LightSIDE

Text Mining and Machine Learning User's Manual

Elijah Mayfield
Carolyn Penstein Rosé

Spring 2012

LightSIDE: Text Mining and Machine Learning User's Manual

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 David Adamson, Philip Gianfortoni, Gregory Dyke, Moonyoung Kang, Sourish Chaudhuri, Yi-Chia Wang, Mahesh Joshi, and Eric Rosé for collaboration and contributions to past versions of SIDE.

Table of Contents

A message from the author

Thanks for using LightSIDE! For beginning or intermediate users of text mining and machine learning, I believe it's the package with the best tradeoff between usability and power of anything created.

If you find any outstanding bugs with the software, or with this manual, please contact me and I'll work with you to figure it out. If there's something you want to do with the software that you don't think it currently supports, please contact me and we can talk about how to make it work, and whether it can be added to the next released version. If there's something unusual you want to do that LightSIDE isn't normally designed to handle, please contact me - it's likely I've already tried, and found a way to do it. A lot of people have used this software for a lot of unusual things, and I've talked to most of them.

You will not be wasting my time - LightSIDE is my job. Instead, hopefully, I can make your life easier by ensuring that you get the best possible experience out of the program.

Happy trails.

Elijah Mayfield

elijah@cmu.edu

1 Installation and Setup

Checking your Java VM

In order to use LightSIDE, your computer must have a Java Virtual Machine installed with support for at least Java 6. 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 LightSIDE. 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 LightSIDE. 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 LightSIDE. 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 LightSIDE. 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 & Running LightSIDE

Now that Java 6 is installed on your computer, you can start using LightSIDE. 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/~emayfiel/SIDE.html

Save the file to your desktop or downloads folder for easy access. Now extract the package to a folder, using your favorite archive manager.

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

Windows XP

- Open the LightSIDE folder.
- Double-click the 'run' icon.

Windows 7

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

Mac OS X v10.6

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

Fedora Core 11 Linux

- Click 'Applications' then 'Administration'.
- Click 'Terminal'.
- Type "cd Desktop/SIDE" to navigate to the location where LightSIDE was extracted. If you saved this folder somewhere else, you will have to navigate to it yourself.
- Type "./run.sh"

2 Using LightSIDE: Feature Extraction

Lesson 1: Formatting your data and loading it into LightSIDE

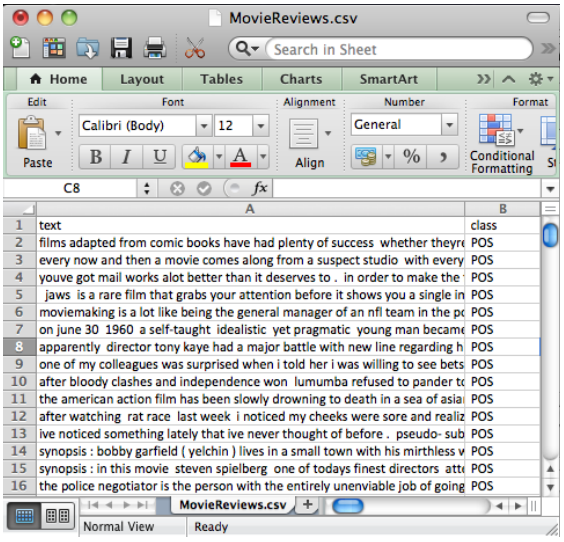
SIDE reads corpora in the form of CSV files, to ensure maximum portability to different systems or users. In addition, multiple files may be added at once so long as their column names are identical.

The first row of your CSV file should be the header names for each column. If your data has text, it should be labelled with the column header "**text**". Likewise, the most common name for a class value to predict is "**class**". If your data does not follow these conventions, LightSIDE will do its best to determine which field is your text column and which is your class value; if it is wrong, you will be able to adjust this through the UI, as explained in Lesson 2.

In your CSV file, a row is assumed to correspond to **one instance**; that row will be assigned one label by the machine learning model that you build in LightSIDE. Before using LightSIDE ensure that your segmentation into rows is the way that you want it to be (for instance, if you want to classify each sentence of a document separately, each sentence must be separated into rows prior to loading your CSV into LightSIDE).

If you would like to include **preprocessed features** in your machine learning model, they should be included as additional columns in your CSV file. These can later be read in to LightSIDE using the Metafeature extractor, as explained in Lesson 2.

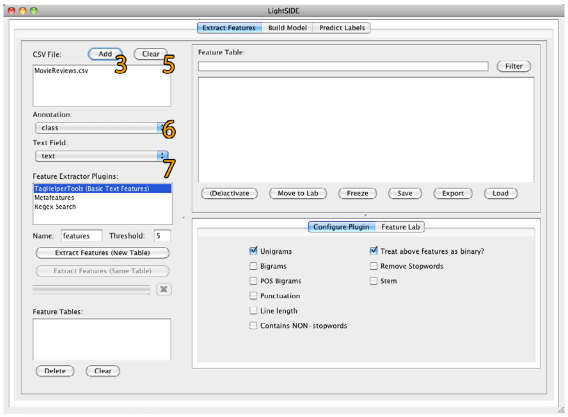
A key assumption made by LightSIDE is that in your data, **every cell** is filled in with some value. Files with missing data may not be processed by LightSIDE as you expect.



