Features creation window.

7. To filter the lists of n-grams to a specific word or set of words, type the word you are searching for in the 'Filter' text box and click "Filter Lists" or press Enter.
8. To search for stemmed n-grams instead of surface n-grams, click the "Turn stemming on" button; to change back, click the same button.
9. To add one of these n-grams as a leaf node for a boolean feature, double click it in the list. It will be added to the list of features in the middle of the window.
10. Double-click a feature in the middle list. This will open up a popup window showing the instances in which this feature occurs in your data.
11. Once you have added a feature to the middle window, click it and the text area to its right will fill with information about the predictiveness of the feature. The information given is precision, recall, and f-score. These are useful indicators of the specificity and generalizability of a feature.
12. To combine features using a boolean operator, select them all in the middle window by click-

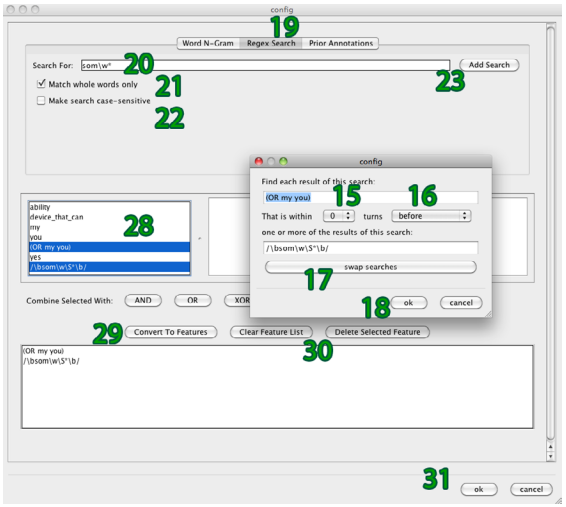

Figure 12: Regular expression and sequencing based feature creation.

ing them while holding down Shift or Control.

13. Once all the features that you want to use are highlighted, click one of the boolean operators (AND, OR, XOR, NOT) and a new feature will be created that is a combination of those features with that operator.

To define a sequencing-based feature:

If your data set is conversational, that is, each document is meant to follow the previous document temporally, then it may be interesting to note sequences of words occuring throughout the conversation, and the order in which they occur. You may want to look for patterns which will give a feature representation of these sequencing criteria. The Sequencing button next to the boolean buttons gives you that option.

14. To create a sequencing feature, highlight exactly two features in the middle window. Then click the 'Sequencing' button and a popup window appears.

15. To change the window in which you will look for a past feature, change the 'within X turns' dropdown menu's setting. Options available to you are from 0 (meaning that the words must occur in the same document) to 5.

16. To change the direction that a feature is looking, the second dropdown menu lets you choose whether to look for the additional feature before, after, or both before and after the current document. This dropdown menu will change that window.

17. To change the ordering of the two features (that is, to look for the second feature listed rather than the first, thereby looking contextually for the first feature rather than the second), click the 'swap searches' button.

18. Once you have defined the sequencing criterion you are interested in, click 'ok' to add this new feature candidate to the middle window.

To search within a document using regular expressions:

Some features that you may be interested in are features which are more complex than n-grams but which are still contained within a single document. For this purpose you are able to create regular expressions which will create a feature based on whether that regex matches the text of each document individually. The syntax of these regular expressions matches that of the Java Pattern class, and includes operators such as *, +, ?, and character classes such as \w, \s, etc. More information on how to use regular expressions is available at:

download.oracle.com/javase/tutorial/essential/regex/

19. To search for a pattern more complex than an n-gram within a document, switch to the 'Regex Search' tab at the top of the window.

20. Enter your regular expression into the 'Search for:' text box, following Java regex syntax.

21. To only find whole words that match your regex (rather than finding subsequences within words),

check the 'Match whole words only' checkbox.

22. If you want your search to be case-sensitive, click the 'Make search case-sensitive' checkbox.

23. To create a feature based on your regular expression, click the 'Add Search' button.

To use information from other annotations of your data:

It will occasionally be useful to search for specific information about your data based on other annotations of the data. Note that this will require all of your data from this point forward to be annotated with exactly the same layers if you use these features, just as using metafeatures in general requires, which is restrictive for fully automatic systems. Currently, these prior annotations are slow to evaluate in SIDE. We intend to improve efficiency in the future.

