CS114 (Spring 2016) Homework 5
Distributional Semantics Takes the SAT

Due May 3, 2016

Introduction

Distributional semantics is one of the modern methods in natural language processing in solving tasks that require semantic knowledge of a word or relations between words. In this homework, we will learn how to create semantic representation of a word from a corpus and how to use it in computational lexical semantic tasks.

1 Create distributional semantic word vectors

Your task is to create distributional semantic word vectors from a small artificial corpus. We need numpy and scipy libraries for this part of the homework. They should have been installed already if you have nltk. This part of the homework should require around 30 lines of code. We break the process down into steps below. For each step, also answer any questions given.

- Compute the co-occurrence matrix. The number of occurrences of word w and context word c = the number of (c,w) and (w,c) bigram in the corpus.

  You should initialize the matrix by
  `C = numpy.zeros((vocab_size, num_context_word_type))`

  `C[i,j] = the number of occurrences of word i and context word j`

- Multiply your entire matrix by 10 (to pretend that we see these sentences 10 times) and then smooth the counts by adding 1 to all cells.

- Compute the positive pointwise mutual information for each word w and context word c
  - `PPMI(w, c) = max(`$\log_2$`(P(w_and_c_cooccur)/(P(w)*P(c)), 0)`

  Note that these probabilities can be computed directly from your count matrix.

- Reweight your count matrix with the PPMI by multiplying them element-wise. Then compare the word vector for "dog" before and after PPMI reweighting. Does PPMI do the right thing to the count matrix? Why? Explain in a few sentences how PPMI helps (short prose will do here; no need to show any math, rather just an intuitive understanding).

- At this point, we have a functional distributional semantic model. Let's try to find similar word pairs. Compute the Euclidean distance between the following pairs (you can use the command `scipy.linalg.norm` to compute the length/norm of a vector):
  - women and men (human noun vs. human noun)
  - women and dogs (human noun vs. animal noun)
  - men and dogs (human noun vs. animal noun)
  - feed and like (human verb vs. human verb)
  - feed and bite (human verb vs. animal verb)
  - like and bite (human verb vs. animal verb)

- Do the distances you compute above confirm our intuition from distributional semantics (i.e. similar words appear in similar contexts)?

- Decompose the matrix using SVD by using the command `scipy.linalg.svd`, using the commands given below. Then verify that you can recover the original matrix by multiplying U, E, and the conjugate transpose of V together.

```
U, E, V = scipy.linalg.svd(ppmi_weighted_count_matrix,
full_matrices = False)
E = numpy.matrix(numpy.diag(E)) # compute E
U = numpy.matrix(U) # compute U
V = numpy.matrix(V) # compute V
Vt = numpy.transpose(V) # compute conjugate transpose of V
```

- Reduce the dimensions to 3 to get word vectors by using the command below. Now your word vectors are in the abstract semantic space.

  ```
  reduced_ppmi_weight_count_matrix = ppmi_weight_count_matrix *
  V[:,0:3]
  ```

- Compute the Euclidean distances of human/animal nouns/verbs again but on the reduced PPMI-weighted count matrix. Does the compact/reduced matrix still keep the information we need for each word vector? (In practice, we reduce the PPMI-weighted count matrix from 300,000 into 500, which is a huge save.)

2 Computing with semantic vectors (word vectors)

You are given two types of word vectors.

1. Classic distributional semantic matrix (like in part I). The matrix is trained using a 2-word context window, PMI weighting, and SVD reduction to 500 dimensions. The co-occurrence statistics are computed from the British National Corpus and the Web-As-Corpus. It is trained by the COMPOSES toolkit. You could in fact make your own using the pipeline from part I, but taking SVD takes hours in a realistic setting—cubic time ($O(N^3)$) on the number of dimensions.
2. Google's souped-up hyped-up 300-dimensional word vectors trained by deep learning. This new algorithm irritates a few distributional semanticists. You dont have to know how this works, but it is a new paradigm in deriving word vectors. The version we provide is trained on Google News corpus and taken from the word2vec toolkit. We will see if their word vectors live up to their hype and how they fare against the more classical method.

We are putting these word vectors into semantic tests. Note that we are using an unsupervised learning technique here, so the dataset is not split into train/dev/test.

1. Synonym detection. You are given ∼900 pairs of synonymous verbs in the dataset. We want to use word vector to detect synonyms. Here is what we need to do:

(a) Create a multiple-choice question test set: For each verb, you will pick 1 of the synonyms and 4 random non-synonyms from the dataset. Make 1,000 of such synonym multiple-choice questions.

(b) Your task is to use the word vectors to pick out the synonym out of the 5 alternatives for each verb and compute the accuracy. Try using Euclidean distance and cosine similarity. Make a table to compare the results from those methods.