	A	B
1	text	class
2	films adapted from comic books have had plenty of success whether theyre	POS
3	every now and then a movie comes along from a suspect studio with every	POS
4	youve got mail works alot better than it deserves to . in order to make the	POS
5	jaws is a rare film that grabs your attention before it shows you a single in	POS
6	moviemaking is a lot like being the general manager of an nfl team in the po	POS
7	on june 30 1960 a self-taught idealistic yet pragmatic young man became	POS
8	apparently director tony kaye had a major battle with new line regarding h	POS
9	one of my colleagues was surprised when i told her i was willing to see bets	POS
10	after bloody clashes and independence won lumumba refused to pander to	POS
11	the american action film has been slowly drowning to death in a sea of asia	POS
12	after watching rat race last week i noticed my cheeks were sore and realiz	POS
13	ive noticed something lately that ive never thought of before . pseudo- sub	POS
14	synopsis : bobby garfield (yelchin) lives in a small town with his mirthless w	POS
15	synopsis : in this movie steven spielberg one of todays finest directors att	POS
16	the police negotiator is the person with the entirely unenviable job of going	POS

Figure 1: The standard CSV format that SIDE is designed to read.

Finally, consider the segmentation of data that you want to use for cross-validation. This is a process, described at length in lesson 4, that allows a user to estimate the effectiveness of their trained model on unseen data. One option that LightSIDE gives is to cross-validate by file, allowing the user to explicitly delimit the subsets to be tested independently. For instance, examples might be separated out by occupation of the writer or speaker. If you wish to make such non-random subsets of your data, create multiple CSVs before loading data into LightSIDE.



CSV File: MovieReviews.csv

Annotation: class

Text Field: text

Feature Extractor Plugins: TagHelperTools (Basic Text Features), Metafeatures, Regex Search

Name: features Threshold: 5

✓ Unigrams
Bigrams
POS Bigrams
Punctuation
Line length
Contains NON-stopwords
✓ Treat above features as binary?
Remove Stopwords
Stem

Figure 2: Loading files into LightSIDE.

To load a data set into LightSIDE:

1. Ensure that the CSV file that you are loading follows the conventions described at the start of this lesson.
2. Open LightSIDE, following the instructions in Chapter 2 of this user's manual.
3. You will see the Feature Extraction tab of LightSIDE open. In the top left corner, click the **Add** button to open a file chooser menu. LightSIDE automatically opens its **data** folder, which contains sample data files.
4. Select the CSV files that you want to open. In this tutorial, we will open the **MovieReviews.csv** file.
5. If at any point you wish to switch datasets without closing LightSIDE, click the **Clear** button. This will allow you to open new files and create new feature tables independently of ones you have already opened.
6. Once you select a file, LightSIDE loads it into internal memory and attempts to guess the annotation you are trying to predict and the column that contains text. Its assumptions are given in the **Annotation** and **Text Field** dropdown menus.
7. Opening these menus allows you to tell LightSIDE to use another column instead by selecting another column. Alternatively, if your data is only columns in the CSV and does not contain text, the **Text Field** dropdown has a final option **[No Text]** which will inform LightSIDE of this.
8. Now that you have loaded your data into LightSIDE, we can build a feature table. Move on to Lesson 2.

Lesson 2: Converting your data set into a LightSIDE feature table

The process of converting text data into a feature table in LightSIDE is done through extractor plugins. Three plugins come available by default. They are the TagHelperTools plugin for extracting n-grams from text, the Metafeatures plugin for converting additional columns from a CSV file into features, and the Regex Search plugin for converting individual searches into features. Each will be explained by the end of this lesson.

To build a feature table in LightSIDE:

1. First, follow the instructions from Lesson 1 to open a data set in LightSIDE.
2. In the middle of the left-hand panel, for now, ensure that **TagHelper Tools** is selected in the **Feature Extractor Plugins** list.
3. In the **Name** field, type in the name that you intend to refer to the extracted feature table by. The default name is simply **features**.
4. The **Threshold** field sets the minimum number of documents that a feature must appear in to be included in your feature table, speeding up processing. The default threshold requires a feature to appear in at least 5 documents.
5. When TagHelper Tools is selected, study the **Configure Plugins** pane. The following options

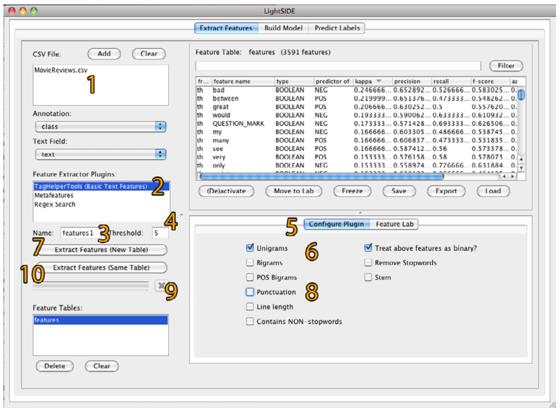
are available. First, in the left hand column, are types of features to extract:

- **Unigrams:** Extracts a bag-of-words representation from your text field. Each feature corresponds to a single word, with a value of true if that feature is present, and false if it is not.
- **Bigrams:** Identical functionality to unigrams, but checks for adjacent pairs of words.
- **POS Bigrams:** Identical functionality to bigrams, but checks for part-of-speech tags for each word rather than the surface form of a word itself.
- **Punctuation:** Creates features for periods, quotation marks, and a variety of other punctuation marks, functioning identically to n-gram features.
- **Line length:** Creates a single numeric feature representing the number of words in an instance.
- **Contains NON-Stopwords:** Creates a single boolean feature representing whether an instance has any contentful words in it.

