NLP Final Project

Fall 2017, Due Friday, December 15 by midnight (last day of exams)

For the final project, choose from three types of projects:

1. (simple) sentiment classification experiment + design a chatbot
2. (extensive) classification experiments on a choice of data or
3. (simple) sentiment classification experiment + lexical chain implementation.

You may also propose your own project, and you are encourage to work in groups of 2-3 people. For each person that is added to the group, plan on increasing the tasks by about 50%.

Simple Sentiment Classification (for options 1 and 3)

For this part, you are to do at least two experiments in the NLTK on the movie review sentence data in which you compare the baseline performance of using just word features with some modified or additional features that you implement.

You will need to write a Python feature function to generate the features that you choose. One idea is to vary the representation of the subjectivity lexicon features or to use positive and negative emotion words from LIWC. Another idea would be to extend the representation of the negation features up to the next punctuation. Another idea would be to combine one or more of these, including bigrams or the POS tag counts.

Note that you will first want to develop your feature function(s) until they work. You can run your experiments by using a random training and test set as we have done in lab, but you are encouraged to run cross-validation to get classifier accuracies. Note that three runs are the absolute minimum, and that you need to make a series of related feature comparison runs for top grades.

Include a description of your experiments in the final project report, together with the feature function code that you wrote and the accuracies of the classifications.

Option 1. Design a plan understanding module (half a chatbot)

For this task, you will design a set of tasks for a chatbot to understand the user's plan, using the Microsoft software LUIS (Language Understanding Intelligent Service). But the software doesn't apparently give you interactive tools to define the other half, where the chatbot responds to the user – this part is implemented through programming.

One major part of this task is identify a user service that could use a chatbot and to develop a user scenario for what users would ask or discuss with the service. In the LUIS tutorials, they imagine that there is a travel planner service and that two example of things that users would ask are to book a flight or to get weather information. LUIS calls these things intents. Each intent can have entities associated with that intent, such as locations and dates. Some of these are prebuilt to help you.

For the project then, you would develop a system of intents with entities, and then help the system to define how to understand intents and entities from the user's natural language messages. Their interactive tool allows you to give example messages, called utterances, and they use that for training data to train an understanding module. You can help it by labeling examples that it doesn't understand. You can also define "features" to help the training by giving more terms or regular expressions to define terms that will expand the range of utterances that the system can understand for an intent. As I understand it, the system doesn't keep a "session", but we can still test it with successive messages that talk about different user intents.

Microsoft has published two YouTube tutorial videos.
Basics:
https://www.youtube.com/watch?v=jWeLajon9M8&index=4&list=PLD7HFcN7LXRdHkFBFu4stPPeWJcQ0VFLx
Advanced:
https://www.youtube.com/watch?v=39L0Gv2EcSk&index=5&list=PLD7HFcN7LXRdHkFBFu4stPPeWJcQ0VFLx

After the system is trained, you can "publish" it, where it will be located at a URL and users can type in a message and see what intents and entities the system understands.

We expect to schedule our own tutorial session where we can help each other to learn LUIS.

Your goal in developing the language understanding system will be to design a set of user scenarios and to enable the system to understand a wide range of language expression for the user to give messages to the system that conveys their intent and its associated entities. For the final report, you will describe your user scenarios, give examples of utterances and what the system understands from them, and discuss the range of language expression that you tried. [TBD: an estimate of how many intents and entities, how many language examples, etc.?]

Since this is a new type of project for us, your report can also include a reflection on this type of software and its ability to help developer with language understanding for chatbots.

Option 2. Processing and Classification of Sentiment or other Data

For this task, you should choose to work on classification a datasets: the email spam data, the Kaggle movie review data, the SemEval Twitter data or a dataset of your choosing (with permission from instructor).

Three Datasets for Text Classification Tasks

For this option, three datasets are provided for your choice (unless you propose your own).

1. Detection of SPAM in email

The first dataset is one produced for detecting Spam emails from the Enron public email corpus. In addition to some small numbers of Spam already in the corpus, additional spam emails were introduced into each user's email stream in order to have a sufficient number of spam examples to train a classifier. The non-Spam emails are labeled “ham”. (See this paper for details: http://www.aueb.gr/users/ion/docs/ceas2006_paper.pdf) The dataset that we have was gleaned from their web site at http://www.aueb.gr/users/ion/data/enron-spam/.

Although there are 3 large directories of both Spam and Ham emails, only the first one is used here with 3,672 regular emails in the “ham” folder, and 1,500 emails in the “spam” folder.