(c) SAT Analogy questions. You are given 374 SAT analogy questions. I am sure most of you did well on the SAT in the past, but apparently they've gotten rid of the analogy portion now that computers can do it. Shame.

(For those of you who did not go to an American college, the SAT test is a national standardized test required by most American colleges as part of the application package. One of the sections ~~is~~ used to be on lexical analogy. For example,

MEDICINE : ILLNESS ::

(a) law : anarchy

(b) hunger : thirst

(c) etiquette : discipline

(d) love : treason

(e) stimulant : sensitivity

MEDICINE is used to combat ILLNESS. Law is used to combat anarchy. (Hunger is not used to combat thirst, etiquette is not used to combat discipline, etc.) So (a) is the correct answer.)

Your task is to use the word vectors to answer these questions correctly. Feel free to exclude out-of-vocab words. Submit a write-up describing what you have tried and, if you are successful at generating answers, make a table to compare the results from those methods. Unlike the synonym task, you have to be a bit creative with how you use the word vectors and you should try some different tactics at generating them (at least 2 strategies for full credit—*describe these in your write up!*).

Think about where the word vectors in the question and the choices stay in the abstract semantic space. Think about how to obtain a relation vector between the two words (e.g. subtracting the two word vectors? adding? multiplying? concatenating?).

> An average SAT taker got around 57% accuracy on the analogy section (ouch, maybe that's why they got rid of it), and the state-of-the-art system is almost as good. Random guesses get 20% accuracy. You should aim for about 30% accuracy, so significantly better than random guesses.

Data and Resources

The datasets are quite large so they are put on Brandeis CS storage. You can access it directly from the terminal when you login to one of the CS department machines.

If you want to avoid working on the remote machine, use the UNIX command to copy the files over to your own local machine. Here's the sample usage of the command scp:

```
scp yourbrandeiscsid@diadem.cs.brandeis.edu:path/to/source/file
path/to/your/machine
```

Artificial corpus for the first part
`/home/j/llc/tet/nlp/corpus/dist_sim_data.txt`

Synonym detection data
`/home/j/llc/tet/nlp/corpus/EN_syn_verb.txt`

SAT questions
`/home/j/llc/tet/nlp/corpus/SAT-package-V3.txt`

Googles word2vec word vectors (zipped 100MB; unzipped 400MB)
`/home/j/llc/tet/nlp/lib/lexicon/google_word_vector/GoogleNews-vectors-rcv_vocab.txt.tar.gz /home/j/llc/tet/nlp/lib/lexicon/google_word_vector/GoogleNews-vectors-rcv_vocab.txt`

COMPOSES's word vectors (zipped 300MB; unzipped 700 MB)
`/home/j/llc/tet/nlp/lib/lexicon/composes_word_vector/EN-wform.w.2.ppmi.svd.500.rcv_vocab.txt.tar.gz /home/j/llc/tet/nlp/lib/lexicon/composes_word_vector/EN-wform.w.2.ppmi.svd.500.rcv_vocab.txt`

Submission

Submit:

- Any code you write in the course of this assignment
- A write-up containing the answers to the questions in section 1, including the vector distances between the word pairs given, once normally and once using the reduced PPMI-weighted count matrix.
- Your synonym test set from section 2. We won't be looking at this too closely but you should submit it so that we know you made one.
- A table comparing the synonym test results using Euclidean distance vs. cosine similarity.
- A write-up discussing methods you used for creating word vectors for the analogy task, and a table showing your accuracy results for each vector-creation method (overall, not broken down for each question).

Grading

Grades will be determined as follows:

- 20% for code submission.
- 20% for section 1 write-up that discusses all points listed above. (14 points on questions and discussion, 6 points on word-pair calculations) .5 points will be deducted for each missing word-pair. Remember, you should have 2 sets of vector distance calculations.
- 10% for synonym test set submission.
- 20% for synonym test results. (10 points for Euclidean distance results, 10 points for cosine similarity results).
- 30% for analogy task discussion and results (20 points for discussion of the methods you used to create word vectors, 10 points for accuracy tables).
- Within these parameters, partial credit will be assigned where possible. You may notice that the writing and discussion is weighted higher than the actual results. This is not an accident. The results are important but demonstrating your understanding of why things worked

(or didn't) is more so (think about it: someone else has already built working distributional semantic algorithms—our job is to prepare you to think in new directions about what you can do with them). For that reason, mediocre results and good discussion will net you more credit than great results without a good discussion. When in doubt, write up the issues you're having and some speculation as to why.

TL;DR: *Please do not skimp on the discussion!*