ANLP Assignment 3 2017: Exploring distributional similarity in Twitter

Sharon Goldwater

This assignment is available as a web page[1] or pdf document[2].

Overview, goals and motivation

This assignment will give you an opportunity to explore methods from distributional semantics using the same Twitter dataset as in Lab 8 (Sentiment Analysis), but going into a bit more depth. In addition, it is intended to start bridging the gap between the kind of fully specified tasks that are typical of many course assignments, and the kind of open-ended work that is typical of research projects and real-world tasks.

In most real projects, some initial exploration is required to start to understand the data or investigate different methods. This assignment focuses on the kinds of tasks that might be involved in this initial exploration step. The high-level question you need to address is: what are (some of) the pros and cons of different methods for computing similarity between words in the Twitter dataset?

We'll give you some general guidance for initial steps of what we'd like you to do, but there will also be considerable scope for you to decide the details of your analysis.

Your mark will be based on a short report you will write, which should describe what questions you decided to investigate, what approach you used, and what results you got.

Submitting your assignment

- **Due date:** Monday 27 November, 3pm, by electronic submission.
- **Format:** Please submit a single report as a pdf file (one per pair of students) along with any python code you wrote. Include both student's ID numbers (s...) at the top of the report. Please do not include your names, only ID numbers.
- **Submission:** Submit your assignment using the following command on a DICE machine:

  ```
  submit anlp cw3 report.pdf <file2.py> ... <fileN.py>
  ```

  where `report.pdf` is your report and the remaining files are your code.

Logistics of multiple submission and late submission are the same as in previous assignments.

Good scholarly practice

As with other assignments in this course, this assignment is inteneded to be done in pairs. You may freely discuss and share work within your pair.

For other policies, please remember the University requirements regarding all assessed work for credit. Details and advice about this can be found at

http://web.inf.ed.ac.uk/infweb/admin/policies/academic-misconduct

[1]http://www.inf.ed.ac.uk/teaching/courses/anlp/hw/assignment3.html

[2]http://www.inf.ed.ac.uk/teaching/courses/anlp/hw/2017/assignment3.pdf

[3]http://www.inf.ed.ac.uk/teaching/courses/anlp/hw/2017/code/load_map.py

[4]http://www.inf.ed.ac.uk/teaching/courses/anlp/hw/2017/code/asgn3.py

Support code

We have provided two files with this assignment: load_map.py[3], which is the same file used in lab 8, and asgn3.py[4].

This assignment assumes you have already worked through Lab 8 (Sentiment Analysis), which explains the format and contents of the data files. If you have *not* done Lab 8, please do so before continuing.

You can run the current (skeleton) version of the code as follows from the iPython console in Spyder:

```
%run load_map.py # You only need to run this once when you first start ipython.
%run -i asgn3.py # The -i is so the word id mappings are passed to this script.
```

The code will print out a list of word pairs with their similarities (currently all zero, you will need to fill in code to compute these correctly) and the frequency of each word.

Before doing anything else, you should familiarize yourself with the support code provided. Roughly speaking, the code defines a list of words, computes a context vector for each word (using PMI scores), and then computes the similarity between each pair of word vectors.

Notice that the word list gives each word in its standard form (e.g., `computer`, `mouse`), but the data set we are using only includes stemmed versions of each word (e.g., `comput`, `mous`). So we added a function to the support code that uses the Porter stemmer provided in NLTK to stem each word before computing its context vector. Tokens that start with `@` (usernames) are not stemmed by our function because they have special meaning in Twitter. (It would probably be good not to stem words starting with `#` either, because they indicate discussion topics, but I wanted to be consistent with the preprocessing of the dataset, which *does* stem these. I didn't notice that until after releasing the assignment and it's too late to change it.)

Preliminary task

Before implementing anything else, complete the functions needed to print out the correct cosine similarity for each of the word pairs in the support code. That is, fill in the missing parts of the following functions (we'll do the first one during Lab 8, you can copy the solution from there):

```
PMI
cos_sim
create_ppmi_vectors
```

You should use PPMI (positive PMI) to create the vectors, and you should use a *sparse vector* representation. That is, rather than storing a huge context vector for each word that mostly consists of zeros, you should use a representation that only stores the non-zero entries of each vector. We recommend doing so using a dictionary. For example, the vector `[1 0 0 1 0 2 0 0 0]` would be stored as `{0:1, 3:1, 5:2}`. (This representation is used in the `counts` file and the `co_counts` data structure already, so it should be easy for you to do.)

This sparse representation saves a lot of computer memory and can be much faster to compute with than full vectors. Consider that for our vocabulary of about 200k words, a full 200k x 200k matrix of cooccurrence counts would contain 40b integers at 1 byte each, taking 40GB of memory!

After you complete this task, save the output (the ranked list with correct similarity scores) to include in your report.

Main task

Your main task is much more open-ended: essentially, we would like you to explore at least *two* other methods for computing similarities. You could choose to look at different methods of computing the context vectors, different similarity measures, or a combination of each.

After implementing your chosen methods, you will need to:

1. Choose a set of words/word pairs with which you will evaluate your results.
2. Run your methods on these word pairs and analyze the results, using both quantitative and qualitative methods.
3. Write a brief summary of your analysis.

You may choose whichever words you like and examine whatever questions you like in your analysis. Below are some suggestions for how to get started and what kinds of questions you might want to consider.

