CS542 (Fall 2021) Programming Assignment 4
Distributional Semantics Takes the SAT

Due November 19, 2021

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 representations of words from a corpus and how to use them in a computational lexical semantic task.

1 Create distributional semantic word vectors

Your task is to create distributional semantic word vectors from a small artificial corpus (see Data and Resources section below). We will need to use Numpy and Scipy for this part of the homework. You break the process down into steps below. Part of this was already demonstrated at the end of Lecture 22 (minus the inclusion of the word "the"), and you are free to use or adapt this code. For each step, also answer any questions given.

- Compute the co-occurrence matrix (read: Numpy array) C, such that:

  $\texttt{C}[\texttt{w}, \texttt{c}]$ = the number of (w, c) **and** (c, w) bigrams in the corpus. Each row in the array should represent the number of co-occurrences of w with each c.

- 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 (PPMI) for each word w and context word c:

$$\texttt{PPMI}(\texttt{w}, \texttt{c}) = \max\left(\log\left(\frac{P(w,c)}{P(w)P(c)}\right), 0\right)$$

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

- Now instead of using your (original) count matrix, use the PPMI matrix as a weighted count matrix. Then compare the word vector for "dogs" before and after PPMI reweighting. **What happened to the elements of this vector that had a smoothed count of 1 in the smoothed count matrix? What happened to the elements of this vector that had the same value (not 1) in the smoothed count matrix?** You can look at the indices of these numbers in the word vector and determine what context word c they correspond to. **Does it seem like PPMI reweighting did the right thing with regard to the representation of each context word c in the word vector for "dogs"? 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* (not cosine as was done in Lecture 22) 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 singular-value decomposition (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 `Vt` together.

```
U, E, Vt = scipy.linalg.svd(PPMI, full_matrices=False)
U = np.matrix(U) # compute U
```

```
E = np.matrix(np.diag(E)) # compute E
Vt = np.matrix(Vt) # compute Vt = conjugate transpose of V
V = Vt.T # compute V = conjugate transpose of Vt
```

- 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 = PPMI * V[:, 0:3]
  ```

- Compute the Euclidean distances of the 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?

2 Computing with distributional semantic word vectors

You are given two types of word vectors.

1. Classic distributional semantic matrix (like in Section 1). The matrix is trained using a 2-word context window, PPMI 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 Section 1, but SVD takes hours in a realistic setting—cubic time ($O(n^3)$) in the number of dimensions.

2. Google's 300-dimensional word vectors trained by deep learning. The version we provide is trained on the Google News corpus and taken from the word2vec toolkit. We will see how their word vectors fare against the more classical method.

We will use these word vectors in to perform a version of the SAT analogy test. Note that we are performing an unsupervised task here, so the dataset is not split into train/dev/test.

You are given 374 SAT analogy questions. I am sure most of you did or would have done well on the SAT in the past, but apparently they've gotten rid of the analogy portion now that computers can do it.

(For those of you who did not go to an American college, the SAT is a national standardized test required by many American colleges as part of

the application package. One of the sections 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 to* illness *as* X *is to* Y," and your job is to pick the pair of X and Y that best answers the question.

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.)

The SAT questions in the provided dataset are formatted like this:

```
190 FROM REAL SATs
word language n:n
paint portrait n:n
poetry rhythm n:n
note music n:n
tale story n:n
week year n:n
c
```

Each question is a "paragraph" in the document, with a blank line before and after. Once you have separated each individual question, you may ignore the first line, e.g., `190 FROM REAL SATs`. The first line after that is the prompt. E.g., this question prompt would be `WORD : LANGUAGE ::`. The following five lines are the answer options, and the final line is the answer. Here, it is C, so `WORD : LANGUAGE :: NOTE : MUSIC`. (A word is a unit of language, a note is a unit of music.) You may find it useful to convert the answer into a numerical index. The final notation on the lines (e.g., `n:n`) denotes the part of speech of the two words on that line, e.g., "word" and "language" are both nouns. This information may or may not be useful to you in this task, depending on the method you use. If you do not make use of this information, you may discard it during preprocessing.

Your task is to use the word vectors to answer these questions correctly. Any sensible method of handling unknown words is fine. Submit a write-up describing what you have tried and your results.

You have to be a bit creative with how you use the word vectors. 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?). **How do the COMPOSES word vectors perform compared to the Google word vectors?**

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

You are given the following data sets, which are uploaded to Canvas:

1. Artificial corpus for the first part: `dist_sim_data.txt`
2. SAT questions: `SAT-package-V3.txt`
3. Google's word2vec word vectors (zipped 100 MB; unzipped 400 MB):
   `GoogleNews-vectors-rcv_vocab.txt.tar.gz`
4. COMPOSES' word vectors (zipped 300 MB; unzipped 700 MB):
   `EN-wform.w.2.ppmi.svd.500.rcv_vocab.txt.tar.gz`

Submission

Submit in a single zip file:

- Any code you write in the course of this assignment.
- A write-up (in PDF format) 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.
  - A write-up discussing methods you used for creating word vectors for the analogy task, and your accuracy results.

Grading

Grades will be determined as follows:

- 20% for code submission. `.py` files or a single Jupyter notebook is acceptable for this assignment.
- 30% for Section 1 write-up that discusses all points listed above. (18 points on questions and discussion, 12 points on word-pair calculations) 1 point will be deducted for each missing word-pair. Remember, you should have 2 sets of vector distance calculations.
- 50% for analogy task discussion and results.
- 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: **You are free from the tyranny of the autograder on this assignment. Please do not skimp on the discussion!**