Second, in the right hand column, are configuration options about how these features should be extracted:

- **Treat above features as binary:** When unchecked, instead of a boolean feature, the extracted features will be numeric and represent the number of times a feature occurs in an instance.
- **Remove Stopwords:** This prevents features from being extracted which correspond to around 100 common words, such as "the" or "and", which tend to be unhelpful for classification.
- **Stem:** Consolidates features which represent the same word in different forms, such as "run", "running", and "runs", into a single feature.

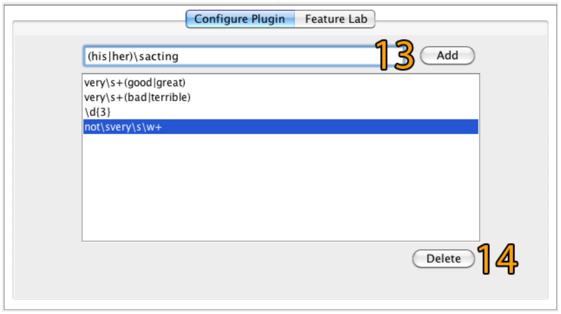
6. For now, leave the options set to the defaults: **unigrams** and **treat above features as binary** should be selected.
7. To create a new feature table, click the **Extract Features (New Table)** button. This takes a few seconds to process; when it finishes, the top right section of the screen will fill with the contents of the feature table. This section will be studied in Lesson 3.



fr...	feature name	type	predictor of	kappa	precision	recall	f-score	ac
th	bad	BOOLEAN	NEG	0.246666...	0.652892...	0.526666...	0.583025...	0.
th	between	BOOLEAN	POS	0.219999...	0.651376...	0.473333...	0.548262...	0.
th	great	BOOLEAN	POS	0.206666...	0.630252...	0.5	0.557620...	0.
th	would	BOOLEAN	NEG	0.193333...	0.590062...	0.633333...	0.610932...	0.
th	QUESTION_MARK	BOOLEAN	NEG	0.173333...	0.571428...	0.693333...	0.626506...	0.
th	my	BOOLEAN	NEG	0.166666...	0.603305...	0.486666...	0.538745...	0.
th	many	BOOLEAN	POS	0.166666...	0.606837...	0.473333...	0.531835...	0.
th	see	BOOLEAN	POS	0.166666...	0.587412...	0.56	0.573378...	0.
th	very	BOOLEAN	POS	0.153333...	0.576158...	0.58	0.578073...	0.
th	only	BOOLEAN	NEG	0.153333...	0.558974...	0.726666...	0.631884...	0.

Figure 3: Feature extraction in LightSIDE.

8. There should currently be 3571 features in your feature table (as shown at the top of the LightSIDE window). Now deselect the **unigrams** option and select **punctuation**.
9. If you wish to stop feature extraction partway through (aborting the extraction process), click the red button next to the status bar.
10. Click the **Extract Features (Same Table)**button in the feature extractor plugin menu. Instead of creating a new feature table, this adds the currently configured plugin's features to the currently open feature table.
11. The remaining plugins are simpler to configure. The **Metafeatures** plugin has no configuration options; it simply pulls all non-class value columns from your CSV file into a feature table.
12. To create features based on regular expressions, first click the **Regex Search** extractor in the feature extractor plugins menu. You will see a text field and an empty list.



```
(his|her)\sacting
```

```
very\s+(good|great)
very\s+(bad|terrible)
\d{3}
not\svery\s\w+
```

Figure 4: Regular expression feature extractor configuration.

13. To create a regex feature, type the search into the top bar, then click **Add**. Your search appears in the bottom list.
14. To remove a search, highlight it and click **Delete**.
15. When you are finished creating searches, create a new feature table or append the searches to your existing feature table as in steps 7 and 9 above.

Now that you have extracted features from your data, you may explore the resulting table in more detail (Lesson 3) or move straight to model building (Lesson 4).

Lesson 3: Exploring and analyzing a feature table in LightSIDE

A key goal of LightSIDE is the ability not just to perform machine learning, but to understand the effectiveness of different approaches to representation of data. This lesson introduces the metrics used by LightSIDE to judge feature quality, and the filtering options for searching for features in a table.

Understanding feature quality metrics in LightSIDE:

1. First, follow the instructions from Lesson 1 and 2 to create a feature table. In this lesson, we are using a unigram and bigram feature space.
2. Now examine the feature table in the top right section of LightSIDE. The window can be stretched to show more columns. The following feature information and metrics are given:

- **From:** The shortened name of the extractor plugin which created this feature.
- **Feature Name:** The name of the feature in this row.
- **Type:** The type of value that this feature can store. Options are boolean (true/false), numeric, and nominal (one of a set number of possible options).
- **Predictor Of:** The simplest metric, this simply tells which annotation label the presence of this feature is most likely to predict compared to random guessing.
- **Kappa:** This metric measures the added value of this feature, from 0 to 1, for predicting the class given in the previous column, compared to random guessing. Kappa value of 1 means that the feature is perfectly correlated with a label, while a negative Kappa value would represent a feature with worse accuracy than flipping a coin.
- **Precision, Recall, and F-Score:** These closely related metrics give an impression of false positive (precision), false negative (recall), and harmonic mean of these two metrics (f-score).
- **Accuracy:** This simple metric tells how well a classifier would perform from using this feature alone to predict class labels for your data set.
- **Hits:** The total number of documents that this feature appears in across your data set.