Choosing words

- You will probably want to include words that you would expect to cover a range of similarities---that is, words where (intuitively) some pairs are very similar, some are moderately similar, and some are not very similar. You can choose words by hand, using online lists or other lexical resources, or in some way automatically (though in the latter case it may be harder to ensure you have some highly similar pairs). You will need to decide how many words to include in your analysis: a smaller list will be easier to look at specific examples, while a larger list may give more representative results if you do an automated analysis.
- You will probably also want to include words from a range of frequencies. If you are having trouble coming up with words of low frequency, do consider alternate spellings as mentioned below. You can also sort the words by frequency and take a look at parts of the list for ideas.
- If you're choosing words by hand, one way to do so is to think of some category of words (like *foods* or *colors*) and add words from that category and perhaps from related categories. You also may want to consider particular relationships between words, like synonyms or antonyms or hyponyms.
- Don't forget that in addition to dictionary words, you can include usernames (`@` words) or hashtags in your analysis if you want. You could also consider alternate spellings, which are typically used for emphasis (like `loove` for `love`) or to more closely reflect casual pronunciation (like `goin` for `going`).
- Remember that stopwords have been removed from the dataset (see list of stopwords), so you can't include those in your analysis. If you want to see the list of removed stopwords, you can find it at http://www.inf.ed.ac.uk/teaching/courses/anlp/hw/stopwords.txt.
- Also remember the data is from 2011! More recent names/events aren't reflected in it.

Your choice of words to test may be closely tied to the particular question(s) you address. So you may want to look at a few different possibilities before settling on what makes sense and will allow you to address the question(s) you're interested in (see below).

Choosing methods

The textbook and slides mention many ways to compute context vectors and similarities. Just pick something you'd like to try. In a few cases, potential pros/cons are already mentioned, so these might be things to try evaluating yourself in your analysis.

If you decide to change the context vectors, you may wish to replace PPMI with some other statistic, or you could investigate other changes, like changing the number of context words used, or reducing the dimensionality of the vectors. Replacing PPMI is the easiest option and is totally fine. The other options may take considerably more work and should not be attempted unless you really want a challenge and/or have previous experience with them.

Due to the limitations of the preprocessed data set, some methods of computing vectors (e.g., `word2vec`) will not be possible.

You are not required to use or build on our support code if you decide to do something rather different and feel it's easier to rewrite from scratch.

What to look for

Some possible questions to consider might be:

- How are the different methods affected by word frequency? Are some methods more sensitive to frequency, or especially good/bad for low-frequency words?
- Do you see any systematic differences in how different methods rank the similarities between words? Do some methods seem to be more accurate than others?
- Do you see any results (either overall, or for particular words/word pairs) that surprise you? If so, are these surprising results consistent across different measures, or do they differ? By looking more closely at specific examples, can you find an explanation of the result?

Or think of your own question!

You do *not* need to answer all of these questions in your report. Rather, you should choose one or two questions to focus on, where you feel you have something interesting to say. Identify which question(s) you are addressing, how you addressed them, and what results you got.

How to approach the question(s)

Ideally, you should use a combination of *quantitative* and *qualitative* analysis. In NLP, the most common form of quantitative analysis is comparing the accuracy of a result against some gold standard. However, there is no gold standard provided here. So your quantitative analysis will need to be something different. For example, you might decide to compute the correlation between the similarity score of each word pair and the frequency of that pair or the words in it.

There are many more options for qualitative analysis. You could decide to visualize some results, discuss the behaviour of some particular examples, or analyse just one surprising pair in more detail by using the Twitter search page to get a better idea of what might be going on.

As you work on the assignment, you may indeed do several of these, but again, please do *not* try to describe of everything you did in your report. Focus on one or two analyses that seem the most interesting or illuminating, and give a clear and concise explanation of those.

What to write up

This assignment is more open-ended than the previous ones, but we are strictly limiting the length of your report (see below). So you must limit the scope of your investigation---we are not looking for you to try every possible thing you can think of or provide completely definitive answers to all questions. We are basically looking for you to implement some fairly simple methods and spend a bit of time thinking about what is in the dataset and see what you can discover about the data or the methods.

As mentioned above, your report should focus on just one or two questions that you investigated. You are more likely to do well by providing a thoughtful analysis of a small number of methods than by attempting to compare a very large number of methods (because you probably won't have time or space to do a good job of that). Your report should describe:

- what words you tested, and how you decided on those words.
- what question(s) you investigated.
- what methods you compared. You do not need to describe (P)PMI or cosine similarity, but you *should* include a brief description of whatever other methods you chose.
- how you analysed the results and what you concluded.

In addition you should include:

- the output of the program after you have completed the Preliminary Task (i.e., the ranked list of pairs we defined with their correct cosine similarities). This can be included as the first part of your "additional two pages" (see below). You do not need to comment on it or relate it to the rest of your report, it's just for us to check the numbers.

Note well: Your report is limited to **two pages of text** (no smaller than 11 point font) plus **two pages of additional material** which can include figures, tables, word lists, example Tweets, bibliography, etc. We may deduct marks for reports that go over these limits.

These limits are very short! Writing a short document well is often harder than writing a long one. Don't wait until the last minute to start writing. You will need time to decide what is the most important information to include and how to edit your text to the right length. If your list of words is very long, you will need to summarize it in some way. All figures must be properly labelled and large enough to be readable without magnifying.

Marking

Marking will be based on the quality of your report. In most cases, we will not even look at your code, although we may do so if the report indicates possible problems with the implementation. We may also do spot checks to ensure that your code matches what you say in the report.