2. Kaggle competition movie review phrase data, labeled for sentiment

This dataset was produced for the Kaggle competition, described here: https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews , and which uses data from the sentiment analysis by Socher et al, detailed at this web site: http://nlp.stanford.edu/sentiment/. The data was taken from the original Pang and Lee movie review corpus based on reviews from the Rotten Tomatoes web site. Socher’s group used crowd-sourcing to manually annotate all the subphrases of sentences with a sentiment label ranging over: “negative”, “somewhat negative”, “neutral”, “somewhat positive”, “positive”.

Although the actual Kaggle competition is over, the data is still available at https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews/data. We are going to use the training data “train.tsv”, and some test data is also available “test.tsv”. There appear to be 156,060 phrases in the training data file, and one of the challenges will be to choose an appropriate subset for processing and training.

3. SemEval shared task Twitter data, labeled for sentiment

The Semantic Evaluation conference, SemEval, has recently added sentiment detection in Twitter messages and other social media genres to its shared tasks. For example, in 2014, it ran as Task 9, sub-task B, Message Polarity Classification: http://alt.qcri.org/semeval2014/task9/. The data was manually labeled and each dataset is available as a Twitter id paired with the manual label. There are 5 labels used for each message: “positive”, “negative”, “objective”, “neutral”, “objective OR neutral”.

To actually get the tweets, there is a script you can run to get them from Twitter itself. If a Twitter user has retracted their tweet, then Twitter will no longer send it out and it is marked as “Not Available”.

The dataset given here was collected in the Spring 2014 from Twitter. You are given access to this data through Blackboard, and it should not be made publically available as it is subject to the Twitter terms of service.

The Text Classification Tasks

Step 1: For your choice of dataset, you will first process the text, tokenize it and choose whether to do further pre-processing or filtering. If you do some pre-processing or filtering, then using the text with and without it can be one of your experiments.

For each dataset, there is a program template that reads the data. You should run the program on your data of choice and investigate some of the data in order to choose pre-processing or filtering.

Step 2: The second step is to produce the features in the notation of the NLTK. For this you should write feature functions as we have been doing in lab. You should start with the "bag-of-words" features where you collect all the words in the corpus and select some number of most frequent words to be the word features.

Now use the NLTK Naïve Bayes classifier to train and test a classifier on your feature sets. You should use cross-validation to obtain precision, recall and F-measure scores. Or you can choose to produce the features as a csv file and use Weka or Sci-Kit Learn to train and test a classifier, using cross-validation scores.

Step 3: Experiments

A. For a base level completion of experiments, carry out at least several experiments where you use two different sets of features and compare the results. For example, you may take the unigram word features as a baseline and see if the features you designed improve the accuracy of the classification. Here are some of the types of experiments that we have done so far:

- filter by stopwords or other pre-processing methods
- representing negation (if using twitter data, note the difference in tokenization)
- using a sentiment lexicon with scores or counts: Subjectivity or LIWC
- combine the use of sentiment lexicons
- different sizes of vocabularies
- POS tag features

B. Choose an additional, more advanced type of task from this list, or propose your own

- using Weka or Sci Kit Learn classifiers with features produced in NLTK
- using an additional type of lexicon besides Subjectivity or LIWC
- in addition to using cross-validation on the training set, train the classifier on the entire training set and test it on a separately available test set (only the SemEval data has these)
    - note that you must save the vocabulary from the training set and use the same for creating feature sets for the test data
- implement additional features
    - in the email dataset, use word frequency or tfidf scores as the values of the word features, instead of Boolean values
    - use POS tagging from the ARK on Twitter
    - twitter emoticons or other features based on internet usage or informal text, such as repeated letters, or all caps words

Step 4: Write a report that describes the data processing, the features and the classification experiment(s). As one of your experiments, you may instead compare results from different classifier algorithms in Weka or Sci-Kit Learn.

Option 3. Programming Projects

Write a program to process text and discover lexical chains. We will work from the paper: Barzilay and Elhadad, “Using Lexical Chains for Text Summarization”, 1999. We will identify a subset of their algorithm that is reasonable to implement in NLTK using WordNet. More details are given in the ProgrammingProjects document in Blackboard.

This algorithm can be used with different word similarity measures and threshold scores. You must identify a corpus of documents to test your algorithm and to run a series of experiments with different similarity measures and thresholds for scores.

The programs should be well-documented in a report that is handed in with the code and with a discussion of test results.

What to Hand In

Each person should hand in a report with the description of all that you did and the discussion of the results. Also submit any python programs that you write, particularly to show additional features that you implement, or the annotation files that you did.

If you worked in a group, the report should include which tasks each person worked on. Each person in the group should submit the (same) report . (Having every person submit the report makes it easier for me to record individual grades in Blackboard.)

As usual, submit these documents to the Blackboard system by the end of the day on the due date.