3. Features can be sorted by these metrics by clicking the title of a column.

Filtering the features shown in the user interface:

4. Sometimes, you will want to check the statistics of a certain subset of your features. To do this, click in the **Filter** text field at the top of the window.
5. To filter by name, simply type in the string of characters you want to see, then press the Enter

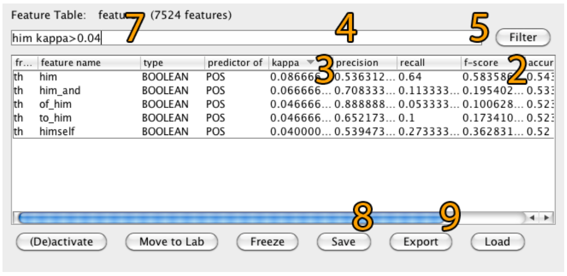
key or click the **filter** button.

For instance, filtering for "**him**" will give not only the unigram "**him**", but also bigrams such as "**to_him**" or longer words such as "**himself**" which contain that substring.

6. To filter based on a numeric column, type that column's name, followed by the operator you want to use, followed by the threshold you wish to set (no spaces anywhere), then press the Enter key or click the **filter** button.

For instance, filtering for "**kappa>0.04**" will remove thousands of features from view in a unigram feature space, as all the features that are no more predictive of a movie review's sentiment than chance (like "**handed**" or "**chief**") are hidden from view.

7. These filters can be combined on a single query, separated by spaces, as shown in Figure 4.


Feature Table: featu (7524 features)

him kappa>0.04

fr...	feature name	type	predictor of	kappa	precision	recall	f-score	accur
th	him	BOOLEAN	POS	0.08666	0.536312...	0.64	0.58358	0.54
th	him_and	BOOLEAN	POS	0.066666...	0.708333...	0.113333...	0.195402...	0.53
th	of_him	BOOLEAN	POS	0.046666...	0.888888...	0.053333...	0.100628...	0.52
th	to_him	BOOLEAN	POS	0.046666...	0.652173...	0.1	0.173410...	0.52
th	himself	BOOLEAN	POS	0.040000...	0.539473...	0.273333...	0.362831...	0.52

Figure 5: Feature quality metrics and filtering in LightSIDE.

8. To save a feature table that you wish to use in another session, or to load a feature table that you previously built, use the **Save** and **Load** buttons in the bottom right corner.

9. To export your feature space to another format to use in another program (such as Weka), click the **Export** button.

3 Using LightSIDE: Model Building

Lesson 4: Training a machine learning model from a feature table

1. First, follow the instructions through Lesson 2 to build a feature table.

2. Then, at the top of the LightSIDE window, switch to the **Build Model** tab.

3. From here, first select the machine learning plugin you wish to use. LightSIDE comes with the Weka toolkit installed by default, as shown in the top left corner of this screen.

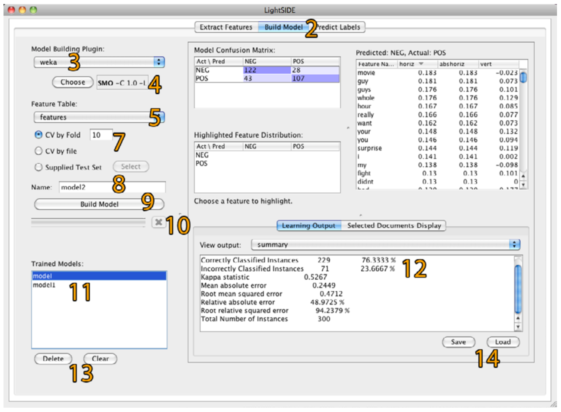
Weka is an extremely robust machine learning package that offers dozens of implementations of various learning algorithms. The three that are most important to remember are the following:

- **NaïveBayes**, in the bayes folder, assumes independence between variables and is based on a generative, probabilistic model of your feature table.

- **SMO**, in the functions folder, is an implementation of support vector machines. This also assumes independence between variables but is a discriminative classifier based on assigning weight to variables.

- **J48**, in the **trees** folder, is an implementation of decision trees, which can model dependence between variables but are often overwhelmed by very large feature tables such as those used in text mining.

Other algorithms which are commonly used and may be helpful for certain tasks are **Logistic**, **MultilayerPerceptron**, and **Winnow** (in the **functions** folder) and **JRip** (in the **rules** folder).



Model Confusion Matrix:

Act \ Pred	NEG	POS
NEG	122	28
POS	43	107

Highlighted Feature Distribution:

Act \ Pred	NEG	POS
NEG		
POS		

Predicted: NEG, Actual: POS

Feature Na...	horiz	abshoriz	vert
movie	0.183	0.183	-0.023
guy	0.181	0.181	0.073
guys	0.176	0.176	0.101
whole	0.176	0.176	0.129
hour	0.167	0.167	0.085
really	0.166	0.166	0.077
want	0.162	0.162	0.073
your	0.148	0.148	0.132
you	0.146	0.146	0.094
surprise	0.144	0.144	0.119
i	0.141	0.141	0.002
my	0.138	0.138	-0.098
fight	0.13	0.13	0.101
didnt	0.13	0.13	0

```
Correctly Classified Instances        229        76.3333 %
Incorrectly Classified Instances       71        23.6667 %
Kappa statistic                      0.5267
Mean absolute error                    0.2449
Root mean squared error                  0.4712
Relative absolute error               48.9725 %
Root relative squared error             94.2379 %
Total Number of Instances              300
```

Figure 6: Model building in LightSIDE.

Advanced users may wish to use **AttributeSelectedClassifier**, **Bagging**, **Stacking**, or **AdaBoostM1**, located in the **meta** folder, which allow you to perform ensemble learning or feature selection within LightSIDE.

