601.466/666 (Spring 2024)
Assignment 3: Text Classification

Due Date: March 28, 2024 11:59 PM

Introduction

The vector models implemented in Homework 2 are quite versatile and can be used for much more than just retrieving documents. In this homework, you will adapt your models from Homework 2 to handle several standard NLP tasks:

- Word Sense Disambiguation: given a word (e.g. "bank"), what sense of the word is it? (it is the slope by a river, or the financial institution?)
- Named Entity Classification: is a given name a person or a place?
- Spam Detection: is a text spam or ham?

These are significant tasks in information extraction and text understanding, yet they are remarkably similar and can borrow heavily from the vector models implemented in Assignment 2.

Data

You have been provided with several datasets in a tab-separated format, where the columns are *id*, *label*, and *text*.

plant.tsv consists of example sentences of the word *plant* in context, where each instance is labelled as (1) a living biological organism, or (2) a manufacturing facility or factory.

tank.tsv consists of example sentences of the word *tank* in context, each labelled as either (1) a military vehicle, or (2) a container.

Both the *plant* and *tank* datasets are limited to just nouns (i.e. the verb forms *to plant a tree* or *to plant evidence* are excluded. Part of speech taggers are much more appropriate for making plant/verb vs. plant/noun distinctions than vector models are.

perplace.tsv consists of proper names such as *Madison*, *Paris* and *Villahermosa* labelled as either (1) a person, or (2) a place. Distinguishing between these two possibilities is a subset of the larger *named entity classification task*.

The text in these three datasets has already been tokenized. The target word is marked with a `.X-` to distinguish it from the other words, e.g.

```
<S> If anything , he's acting as a wimp , '' .X-Koch said today on `` CBS
This Morning . </s> </S>
```

smsspam.tsv consists of SMS text messages[1], labelled as either (1) spam, or (2) not spam.

WSD and named entity classification are targeted tasks (i.e. you label a single word based on its context). In contrast, in spam detection, your model labels the entire text as spam rather than just a single word. The text in the SMS spam dataset has not been preprocessed. You may consider tokenizing as in Assignment 2.

Part 1 – Vector Classification Model

First, please use the provided `splitdata.py` script to split the data into train and dev splits. You should only train on the training data and test on the dev data (obviously).

You will implement a nearest centroid classifier (also called a Rocchio classifier) using a process similar to what you already did in Assignment 2:

1. Initialize the "document" vectors, where each example sentence is its own document. The weighting options will be discussed below. In addition, store the classification label for later use.
2. Using the vectors in the training set, create two profile vectors $V_{profile_1}$ and $V_{profile_2}$, where $V_{profile_i}$ is the average (or centroid) of all of the training vectors labelled as sense i. The details of this process will be discussed in class.

$$v_{profile_1} = \frac{1}{\|V_1\|} \sum_{v \in V_1} v \qquad \text{where } V_1 = \{v : label(v) = 1\}$$

3. For each vector in the test set, compute its similarity to each profile vector:

```
sim1 = similarity(testvec[i], v_profile1)
sim2 = similarity(testvec[i], v_profile2)
```

   If $sim1 >= sim2$, then label the vector as sense 1, otherwise sense 2. For ease of evaluation, you may also wish to print $sim1 - sim2$ and sort by this value. Large positive numbers will indicate examples that are strongly sense 1, large negative numbers will indicate examples that are strongly sense 2, and values close to 0 are examples that are ambiguous.

4. In Step 3, keep a running count of the total number of the test examples that your program classifies correctly and incorrectly. At the end, print out the percent correct ($\frac{total\_correct}{total\_correct+total\_incorrect}$).

Part 2 – Variations on the Model

Implement and explore the following variations on this basic model:

1. **Position weighting:** The base model assumes that all words surrounding an ambiguous word like *tank* contribute equal weight to its disambiguation, regardless of position. This is clearly not the case. Implement a weighting function sensitive to distance from the ambiguous word (e.g. `.X-tank`) when initializing the training vectors, much like you implemented region weighting in Assignment 2. Specifically, implement

[1] `https://archive.ics.uci.edu/ml/datasets/sms+spam+collection`

(a) smooth exponential distance decay, where a word's weight is `1 / (distance of the word to the target word)`.

