School of Computing and Information Systems
The University of Melbourne
COMP90042
WEB SEARCH AND TEXT ANALYSIS (Semester 1, 2019)

Workshop exercises: Week 6

Discussion

1. Give illustrative examples that show the difference between:
   (a) **Synonyms** and **hypernyms**
   (b) **Hyponyms** and **meronyms**
2. Using some Wordnet visualisation tool, for example,
   `http://wordnetweb.princeton.edu/perl/webwn` and the Wu & Palmer definition of **word similarity**, check whether the word *information* is more similar to the word *retrieval* or the word *science* (choose the sense which minimises the distance). Does this mesh with your intuition?
3. What is **word sense disambiguation**?
4. For the following term co-occurrence matrix (suitably interpreted):

	`cup`	`not(cup)`
`world`	55	225
`not(world)`	315	1405

   (a) Find the Point-wise Mutual Information (PMI) between these two terms in this collection.
   (b) What does the value from (a) tell us about **distributional similarity**?
5. In the `WSTA_N10_distributional_semantics` iPython notebook, a document–term matrix is built for the purposes of IR-style document retrieval.
   (a) What is the Singular Value Decomposition (SVD) method used for here? Why is this helpful?
   (b) What is the significance of the `transform_query()` function?
6. What is a **word embedding** and how does it relate to **distributional similarity**?
   (a) What is the difference between a **skip-gram** model and a **CBOW** model?
   (b) How are the above models trained?

Programming

1. Consider the iPython notebook `WSTA_N9_lexical_semantics`. Repeat the exercise about word similarity from the Discussion problems above; confirm that you get the same answer. Now try the Lin similarity — do you get the same result? Why or why not?
2. Use the `WSTA_N10_distributional_semantics` iPython notebook to find some interesting collocations, using PMI.

(a) Write a wrapper function which finds the 10 collocations with the greatest PMI, amongst all of bi-grams in the collection. (Note that you might want to be careful about your strategy for doing this in a very large collection!)

(b) NLTK has an in-built method `collocations()` (of a `Text` object) — does it come up with the same collocations as PMI? Why do you think this is the case?

Catch-up

- What is **information**, with respect to **entropy**? How might we calculate the information of a word in a corpus? How might we calculate the information of a (textual) message with respect to the information in a corpus? (There are many different ways!)
- What is **WordNet**? What is a **synset**?
- What is the **cosine similarity** and how is it calculated?
- What is **entropy** and how is it calculated? What is entropy attempting to measure?
- What is a **term–document matrix**? How is it different to an **inverted index**?
- What is a TF-IDF model? What are its intuitions and how do they appear in a typical model (formula)?

Get ahead

- Choose an individual word and consider its different synsets in Wordnet.
  - Find a number of instances of that word in a corpus. Assign each token to its sense. Is one sense more frequent than the other senses?
- Build a system which attempts to use the **Lesk** strategy of WSD based on the Wordnet bindings in NLTK. Choose some word(s) in some sentence(s) and observe its output — does the correct sense get returned? Why or why not?
- In the notebook `WSTA_N10_distributional_semantics:`
  - Try doing the same calculations on the collection without the SVD method. How much time does the truncation step save?
  - Try different values for truncating the decomposition; at what point do the results seem to become noticeably worse?
  - How important is the TF-IDF step? Try the retrieval without it; do the results change? What if you omit it from only the document matrix, or only the query vector?
  - Try to find some queries where the results are different with and without the TF-IDF transformation.