4. For this tutorial, we will choose the **SMO** classifier. Ensure that it is selected in Weka's **Choose** menu.

5. Select the feature table that you would like to use for training in the **Feature Table** dropdown.

6. Now choose the type of cross-validation that you would like to use to test the validity of the model:

- **CV by Fold:** The default setting performs N-fold cross validation. N models are built, each on (N-1/N)% of the data, and tested on the remaining

(1/N)%. Instances are chosen in a round-robin fashion, for instance, in 5-fold cross-validation, the 1st, 6th, 11th, etc. instances are chosen for testing in the first fold, followed by the 2nd, 7th, 12th, etc. held out in the second fold. 10-fold cross-validation is selected by default.

- **CV by File:** This setting assumes that your feature table was built from multiple files. In each fold of cross validation, all but one file is used for training, and the remaining file is used for testing. This repeats through each file in your data set.
- **Supplied Test Set:** One model is built, on your full training set, and it is evaluated on a second feature table from a file that you select here.

7. For this tutorial, leave the **CV by fold** option selected and the number of folds set at 10.
8. If you wish to specifically name this model, change it in the **Name** text field.
9. Click **Build Model** to train and evaluate a model.

For large feature tables (either many documents or many features), this may take a large amount of memory (greater than 1 GB) and may take several minutes to complete. Be patient and close other programs as necessary. For our example feature table, however, model building should only take a few seconds.

10. To stop a model prematurely because it is taking too long, or using too much memory, or performing poorly in cross-validation, click the red button.
11. When the model is finished, your model will appear in the bottom left corner in the **Trained Models** list.
12. At the same time, the right hand side of the window will populate with data about the model. For now, the most important information is in the bottom panel, giving the accuracy (**Correctly Classified Instances**) and improvement over chance (measured by **Kappa statistic**). If you followed the instructions in Lesson 2, your performance will be around 75% accuracy and .5 kappa.
13. To clean out models that you don't intend to use again, use the **Delete** or **Clear** buttons in the bottom left corner.
14. To save a model for future use, or to load a model that you've trained previously, click the **Save** and **Load** buttons in the bottom right corner.
15. For more detailed analysis of your model's performance, move on to Lesson 5.

Lesson 5: Adding labels to unlabeled data with a trained model

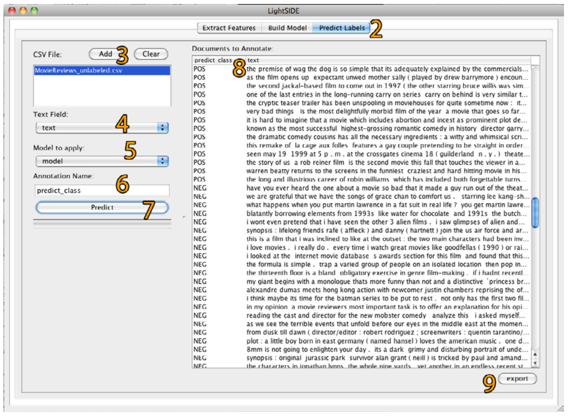
A key feature of models built with LightSIDE is that they are not just for evaluation of your labeled data set. The models that you train can be used to annotate more data in a fraction of the time that it would otherwise take a trained expert, at roughly the accuracy given in the summary from the Model Building panel.

1. First, follow the instructions through Lesson 4 to train a machine learning model.
2. Then, at the top of the LightSIDE window, switch to the **Predict Labels** tab.
3. Select the CSV file that you wish to annotate in the top left corner with the **Add** button. If you are extracting metafeatures from the CSV, column titles must match exactly in order for feature spaces to align between documents.
4. Ensure that the **Text Field** option has correctly guessed the column to use for text input (or choose **[No Text]** if you are not using text feature extraction).
5. Choose which model will annotate your data in the **Model to apply** field.
6. Select a name for the predicted column in the

Annotation Name field.

7. Click **Predict** to use the selected model to annotate the documents in the selected file.
8. When annotation is finished, the predicted labels will be displayed in the right window, next to the document they correspond to.
9. If you wish to export these predicted labels to a new CSV document, click the **Export** button in the bottom right corner.

If the performance that you're getting so far is insufficient for your liking, continue to Lessons 6 and 7 to improve your models.



CSV File: MovieReviews_unlabeled.csv

Text Field: text

Model to apply: model

Annotation Name: predict_class

Documents to Annotate:

