Shreayan Chaudhary
March 28, 2023
Homework #3b Information Retrieval 601.666

The results table:

Id	Stemming	Position Weighting	Local Collocation Modelling	tank	pers/place	plants	smsspam	Distance Metric
1	unstemmed	#0-uniform	#1-bag-of-words	0.6825	0.76	0.7875	0.8136	term freq
2	stemmed	#1-expndecay	#1-bag-of-words	0.7075	0.8225	0.74	--	term freq
3	unstemmed	#1-expndecay	#1-bag-of-words	0.69	0.8225	0.725	--	term freq
4	unstemmed	#1-expndecay	#2-adjacent-separate-LR	0.785	0.7675	0.77	--	term freq
5	unstemmed	#2-stepped	#1-bag-of-words	0.7775	0.77	0.81	--	term freq
6	unstemmed	#3-yours	#1-bag-of-words	0.715	0.8	0.825	--	term freq
7	unstemmed	#0-uniform	#1-bag-of-words	0.895	0.8175	0.9225	0.9552	tfidf
8	stemmed	#1-expndecay	#1-bag-of-words	0.915	0.8425	0.9225	--	tfidf
9	unstemmed	#1-expndecay	#1-bag-of-words	0.925	0.8425	0.92	--	tfidf
10	unstemmed	#1-expndecay	#2-adjacent-separate-LR	0.8625	0.8825	0.795	--	tfidf
11	unstemmed	#2-stepped	#1-bag-of-words	0.9175	0.8625	0.9225	--	tfidf
12	unstemmed	#3-yours	#1-bag-of-words	0.8725	0.8125	0.8375	--	tfidf
13	stemmed	#1-expndecay	#1-bag-of-words bayes	0.9	0.82	0.9	--	term freq

Part 2 Variations:

1. Positional Weighting

   a. Smooth Exponential Decay
   A word's weight is inversely proportional to its absolute distance from the target word.

   b. Stepped Weighting
   Adjacent words are given a high weightage, the words adjacent to the adjacent are given a lower weight, and so on…

   c. Custom Weighting - Exponential decay with a window threshold of 5
   (context size of 5 on left and right)
   I have implemented the exponential decay with a window threshold of 5 (context size of 5 on left and right). It works well to restrict the context size. If we don't restrict the context size, the words that are far away will affect the weight of the current term, no

matter how small it might be, even if it's not relevant to the current word. Hence, it helps to have some kind of context window, outside which the weight of all terms will be 0.

2. Special Adjacent Tokens - Bigram

   I have updated the bigram model used in hw2 to include only the bigrams adjacent to the target word. This will allow the system to capture word sequences. While observing word sequences, word1, word2 will not be treated in the same way as word2, word1.

Vector Model for targeted tasks vs non-targeted tasks

A vector model is a popular approach for NLP tasks that involve representing words as high-dimensional vectors. The vector model can be used for a variety of NLP tasks, including targeted tasks like Word Sense Disambiguation and Named Entity Classification, as well as non-targeted tasks like SMS spam or not-spam classification.

Given below are the differences in the way the vector model is applied to targeted and non-targeted tasks:

1. Targeted Tasks: In targeted tasks like Word Sense Disambiguation (WSD) and Named Entity Classification (NER), the goal is to identify the correct meaning of a word or the type of entity it refers to in a specific context. The vector model is used to capture the distributional properties of words in a given context, which can help disambiguate the correct meaning or entity. For example, if the word "bank" appears in a sentence about finance, the vector model can be used to distinguish it from a sentence about a river bank. In targeted tasks, the vector model is often trained on specialized corpora or datasets that are specific to the task at hand. In every sentence, we have a target word that starts with ".X-" prefix, and the weights of the context words are calculated based on the distance from the target word.

2. Non-Targeted Tasks: In non-targeted tasks like SMS spam or not spam classification, the goal is to classify text messages into different categories based on their content. The vector model is used to capture the distributional properties of words in the entire dataset, which can help distinguish between spam and non-spam messages. In non-targeted tasks, the vector model is often trained on large, general-purpose corpora like Wikipedia or web text. In non-targeted tasks, since we don't have any target words, the entire sentence is labeled with a single label instead of having different labels for each of the words in the sentence.

Overall, the main difference between using the vector model for targeted and non-targeted tasks lies in the way the model is trained and the context in which it is applied.

Part 3 Extensions and Variations:

Simple Bayesian Model For Sense Disambiguation

A Simple Bayesian classifier works on assuming independence of the vectors upon other vectors. This is how I have implemented the Bayes classifier:

1. I have created vectors for all of the 4000 contexts as in the vector model, using simple term frequency (TF) for the vector values, or a region weighting of your choice. I have also excluded the stopwords and stemmed the text.

2. I have created V_sum1 and V_sum2, containing the sum of all the vectors assigned to sense 1 or 2 respectively, identical to the process of computing V_profile1 and V_profile2 in the vector model, except not dividing it by the number of vectors.

3. For all terms in V_sum2, I have computed the LogLikelihood ratio. I have used an epsilon value of 0.2 for the simple smoothing model.

4. I have classified the new test vectors by using the model's LogLikelihoods. For each term in the test vector, the corresponding LogLikelihood value from the model is multiplied by the frequency of the term in the test vector and then sums up these products. This sum of LogLikelihoods is then compared to 0, and based on the sign of sumofLL, the test vector is classified into one of the two classes (class 1 or class 2). If it is exactly zero, both classes are equally likely. The larger the deviation of sumofLL from 0, the higher the confidence in the classification of the test vector. The code then reports the true class and predicted class for each test vector, along with the corresponding sumofLL and the title of the test case.