

600.466 - ASSIGNMENT 3

Lexical Ambiguity Resolution and Text Classification

Due date: Thursday, April 9th, 2020 11:59 PM

Overview

The goal of this assignment is build a general-purpose classifier for resolving various forms of lexical ambiguity and classifying text by such aspects as spam/not-spam, sentiment, gender of author, language (e.g. Danish vs. Swedish), etc. The lexical ambiguity problems include word sense disambiguation and the classification of a proper name as a person, place, etc. Both are significant tasks in information extraction and text understanding, yet they are both remarkably similar and can borrow heavily from the vector models covered in Assignment 2.

Data

You have been provided with 4 data sets. The first consists of example sentences of the word *plant* in context, where each instance is labelled as one of the following senses:

plant/1 = *a manufacturing plant or factory*

plant/2 = *a living plant*

The second data set consists of example sentences of the word *tank* in context, each labelled as either:

tank/1 = *a military vehicle*

tank/2 = *a container*

In both cases, the labelled words 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.

The third data set consists of proper names such as *Madison*, *Paris* and *Villahermosa* labelled as either sense 1=PERSON or 2=PLACE. Distinguishing between these two possibilities is a subset of the larger *named entity classification task*.

The fourth data set is a spam/not-spam classification task. It contains sentences from a variety of emails, labelled as either Spam (1) or Not-Spam (2). This is another classic classification task.

The actual format of the data is nearly identical to the vector representations used in Assignment 2, where each training sentence is equivalent to a “document”. The only differences is that the start of document header (e.g. .I 2407) has a second numeric field (e.g. e.g. .I 2407 1 or .I 2407 2) indicating whether that sentence is an example of sense 1 or 2 of the target word. Also, the target word itself is marked with an .X. For example:

```
.I 2407 1
Abrams
M1A1
.X-tank
divisions
were
...
.I 2408 2
Leaks
in
the
liquid
oxygen
.X-tank
threatened
....
```

By treating each of the example sentences as documents, sense classification can be treated exactly like a document routing problem in the vector model, where tagged training examples are used to create vector “profiles” of each of the target senses.

We provide you tsv’s corresponding to each of the above tasks. Each data file contains 4000 training examples/“documents”. The first 3600 are for training and the last 400 are for testing, as explained below. We provide a simple preprocessing script to do this for you.

Part 1 - Implementing the Vector Classification Model

The basic steps in the classification model are as follows:

1. Initialize the “document” vectors as in Assignment 1, where each example sentence is its own document. The weighting options will be discussed below. When encountering the start of a new example (e.g. `.I 2408 2`), store its assigned sense number in an array for later use (e.g. `$sensenum[2408]=2`).
2. From the first 3600 vectors (the “training” examples), create two new vectors $V_{profile1}$ and $V_{profile2}$, 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.
3. For each of the remaining 400 “test” vectors ($i=3600$ to 3699), compute `$sim1 = &similarity($v_profile1, $vecs[$i])` and `$sim2 = &similarity($v_profile2, $vecs[$i])`. If `$sim1 > sim2` then label test example i as Sense 1, else Sense 2. As output, you should print the horizontal “title” line for the test document (`$titles[$i]`), along with `sim1`, `sim2`, the algorithm’s choice of sense number, the correct sense number, and a `+` or `*` indicating if the algorithm’s choice was correct or not. For ease of evaluation, you may also wish to print `$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) a smooth exponential distance decay, where $vecs[docn]term += 1/(\text{distance_of_term_to_ambiguous_word})$ - i.e. adjacent words have weight 1, words 2 away have weight 1/2, words 3 away have weight 1/3, etc. (2) Implement a stepped weighting function, where adjacent words are given weight 6.0, words 2-3 away are given weight 3.0 and where all other words are given weight 1.0. (3) Implement a weighting scheme of your own choice.
2. **Improvements on the bag of words model:** 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 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  FACTORY      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 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.

3. **Stemming:** Please also investigate the effect of stemming on these tasks.

Please feel free to leverage the starter code given in assignments 1 and 2 to help you complete this assignment.

Part 3 - 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 of each (i.e. % correct) on the *plant*, *tank* and PERSON/PLACE data, as shown. Note that the values given below are only samples, and should not be considered to be values that your system should obtain. Descriptions of the 4 types of position weighting methods and the 2 kinds of proposed local collocation modeling methods are given in Part 2 of the assignment.

	Stemming	Position Weighting	Local Collocation Modelling	ACCURACY		
				tank	plant	pers/place
1	unstemmed	#0-uniform	#1-bag-of-words	.90	.84	.83
2	stemmed	#1-expndecay	#1-bag-of-words	.84	.80	.78
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	The results of your implemented extension					
8	The results of your implemented extension					
9	The results of your implemented extension					

Part 4 - Extensions to the Classification Model

For up to 30 points extra credit, students are *very strongly* encouraged to implement 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.

What to Turn in:

Submission of HW3 is via gradescope, using the same procedure as on previous assignments.