predict_class	text
POS	the premise of wag the dog is so simple that its adequately explained by the commercials...
POS	as the film opens up expectant unwed mother sally (played by drew barrymore) encoun...
POS	the second jackal-based film to come out in 1997 (the other starring bruce willis was sim...
POS	one of the last entries in the long-running carry on series carry on behind is very similar t...
POS	the cryptic teaser trailer has been unspooling in moviehouses for quite sometime now : it...
POS	very bad things is the most delightfully morbid film of the year a movie that goes so far...
POS	it is hard to imagine that a movie which includes abortion and incest as prominent plot de...
POS	known as the most successful highest-grossing romantic comedy in history director garry...
POS	the dramatic comedy cousins has all the necessary ingredients : a witty and whimsical scri...
POS	this remake of la cage aux folles features a gay couple pretending to be straight in order...
POS	seen may 19 1999 at 5 p . m . at the crossgates cinema 18 (guilderland n . y .) theate...
POS	the story of us a rob reiner film is the second movie this fall that touches the viewer in a...
POS	warren beatty returns to the screens in the funniest craziest and hard hitting movie in his...
POS	the long and illustrious career of robin williams which has included both forgettable turns...
NEG	have you ever heard the one about a movie so bad that it made a guy run out of the theat...
NEG	we are grateful that we have the songs of grace chan to comfort us . starring lee kang-sh...
NEG	what happens when you put martin lawrence in a fat suit in real life ? you get martin lawre...
NEG	blatantly borrowing elements from 1993s like water for chocolate and 1991s the butch...
NEG	i wont even pretend that i have seen the other 3 alien films . i saw glimpses of alien and...
NEG	synopsis : lifelong friends rafe (affleck) and danny (hartnett) join the us air force and ar...
NEG	this is a film that i was inclined to like at the outset : the two main characters had been inv...
NEG	i love movies . i really do . every time i watch great movies like goodfellas (1990) or rai...
NEG	i looked at the internet movie database s awards section for this film and found that this...
NEG	the formula is simple . trap a varied group of people on an isolated location then pop in...
NEG	the thirteenth floor is a bland obligatory exercise in genre film-making . if i hadnt recentl...
NEG	my giant begins with a monologue thats more funny than not and a distinctive `princess br...
NEG	alexandre dumas meets hong kong action with newcomer justin chambers reprising the of...
NEG	i think maybe its time for the batman series to be put to rest . not only has the first two fil...
NEG	in my opinion a movie reviewers most important task is to offer an explanation for his opi...
NEG	reading the cast and director for the new mobster comedy analyze this i asked myself...
NEG	as we see the terrible events that unfold before our eyes in the middle east at the momen...
NEG	from dusk till dawn (director/editor : robert rodriguez ; screenwriters : quentin tarantino/...
NEG	plot : a little boy born in east germany (named hansel) loves the american music . one d...
NEG	8mm is not going to enlighten your day . its a dark grimy and disturbing portrait of unde...
NEG	synopsis : original jurassic park survivor alan grant (neill) is tricked by paul and amand...
NEG	the characters in jonathan lynns the whole nine yards yet another in an endless recent st

Figure 7: Labeling new data with LightSIDE.

Lesson 6: Error analysis of a machine learning model in LightSIDE

In some cases, a basic model (utilizing unigrams alone or another simple representation) is sufficient for a model's accuracy. However, in many cases, we wish to improve upon this performance. One of the best ways to do this is to understand what shortcomings the current representation has, and build new features (either individually or as a class of many features) and use them in addition to the simple representation. LightSIDE offers many tools for understanding the strengths and weaknesses of a model.

1. First, follow the instructions through Lesson 4 in order to have a built model open in LightSIDE.
2. The first step in understanding a model's errors is to check the **confusion matrix**. This is visible in the top center of the **Model Building** window.

In the confusion matrix, columns corresponding to the possible labels you are trying to classify show your model's predictions. Rows correspond to the same labels, but show the true value of the data. For instance, a cell at the intersection of row POS and column NEG shows the number of instances that were originally labeled POS, but that the model incorrectly classified as NEG. In our test example, there were 36 such cases.

3. Click a cell in the confusion matrix. This populates the list on the top right of the Model Building window. This list now shows the most "confusing" features, as measured by multiple metrics.

The features in this list identify those features which make the instances in the highlighted cell most deceiving. For instance, a feature which is highly predictive of POS might sometimes occur in a different context in NEG documents. If this happened, but not frequently enough to decrease the model's certainty of a feature's predictiveness, it is likely to occur unusually frequently in the clicked cell, bringing these features to the top of the list.

4. Click a feature in the top right list to populate the bottom confusion matrix, located in the center of the LightSIDE window. This shows the distribution of that feature in the instances in each cell.

Example: When reading the value of **Horizontal Comparison**, near the top of the list for predicted POS documents which were actually NEG, we find the word "**becomes**". When clicked, the bottom confusion matrix in LightSIDE displays the following information:

Distribution of feature “**becomes**”:

Actual \ Pred	NEG	POS
NEG	.159	.351
POS	.139	.246

What does this tell us? The distribution shows that among the 113 documents correctly identified as NEG, the term “becomes” only appeared in 15.9% of those documents. On the other hand, in documents correctly classified as positive, the term occurs in 24.6% of the documents. And for documents in the error cell that we’ve highlighted? It occurs in a much higher 35.1% of the 37 documents (working out to 13 of those 37).

This is likely to mean that the word “become” has a tendency to occur in contexts which are predictive of sentiment but can be deceptive.

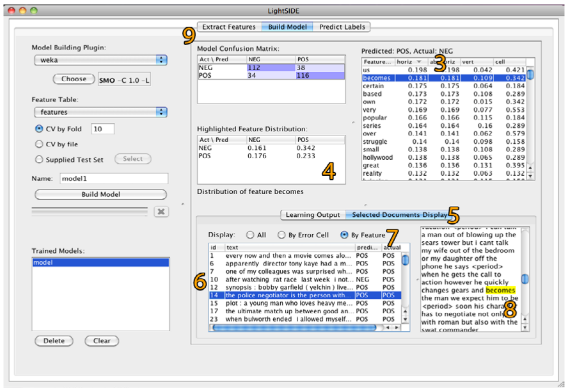
Vertical Comparison performs this same comparison, but instead of measuring against the diagonal across from an error cell, it measures against the diagonal cell above or below the highlighted cell.

5. One way to better understand these aggregate statistics is to look at examples in text. For this, click the “Selected Documents Display” tab at the bottom right of the LightSIDE window.

6. This display shows a list of instances on the left hand side, and focuses in on a single clicked instance on the right hand side. To test this, click a line in the list. The LightSIDE window can be stretched to give more room to the instance being highlighted.

7. The left hand list shows the predicted and actual label for each instance, and can be filtered in three different ways:

- **All:** This option simply shows all instances in the order they were listed in the original CSV.
- **By Error Cell:** The documents which occur in the error cell that is currently highlighted at the top of the screen will be displayed.



