

Word categories

John Goldsmith

November 24, 2008

1 First week

Write a program to model each word w 's distribution in the corpus by a vector (“ w 's context vector”) of length $2V$, where V is the size of the vocabulary in the corpus; the first V coordinates give the counts of the words found to the immediate left of w , and the last V coordinates give the counts of the words found to the immediate right of w .

For each of the 50 most frequent words in the Brown Corpus, find the 15 most similar words, where the measure of similarity is the cosine of the angle between between the two context vectors. Remember that $v \cdot w = |v||w|\cos \alpha$, where α is the angle between v and w , and that when the vectors are expressed in a coordinate system, $v \cdot w = \sum v_i w_i$.

1.1 Background

We have a corpus C , composed of words, and from it we establish its lexicon, \mathcal{L} , whose size is n . A bag (“multiset”) of words is a set of words, each of them associated with a count (where counts are, as usual, discrete, i.e., integral). We assume that an ordering is available, so that we can refer to the set as $\{w_i\}$. Any such bag of words can be represented as a vector of length n , where the i^{th} coordinate indicates the integer associated with word w_i , and in this case, we interpret that as meaning how “often” that word occurs in the bag.

Let us choose a particular word in \mathcal{L} , say w , which occurs k times in the corpus. We ask what words occur immediately to the right of it, in each of its k appearances, and we make a bag of words from that. The vector representing those right-hand neighbors is w_r ; the vector representing left-hand neighbors, in parallel fashion, is w_l .

Each of these vectors tells a partial story about what syntactic distribution each word occurs in. Two words are similar with respect to their right-hand environment if their right-hand neighbors are similar. Similar in what sense? It must be a sense which normalizes (ignores) a difference in the frequency of the two words. The usual notion of similarity used here is the cosine of the angle between the two vectors—which is the same as the dot product of the two vectors, as long as we have normalized all of our vectors first. Normalization of a vector means finding a vector going in exactly the same direction, but which is of length (norm) 1.0. We find such a vector by calculating the length of the vector (Pythagorean theorem) and dividing each coordinate by that length. Pythagorean theorem: $|v| = \sqrt{\sum_i v_i^2}$.

If we compare words based on their right-hand neighbors, will we get the same result as if we compare the same words on the basis of their left-hand neighbors? Why or why not?

How can we compare words based on both kinds of information? By creating a context vector which is twice as long: of length $2n$, where the first n coordinates represent the left-hand context, and the second n coordinates represent the right-hand context. Then we compute the angle between the context-vector (in *this* sense) between any two words to see how similar their contexts are.

2 Second week

In this second part, the goal is to construct 20 classes of words which are good categories from a distributional point of view.

To make this problem a bit easier, you will only assign the 1,000 words with highest frequency in the Brown Corpus to 20 distinct (non-overlapping) categories.

The first challenge is to find a good method of evaluation your success.

2.1 Evaluation method

The natural way for us to test success is by measuring the amount of mutual information that the analysis captures. Let's call the vocabulary of the Brown Corpus \mathcal{B} , and its length is the token count of \mathcal{B} , that is $[\mathcal{B}]$.

We will call the 1,000 words of highest frequency in the corpus the set \mathcal{W} . The token count of \mathcal{W} is $[\mathcal{W}]$. The bigrams of the Brown Corpus is the multiset of all pairs of words (w_i, w_j) that occur in that order in the Brown Corpus. Each such pair is associated with a total mutual information: $[(w_i, w_j)] \log \frac{\text{freq}(w_i, w_j)}{\text{freq}(w_i)\text{freq}(w_j)}$. As ever, $\text{freq}(w_i) = \frac{[w_i]}{[\mathcal{B}]}$.

Step 1. Calculate this amount for just the words in \mathcal{W} (not the whole set \mathcal{B}) – it is the maximum amount of mutual information that we could extract from just these words in this corpus. In this step, you use frequency exactly as defined just above.

Step 2. Let's consider a partition \mathcal{C} of \mathcal{W} , i.e., a set of categories. \mathcal{C} is composed of 20 disjoint subsets, C_i . Calculate the total mutual information:

$$\sum_{i,j=1}^{20} [C_i] \log_2 \frac{[\mathcal{B}][C_i, C_j]}{[C_i][C_j]} \quad (1)$$

where $[C_i, C_j]$ is the count of the number of bigrams uv in the corpus where $u \in C_i$ and $v \in C_j$.

This is our figure of merit, our way of evaluating, \mathcal{C} .

2.2 Discovery process

The following method is based on Chris Biemann's "Chinese whispers" algorithm.

Construct a graph G in which each node (vertex) is a category, initially identified uniquely with one of the words of \mathcal{W} . The weight of the edge between any two nodes is equal to the number of left-neighbors and right-neighbors

that the two categories have in common (initially, that the two *words* have in common).

Start: $i = 1000$.

Pick a node at random from among the i nodes; call it v . Find the node w that is adjacent to v for which the edge weight of (v, w) is the greatest. Collapse the words associated with w and v into a single category, and recalculate the edge weights of all nodes that were adjacent to either w or v .

Decrement i , and stop when $i = 20$; otherwise repeat preceding step.

Run this algorithm several times, and determine how similar the results are (quantitatively), and do your best to answer two questions:

1. How similar are the categories that the algorithm finds each time?
2. How similar are the categories to those taught in school?
3. How satisfied are you with this method?