24. Switch to the 'Prior Annotations' tab at the top of the window.

25. Select the annotation that you are interested in from the left list, which will produce the set of labels from that annotation.

26. Select the label that you want to search for from the right list by double clicking on it.

27. If you want to know simply whether the label of the current segment is the same or different from the previous segment, you can choose those options from the right list as well.

Note: In order to use the prior annotation feature, the segmentation of those annotations must be the same as the annotation that you are trying to automate. Thus, you cannot have a document-level segmentation for the annotation you are predicting while using features based on a sentence-level segmentation.

To finalize the set of features to add to your feature space:

The set of features that you have in the middle window will be comprised of partial or incomplete

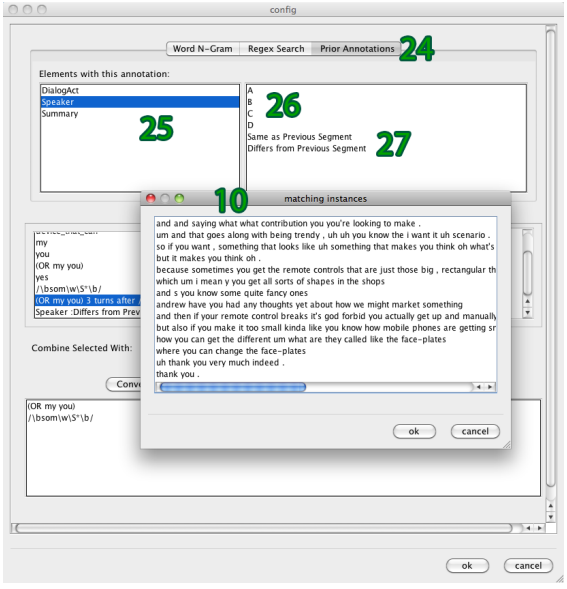

Figure 13: Prior annotation features, and matching instances window.

features; it is through constructing combinations of these features that you gain information. Obviously, you don't want to add all of the intermediate steps to your feature space. Thus, there is a separate step to create the final set of features to be added.

28. Highlight the feature or features that you want to add to your final feature space in the middle window by clicking on them with Shift and Control.

29. Click the 'Convert to Features' button. The highlighted features will be shifted to the bottom window.

30. To correct mistakes in the bottom list, press the 'Delete Selected Feature' button to delete the currently highlighted feature, or press the 'Clear Feature List' button to clear the list. The features in the middle window will remain.

31. When you have finalized your list of features to add to your feature space, click 'ok' at the bottom of the window.

Lesson 8: Automatically annotating your data with your model

To automatically annotate a UIMA file:

1. Before beginning this lesson, make sure you have loaded a model in the 'Feature Table & Model Building' window.
2. Click the 'Segmentation & Annotation' button.
3. Click the 'load file' button and select a file to annotate automatically.
4. Open a file which does not currently have the annotation that you are interested in.
5. If there are no current annotations of the text, the right side of the screen will be blank. If this is the case, you must first segment the text by clicking the 'new annotation scheme' button. This will open a popup; choose 'native' segmentation.
6. In the 'model' dropdown menu in the bottom left, select the model to use for annotation.
7. In the 'segmentation:' dropdown, ensure that 'use current segmentation' is checked.
8. Give the new scheme a meaningful name in the 'annotation name' text area.
9. Click the 'annotate using model' button. This may take awhile, particularly for large files.
10. Once this finishes, the new annotation scheme will be in view automatically. Note the new item in the interface, a partially filled bar on the right side of each segment. This shows the classifier's confidence in each prediction.
11. Click 'save' in the bottom left of the window if you wish to use this annotation in the future.

