ITCS 4111/5111 Introduction to Natural Language Processing

Assignment 2 Due: October 6th, 2019 11:59 pm

Total points: 200

Your submissions should be in a single python notebook. Make sure code is documented well. All submissions should be uploaded to Canvas. Email submissions will not be considered for grading.

The problems marked with ** are optional for students enrolled in the ITCS 4111 section of this course, all other problems are mandatory. ALL problems are mandatory for the students enrolled in ITCS 5111.

The learning goals of this assignment are:

- Understand and implement common NLP evaluation methods - precision, recall and F1 measure – as discussed during lecture.
- Employ experimental design practices in NLP. Splitting the corpus into training/development/test sets, implementing baselines to determine how difficult the task is, and experimenting with features and models.
- Getting acquainted with sklearn; commonly used machine learning Python package.
- Learning about basic text classification algorithms
- Learning about vector semantics and getting familiarity with word embeddings

Text Classification

Problem 1 (15 points): We need functions to assess and evaluate the performance of our models. We will implement those first.
Create one function to calculate precision, one function to calculate recall and one function to calculate f-measure.

Your function should look something like this: get_precision(y_pred, y_true)
Where y_pred is the set of predictions from your classifier and y_true is the set of true (annotated/ground truth/gold) labels.

NOTE: Do not use sklearn built in functions to accomplish this – you are supposed to write your own functions.
For this part of the assignment, you will be using the Large Movie Review Dataset [1] – uploaded to Canvas in assignment 2 folder.

[1] Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011, June). Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1* (pp. 142-150). Association for Computational Linguistics.

It contains movie reviews in two classes – positive and negative. The dataset is split into training and test directories already. Raw text and already processed bag of words formats have also been provided by the authors – c.f. README in the directory.

Problem 2 (10 points): Majority Class Baseline. We will create majority class baseline to evaluate our initial model performance – which is the simplest baseline. The label for the test data should be the majority class found in training data.
You should report P, R and F-score (using the functions you wrote to solve Problem 1) for both training and test data to obtain full marks on this problem.

Problem 3 (25 points): Review Length Baseline. We will create baseline to evaluate our model performance – which takes into account length of the review.

For this baseline, you should try setting various thresholds of review length to classify them as positive or negative. For example, all reviews > 50 words in length can be classified as positive. You should experiment with at least 3 different thresholds and document your reasons why you chose these thresholds.

For each threshold, you should report P, R and F-score (using the functions you wrote to solve Problem 1) for both training and test data to obtain full marks on this problem.

Problem 4 (20 points): Implementing the Naïve Bayes classifier. You can use the built-in Naive Bayes model from sklearn to train a classifier. Here is the documentation for how to implement GaussianNB (discussed in class) using sklearn.
You will train your classifier on the words contained in the positive and negative reviews, based on the algorithm discussed in class.

You should report P, R and F-score (using the functions you wrote to solve Problem 1) for both training and test data to obtain full marks on this problem.

Problem 5 (20 points):** Implementing your own classifier. You can implement your own classifier – by using another classifier from sklearn package – SVM or Logistic Regression (for example) or you can experiment with different features and add them to your Naïve Bayes model. As before, you should report P, R and F-score (using the functions you wrote to solve Problem 1) for both training and test data to obtain full marks on this problem.

Problem 6 (10 points). What do you observe about the performance of the different models?
ITCS 4111 students should report on observations based on their output for Problems 2-4.
ITCS 5111 students should report on observations based on their output for Problems 2-5.

Vector Semantics

Similar to language models, embeddings are trained directly from a large collection of natural language text without being tied to a specific NLP application or subtask. How can we measure the quality of learned embeddings? Some of the commonly accepted extrinsic evaluation methods are based on word similarity tasks, word analogies (e.g., "man is to woman as king is to queen) In this part of the assignment, we will explore an analogy task.
The analogy prediction task is defined as follows. Given a pair of words

<a, b> and a third word c,
choose a fourth-word d
so that the analogy <a is to b> as <c is to d> holds.

NOTE: the system need not characterize this relationship or give it a name! The relationship is taken to be implicit in the pair <a, b>

Mikolov et al. [2] proposed that simple algebraic operations could be applied to embeddings to find an analogy prediction.

[2] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).

Let v_a be the vector for *a*, v_b the vector for *b*, and so on. For the *d* such that the analogy holds, we expect

$$\mathbf{v}_b - \mathbf{v}_a \approx \mathbf{v}_d - \mathbf{v}_c. \quad (1)$$

We therefore seek

$$d = \arg \max_{d \in \mathcal{V} \setminus \{a,b,c\}} \cos\left(\mathbf{v}_d, \mathbf{v}_b - \mathbf{v}_a + \mathbf{v}_c\right). \quad (2)$$

Analogy Dataset Mikolov's analogy dataset [2] includes four semantic relations and four syntactic relations. In the test files, each line represents one analogy question, in the form of four words *<a, b, c, d>*.

For example: "Bangkok Thailand Cairo Egypt"

A question is counted as correctly answered only if the predicted word is the same as the given word. For example, given the first three words "Bangkok Thailand Cairo", the task is to predict "Egypt".

The full set of analogy questions can be found in the file *word-test.v1.txt.*
Available here: http://www.fit.vutbr.cz/~imikolov/rnnlm/word-test.v1.txt

The groups of relations are delimited by lines starting with a colon (:) and you should only work with these groups: capital-world, currency, city-in-state, family, gram1-adjective-to-adverb, gram2-opposite, gram3-comparative, and gram6-nationality-adjective.

Word Embeddings "Pretrained" word embeddings are word embeddings that are already constructed in advance of your NLP project (whether your project is a neural language model, a text classifier, or a class assignment). The advantage of pretraining is that it simplifies learning your model, because the embedding parameters are fixed in advance. The disadvantage is that, if your embeddings happen to be bad for your task, you're stuck with them. For this part of the assignment, you are free to use any pretrained word embeddings you can find, as long as you cite their papers in the writeup.

Here are some popular choices:
• Word2vec: https://code.google.com/archive/p/word2vec/
• GloVe: http://nlp.stanford.edu/projects/glove/
• Dependency-based. embeddings:
https://levyomer.wordpress.com/2014/04/25/dependency-based-word-embeddings/

NOTE: It is okay to use existing tools to efficiently find the most similar vector; as always, cite the tools you used. You may have to settle for lower-dimensional word embeddings or files that contain smaller vocabularies of word embeddings.

Problem 7 (60 points): Run the analogy test. Pick any **two** sets of pretrained embeddings, implement the analogy prediction method described in Equation 2, and compare their accuracies on the eight analogy tasks listed above. Make sure to mention the details of your selection in writing.

Problem 8 (20 points): One known problem with word embeddings is that antonyms (words with meanings considered to be opposites) often have similar embeddings. You can verify this by searching for the top 10 most similar words to a few verbs like *increase* or *enter* that have clear antonyms (e.g., *decrease* and *exit*, respectively) using the cosine similarity. Discuss why embeddings might have this tendency.

Problem 9 (20 points):** Design two new types of analogy tests that are not part of Mikolov's analogy dataset. You will create your own test questions (3 questions for each type, so in total 6 new questions). Report how well the two sets of embeddings perform on your test questions. You're encouraged to be adversarial so that the embeddings might get an accuracy of zero! Discuss any interesting observations you have made in the process.