Model Building Plugin: weka

SMO -C 1.0 -L

Feature Table: features

(●) CV by Fold 10
( ) CV by file
( ) Supplied Test Set

Name: model1

Trained Models: model

Model Confusion Matrix:

Act \ Pred	NEG	POS
NEG	112	38
POS	34	116

Highlighted Feature Distribution:

Act \ Pred	NEG	POS
NEG	0.161	0.342
POS	0.176	0.233

Distribution of feature becomes

Predicted: POS, Actual: NEG

Feature...	horiz	abs horiz	vert	cell
us	0.198	0.198	0.042	0.421
becomes	0.181	0.181	0.109	0.342
certain	0.175	0.175	0.064	0.184
based	0.173	0.173	0.108	0.289
own	0.172	0.172	0.015	0.342
very	0.169	0.169	0.077	0.553
popular	0.166	0.166	0.115	0.184
series	0.164	0.164	0.16	0.289
over	0.141	0.141	0.062	0.579
struggle	0.14	0.14	0.098	0.158
small	0.138	0.138	0.108	0.289
hollywood	0.138	0.138	0.065	0.289
great	0.136	0.136	0.131	0.395
reality	0.132	0.132	0.063	0.132

Learning Output | Selected Documents Display

Display: ( ) All ( ) By Error Cell (●) By Feature

id	text	predi...	actual
1	every now and then a movie comes alo...	POS	POS
6	apparently director tony kaye had a m...	POS	POS
7	one of my colleagues was surprised wh...	POS	POS
10	after watching rat race last week i not...	NEG	POS
12	synopsis : bobby garfield (yelchin) live...	POS	POS
14	the police negotiator is the person with...	POS	POS
15	plot : a young man who loves heavy me...	POS	POS
17	the ultimate match up between good an...	POS	POS
23	when bulworth ended i allowed myself...	POS	POS

a man out of blowing up the sears tower but i cant talk my wife out of the bedroom or my daughter off the phone he says <period> when he gets the call to action however he quickly changes gears and becomes the man we expect him to be <period> soon his character has to negotiate not only with roman but also with the swat commander

Figure 7: Error analysis in LightSIDE’s Model Building pane.

- **By Feature:** Only instances which contain the feature you currently have selected in the top right corner of LightSIDE will be displayed.

8. For surface features such as n-grams, which have a defined start and end point, the bottom right window indicates their location in a document. Selecting a feature in the top right interface highlights that feature in the bottom right panel.

9. For the next step, if you trained a model using Weka’s **SMO** classifier, switch back to the **Extract Features** tab.

10. In the feature table list, scroll to the far right. You will now see a new column titled “**weights1**”. This corresponds to the SVM weight for each feature.

What does the SVM weight tell us about a feature? Primarily, whether it is actually having an impact on a model. If a feature looks highly confusing by horizontal comparison but is being given a weight near zero, then it is unlikely to be a true source of confusion for the model.

Using these tools can make it easier to recognize what a model is doing and what instances are being misclassified and for what reasons. The next step is to attempt to change a feature table to improve its ability to represent your data. For instruction on how to do this, move on to Lesson 7.

Lesson 7: Constructing features with LightSIDE's feature lab

LightSIDE gives you the opportunity to make many changes to a feature table. Here we highlight two different possibilities: removing features that are superfluous or detrimental to performance, and introduction of new features which incorporate relationships between existing features.

Deactivating features from a feature table

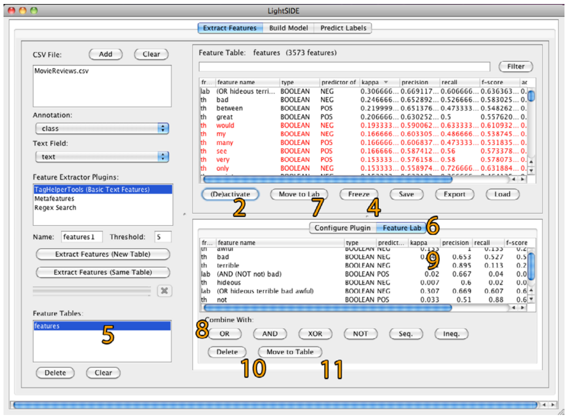
1. Before beginning, ensure that a feature table has been built in LightSIDE.
2. If a feature seems to be detrimental to performance, highlight it in the feature table exploring interface in the **Extract Features** tab, then click the **(De)activate** button. It will turn red.
3. Reversing this decision can be done by clicking the **(De)activate** button to reactivate any highlighted features.
4. Once you have settled on what features to remove, click **Freeze**.
5. A new feature table will now appear in the bottom left corner, which does not contain the deactivated features. This new feature table can be passed on to the Model Building window.

Constructing new features with the Feature Lab

6. If you have an idea for a new feature combining multiple sources of evidence, click the **Feature Lab** tab on the bottom of the screen.
7. Now, for each component of your new feature, find it in the top panel (filtering will be helpful), highlight it, and click the "**Move to Lab**" button.

The Feature Lab allows combinations of features using various tools:

- **OR, AND, XOR, NOT:** Takes as arguments any number of features, and combines them using this boolean operator. For instance, combining "**awful**" and "**bad**" with an **OR** operator would create a new feature which recognizes either word in a document, effectively merging the two features.
- **Seq:** Takes as arguments two features, and checks whether they occur in consecutive instances. This feature assumes that your data is sequential, such as in a conversation or forum thread. In a set of unrelated data, such as detached movie reviews which are not related to one another, such a feature is not useful.
- **Ineq:** Takes as an argument a single numeric feature. When clicked, this button will open a popup asking you to define a threshold (for instance, "**>0.5**". It will then produce a boolean feature which is true every time the numeric value of the selected feature matches that threshold.