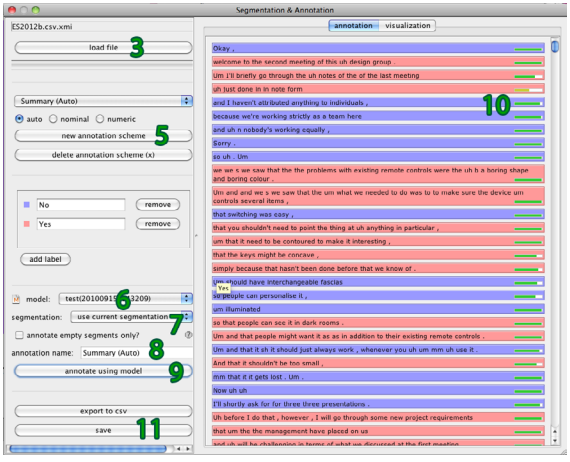

Figure 14: Automatic annotation of text using a trained classifier.

5 Using SIDE: Summarization

At this point, you have a filter that can be used to segment and annotate files. Once a file has gone through this process, that imposed structure can be used in the process of generating a summary so that you can retrieve just those parts of the document that are relevant for the summary that you want.

The first step is to define the selection criteria, which will be used to extract a subset of segments from the text. The simplest model of this criteria is annotated in the AMI meeting corpus data that SIDE comes with. It contains a layer with a Yes/No annotation, 'Summary' where the segments that should be included in a summary are marked as 'Yes.' There is ample room for adding additional layers, however, by automatically identifying segments based on topic or rhetorical structure, for instance.

The selection criteria is specified in terms of annotations that have been defined. If you refer to a filter in the selection criteria, that filter will be applied to the text. Then the conditions placed on the filter will be applied. What that means is that you can indicate that a subset of the annotations that can be applied to the filter indicate that the corresponding segments should be selected. Using a combination of boolean operators, you can create arbitrarily complex selection criteria involving multiple filters.

Lesson 9: Defining a recipe for summarization of your data

To write a recipe for summarization:

1. Before beginning this lesson, complete Lesson 5, or make sure SIDE is opened to the launch panel and you have loaded a model in the 'Feature Table & Model Building' window.
2. Click the 'Summary Panel' button.
3. In the 'Expression builder' menu, click the black triangle and select 'is' and then the model that you want to get a label from.
4. Click the '...' button in this recipe and select the label that you want to use as a filter for this summary.
5. If you would like to include more categories, you can click the green '+' icon to include bool-

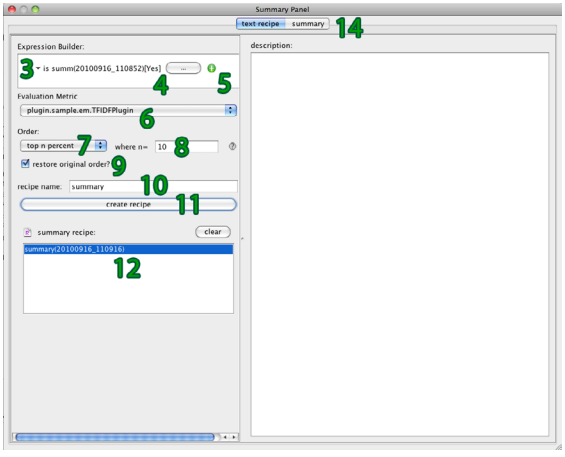

Figure 15: Defining a summary recipe in SIDE.

ean operators to additional filters.

6. Select the built-in TF/IDF plugin from the 'Evaluation Metric' dropdown menu for the segments that best fit the conditions you have specified in your recipe.
7. Select from the 'Order' dropdown the type of results you wish to receive.
8. In the 'where n=' text area enter the number of results you wish to receive, or the percent of items in the set if you wish to receive a variable number based on the size of the document.

Note: When entering percentages, input the number as a decimal; for instance, type "0.35" for 35%.

9. To choose whether to order the resulting summary in its original order or by highest rank, check or uncheck the 'restore original order?' checkbox.
10. Fill in the 'Recipe name' text area with a name.
11. Click the 'create recipe' button and the recipe object will be generated for your use.

To save/load models for repeated use of a classifier:

12. To save this recipe for future use, right click its name in the 'summary recipe' list and select 'save.'
13. Right-clicking in the 'summary recipe' list and selecting 'load recipe' will allow you to use an existing recipe.
14. At this time, click the 'summary' tab at the top of this window and proceed to Lesson 9.