(b) stepped weighting, where adjacent words are given weight 6.0, words 2–3 away are given weight 3.0, and all other words are given weight 1.0.

(c) a weighting scheme of your own choice

An example of weights for each of these schemes is shown below:

	as	a	wimp	,	”	.X-Koch	said	today	on	“	CBS
(a) exp	1/5	1/4	1/3	1/2	1		1	1/2	1/3	1/4	1/5
(b) step	1	1	3	3	6		6	3	3	1	1

2. **Special adjacent tokens:** Regardless of the weighting scheme used, the basic vector approach is still a bag of words model. The examples *... pesticide *plant* ...* and *... *plant* pesticide ...* will be represented as the same vector. Options for capturing word sequence, at least locally, will be discussed in class. The simplest case would be to treat the immediately adjacent words (to the left and right) as special tokens:

```
1329  1  FACTORY    statements by the L-pesticide *plant* R-manager ...
1330  2  LIVING    is a very effective L-tropical *plant* R-pesticide ...
1331  1  LIVING   pesticide often found on L-the *plant* R-roots and stems
```

This would allow *pesticide* to contribute to the collective sense profiles for plant in different counters, depending on whether it occurred immediately to the left, right, or in another position. Alternately, those who implementing a bigram model in Assignment 2 could modify this slightly to include only bigrams adjacent to the target word. Details will be discussed in class.

Part 3 – Extensions to the Classification Model

For up to 30 points extra credit, students are *very strongly* encouraged to implement (and comparatively evaluate) one or more of the 4 specified extensions to the vector model for sense disambiguation, including a Naive Bayes classifier, k nearest neighbor classifier, EM classifier for person-place disambiguation, and/or hierarchical clustering of the sense-ambiguity training examples. Details are provided in the hw3 class subdirectory in the `NOTES` file. The total number of points of extra credit will be commensurate with the substance and quality of the implementation. **These optional extensions may be handed in up until March 29th at 3:00 PM EST without penalty**. The goal here is to encourage students to explore interesting variations and novel techniques applied to the broad family of tasks and approaches taught here, and learn what performs well or not based on empirical evaluation. Gaining insight is the key objective and students have an additional 11 days to try out any such optional investigations that they wish.

Evaluation

In order to test the effectiveness of the two variations above, as well as the effectiveness of the use of stemming in this case, you should test your system on the 6 different permutations of parameters, given below. For each permutation, print the accuracy (i.e. % correct) of each permutation on each dataset as shown. Note that the values given below are only samples, and should not be considered to be values that

your system should obtain. For consistency and ease of grading, please use the standard term frequency, cosine similarity, and do not preprocess the text (besides splitting by spaces).

Then, experiment with your different model and preprocessing choices from Assignment 2 and present your best model's performance on these 6 permutations in rows 7-12 of the output table. Please describe the details of these high performing models (term frequency, similarity, preprocessing, etc.) in your writeup.

	Stemming	Position Weighting	Local Collocation Modelling	ACCURACY			
				tank	plant	perplace	smsspam
1	unstemmed	#0-uniform	#1-bag-of-words	.90	.84	.83	.74
2	stemmed	#1-expndecay	#1-bag-of-words	.84	.80	.78	.67
3	unstemmed	#1-expndecay	#1-bag-of-words				
4	unstemmed	#1-expndecay	#2-adjacent-separate-LR				
5	unstemmed	#2-stepped	#1 or #2 (specify)				
6	unstemmed	#3-yours	#1 or #2 (specify)				
7-12	Your best performance on permutations 1-6						
13	Extra Credit: The results of your implemented extension						

Writeup

In a writeup named `results.txt` or `results.pdf`, please include:

1. the results table as described above
2. a short description of your variations in Part 2, extensions in Part 3 (if any), and best models in the Evaluation section
3. a short explanation of the differences of using vector models for targeted tasks (WSD, named entity classification) compared to non-targeted (spam) tasks
4. a description of any non-standard approaches you used or questionable assumptions you made

Submission

Submit a single compressed file via gradescope using the same procedure as on previous assignments.