CSV File: MovieReviews.csv

Annotation: class

Text Field: text

Feature Extractor Plugins: TagHelperTools (Basic Text Features), Metafeatures, Regex Search

Name: features1 Threshold: 5

Feature Tables: features

Feature Table: features (3573 features)

fr...	feature name	type	predictor of	kappa	precision	recall	f-score	ac
lab	(OR hideous terri...	BOOLEAN	NEG	0.306666...	0.669117...	0.606666...	0.636363...	0.
th	bad	BOOLEAN	NEG	0.246666...	0.652892...	0.526666...	0.583025...	0.
th	between	BOOLEAN	POS	0.219999...	0.651376...	0.473333...	0.548262...	0.
th	great	BOOLEAN	POS	0.206666...	0.630252...	0.5	0.557620...	0.
th	would	BOOLEAN	NEG	0.193333...	0.590062...	0.633333...	0.610932...	0.
th	my	BOOLEAN	NEG	0.166666...	0.603305...	0.486666...	0.538745...	0.
th	many	BOOLEAN	POS	0.166666...	0.606837...	0.473333...	0.531835...	0.
th	see	BOOLEAN	POS	0.166666...	0.587412...	0.56	0.573378...	0.
th	very	BOOLEAN	POS	0.153333...	0.576158...	0.58	0.578073...	0.
th	only	BOOLEAN	NEG	0.153333...	0.558974...	0.726666...	0.631884...	0.

fr...	feature name	type	predict...	kappa	precision	recall	f-score
th	awful	BOOLEAN	NEG	0.133	1	0.133	0.2
th	bad	BOOLEAN	NEG	[illegible]	0.653	0.527	0.5
th	terrible	BOOLEAN	NEG	[illegible]	0.895	0.113	0.2
lab	(AND (NOT not) bad)	BOOLEAN	POS	0.02	0.667	0.04	0.0
th	hideous	BOOLEAN	NEG	0.007	0.6	0.02	0.0
lab	(OR hideous terrible bad awful)	BOOLEAN	NEG	0.307	0.669	0.607	0.6
th	not	BOOLEAN	POS	0.033	0.51	0.88	0.6

Figure 8: Feature construction and deactivation in LightSIDE.

8. To combine features in the feature lab, highlight all of the features you would like to combine, then click the button of the appropriate operator.

9. Note that the quality of these features is automatically calculated in columns as soon as they are created.

A key feature of these combination functions, especially the boolean operators, is that they can be constructed into larger trees, for instance, a feature such as "**(AND good (NOT bad))**".

In Figure 8, a rudimentary lexicon of negative terms has been built: "(OR awful bad hideous terrible)". This feature turns out to have a higher kappa than any of the unigrams individually, which intuitively makes sense; this allows grouping of many weak sources of evidence into a single stronger source of evidence.

10. To delete component parts that are no longer necessary (and are cluttering your view of the lab), highlight them and click the **Delete** button.
11. When a feature has been built to your satisfaction, move it back into the feature table with the **Move to Table** button.
12. The revised feature table can now be trained using the same steps outlined in Lesson 4.

4 Using LightSIDE: Plugin Writing

For users with some Java programming experience and an idea for a more complicated representation of their documents, plugin writing is the next step in the feature space development process if TagHelperTools, metafeatures, and the Feature Lab are insufficient.

Begin by opening a new project in Eclipse or another IDE, and adding the LightSIDE source folder to your build path. This will give your new project access to the interfaces that are necessary to write a plugin.

LightSIDE internally views a feature table as a set of objects of type `FeatureHit`. These objects contain within them a `Feature` object, a value (which can be any object, but is usually a `double` or a `String`), and an `int` documentIndex. These feature hits are specific to a given data file, passed to your plugin as a `DocumentListInterface.`

A `Feature` object is a more abstract concept, representing the dimension in your feature table that this feature hit corresponds to. They need to know their `String` featureName, their `Type` (from the options `NUMERIC`, `BOOLEAN`, and `NOMINAL`), and if they are nominal, they need to know what possible values they can take.

Each instance in a document list can be represented as the set of all feature hits with that instance's document index. At feature table construction time, that instance will be given the value of those hits in its feature space; all other dimensions will be set to 0, or false.

To write a feature extraction plugin, create a new class extending **FeaturePlugin**. This is an abstract class that already includes some functionality. However, you must provide the following methods:

- `String getOutputName()`

The short prefix string displayed throughout the LightSIDE GUI for features created by your plugin.

- `List<FeatureHit> extractFeatureHits(DocumentListInterface documents, JLabel update)`

This method must iterate through your data file, extracting `FeatureHit` objects for each instance, creating `Feature` objects as necessary. Creating a new feature should be done through the static method `Feature.fetchFeature(String prefix, String name, Feature.Type type)` in order to allow caching.

- `Component getConfigurationUIForSubclass()`

This method returns any Swing component, and will display it in the bottom right corner for users to edit configuration options. At feature table creation time, a method `uiToMemory()` is called, which by default does nothing. If you wish to move information from the UI to your extraction code, it should be done by overriding this method.

Your plugin also knows about a field `boolean halt`. This field is set to true when the red button is clicked to stop feature extraction. To make that button fuctional, you must set up points in your code for your plugin to fail gracefully and return null.

You’ve reached the end of the manual!