Linguistics 439/539
Fall 2015

Final Project / Assignment #6 (Preliminary)

Note: **Start today**, *and make use of the office hours if you have any questions. DO NOT leave this assignment until the last minute!*

1. **q1.py: Information Retrieval and Machine Learning: SPAM E-mail Classification Task**
   Your task is to compete in a mini-version of an official popular NLP conference evaluation, the TREC SPAM task. You will use NLTK's machine learning frameworks and your knowledge of computational linguistics to learn to classify a corpus of e-mails as either normal e-mails or spam.

   - **Readings:** You should begin your readings with the NLTK book's Chapter 6, *Learning to Classify Text*. This is available online at: `http://www.nltk.org/book/ch06.html`.
   - **Corpus:** Your corpus is the official Text REtrieval Conference (TREC) 2007 Public Spam Corpus, available here: `http://plg.uwaterloo.ca/ gvcormac/treccorpus07/`
   - **Gold Data:** The correct classifications for each message (used for training the classifier, then evaluating its performance) are listed in the gold file located at `/trec07p/partial/index` .
   - **Train/Development/Test splits:** Use the first 1,000 messages in the corpus directory `/trec07p/data/inmail.*` as your training corpus, and messages 1,000-2,000 (1,000 total) as the development corpus. Use the development corpus to test and tune your work as you go.

     Once you're happy with the performance on the development set, **test** on messages 2,001-3,000 (1,000 total). You should only ever test on this corpus one time, at the very end of your development cycle, *when you are happy with performance on the development set.* This is done on the honour system. It is normal for performance to differ a little between development and test sets.
   - **Features:** Your main task is to design features that the classifier can use to identify SPAM messages from normal messages. You must include at least one or two features that make use of the cosine similarity between

vectors. You may wish to implement a *tf.idf* weighting scheme for increased discriminatory power. Others features might be based on lexical cues (i.e. seeing a particular word or set of hand-coded words that are common in SPAM than in normal messages), syntactic features, part of speech features, or any other features that you'd like to implement. The only limit is that it must not require any external libraries (other than NLTK) to run.

Some examples for IR features might include the maximum cosine similarity between a given message in the dev/test corpus (i.e. the "query" vector) with the vectors for all spam messages in the training corpus (the "document collection"). This would be a single feature. Another feature might be the same, but taking the maximum cosine similarity between all normal (i.e. non-spam) messages in the corpus.

539 Only: You must design and include at least 5 features in your classifier.

- **Off-limit Features:** Many of the normal (i.e. non-spam) messages contain explicit tags in the message header that detect if the message is SPAM or not using SpamAssassin (e.g. `X-Spam-Checker-Version: ...`). You may not use this information in your framework – you have to design your own features and detection framework based on the text of the message.
- **Classification Framework:** Feel free to use any out-of-the-box classification framework that NLTK offers (linear regression, naive bayes, decision trees, perceptron, etc). You might try how each one works on the development set, and choose the one that seems to work best.
- **Evaluation:** Include the performance (absolute number correct and percent correct) of your system on the development and test sets, in separate tables, as in Table 1.
- **Write-up:** In your write-up, include a description of each feature that you have included, and example scores for that feature from one spam and one normal message from the training corpus to illustrate how it works.
- **A note on overfitting:** With machine learning systems, the goal is to try and define general features that generalize to novel circumstances, rather than features that work especially well for the training and development corpora. Features that are too specialized to the training or

Performance on dev.	True Spam	True Normal
Classified as Spam	450 (75%)	150 (25%)
Classified as Normal	200 (40%)	300 (60%)

Performance on test	True Spam	True Normal
Classified as Spam	... (...%)	
Classified as Normal		

Table 1: Example performance table, showing that some model correctly classifies spam 75% of the time, and correctly classifies normal messages 60% of the time (on the development corpus). Performance on the test corpus not shown.

development corpora are said to be overfitted, which usually results in a substantial decrease in performance from development to test set. (Although, sometimes you can get this performance decrease anyway, especially if the corpus size is small, and you're unlucky).

- **Bonus for Top Performance:** The top 10 performing systems (on the test questions) will receive a bonus of +10% on this assignment. The top-scoring system will receive the coveted 539 Information Retrieval Award! (Bonuses only apply to on-time assignments, I'll give out the award in class on December 8$^{\text{th}}$.) *Make sure your performance tables are easy-to-find, properly formatted and calculated, and easy to read!*

Things to remember:

1. The due date is **Friday December 4$^{\text{th}}$ by 5pm**. You should aim to have the entire assignment complete and submitted by then. If you are having last-minute issues, you may take up to *Monday December 7th by 5pm* without late penalty. All submissions must be through D2L.
2. Code must be in the form of working/runable .py files that you submit. For this assignment, everything will be evaluated with Python 2.7.
3. Please include your name, the question number, and a very brief description at the top of each .py file.
4. Please **include brief comments in your Python source code** so that I can easily follow what the functions are doing. Try to write readable (instead

of complex or nuanced) code.

5. Prose/graphs/etc. should be submitted as a **single PDF**, with question number clearly labelled.
6. Runtime: Your programs **should run in seconds or possibly a few minutes, not hours or days**.
7. Submission: Please ZIP up all the files included with your assignment submission, and label it as `firstname_lastname_assignment6.zip`.
8. Remember: No late assignments – **do not wait until the last minute to do this**. We are working against the final mark submission deadline – assignments will not be accepted past December 8^{th}.