Lesson 10: Generating a summarization of a text using a recipe

To generate a summary:

1. Before beginning this lesson, complete Lesson 8, or make sure you have loaded a summary recipe in the 'Summary Panel' window.
2. Click 'add' to choose which data to summarize.
3. In the 'segmentation' menu, choose the segmentation option that you want to use; if you want to use segments as defined in your original file, choose 'native' as your segmentation option.
4. Select in the 'summary recipe' menu which recipe you will use for summarization.
5. Name your summary in the 'summary name' field.
6. Click 'create summary object' to add a summary object to the 'summary' list in the bottom left corner of the panel.
7. Click 'summarize' to automatically annotate your document using the model in the recipe, and give a list of the segments that you chose in your recipe as the most important. These results will be shown in the main panel of this window.
8. Choose a visualization plugin and click the 'visualize' button to produce a graphical summary of your data.
9. The summary is also produced in plaintext in the terminal window where SIDE was opened.

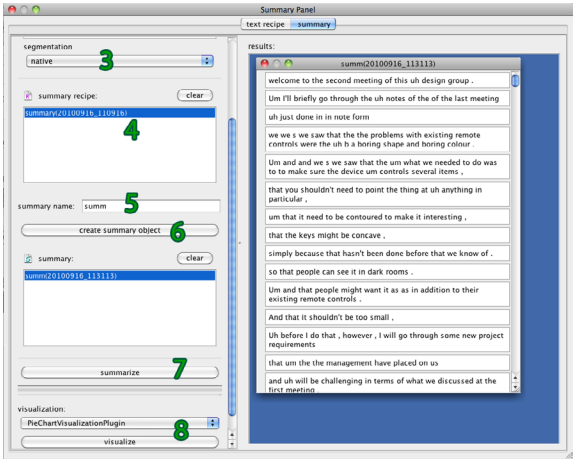

Figure 16: Generating a summary automatically in SIDE.

6 Extending SIDE: Writing Plugins

SIDE is designed to be a fully extendable framework for machine learning and summarization. Thus, for each of the major functionalitys of the program, there is a common interface that must be implemented, which has an expected output. Listed below are those plugin interfaces, along with brief descriptions of how they are used within the SIDE workflow:

- **MLAPlugin**: This interface builds a classification model, given a feature table representing a set of documents. A plugin implementing MLAPlugin must be able to build a model based on a feature table of training data, and apply that model to predict the labels of a given unlabeled dataset. SIDE comes with the WekaPlugin implementing this interface by default.
- **FEPlugin**: This interface takes as input a list of documents and converts those documents to a feature table. A plugin implementing FEPlugin must be able to take as input a list of strings and convert them to a map of feature names to numbers. SIDE comes with the TagHelperExtractor and DefinedFeatureExtractor implementing this interface by deault.
- **FeatureTableConsumer**: This interface converts a feature table to some external representation of your data, which can be used by other programs. SIDE comes with consumers which convert feature tables to HTML and ARFF formats.
- **DocumentReaderPlugin**: This interface converts external documents into the UIMA format that SIDE uses internally to store documents. A plugin implementing this interface must be able to produce .xmi UIMA files from some file input. SIDE comes with document readers which read plaintext, CSV, and DeXML files by default.
- **EMPlugin**: These plugins must produce an ordered list of documents based on some evaluation metric. This is used for summarization, where a way of determining which segments to include must exist. SIDE comes with a length counter and a TF/IDF plugin by default.
- **SegmenterPlugin**: This plugin must be able to take as input a string containing all the text of a data set, and produce a list of indices at which to split this text into segments. In addition to native segmentation (each line of each file is one segment), SIDE comes with DocumentSegmenters (each file is one segment), SentenceSegmenter (using a trained English sentence identifier), and WordSegmenter (where each word is treated as a separate segment).
- **VisualizationPlugin**: This plugin must be able to take as input a list of documents that have been automatically classified by some model, and produces a Java Swing component which represents that data in some visual way. SIDE comes with the PieChart, TimeSeries, and Periodic visualizations by default.
- **FeatureAnalysisPlugin**: This plugin takes as input a list of documents that have been automatically classified by some model, and produces a Java Swing component which provides some insight into the behavior of that model. SIDE comes with the SideBySideErrorAnalysis plugin by default.