Hot topics in the distributional semantics world

Some aspects of meaning are problematic

- Detecting **hyponyms**, **hypernyms** and **antonyms**: they appear in **similar contexts**, but...
- cannot be replaced by each other:
- their **paradigmatic relations** are complex.
- Solutions:
  - integrating **lexical contrast** [Nguyen et al., 2016]
  - integrating **syntactic paths** [Shwartz et al., 2016]
  - etc.

Some aspects of meaning are problematic

- Distributional models are not aware of **implicit knowledge**:
  - sky is blue
  - bananas are yellow
  - violins are brown.
- The answer is ‘**grounding**’:
- integrate **language** and **vision**.
- Aligning image embeddings with word embeddings.
There is more than one language in the world

- Can we train bilingual or multilingual distributional models?
- We can!
- Lots of approaches emerged in the last 3 or 4 years.
- Thorough review of cross-lingual word embeddings in [Upadhyay et al., 2016]

How can we evaluate our models better?
Generate new and more natural gold standard datasets! Perhaps, using crowd-sourcing and gamification.

A cute, hairy wampumuk is sitting on the hands.
Hot topics in the distributional semantics world

http://comp3096.herokuapp.com/
[Parasca et al., 2016]

Table 1: Successful game in 2 rounds for banana

<table>
<thead>
<tr>
<th>Round</th>
<th>Narrator’s clue</th>
<th>Guesser 1</th>
<th>Guesser 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a</td>
<td>fruit</td>
<td>orange</td>
<td>apple</td>
</tr>
<tr>
<td>1b</td>
<td>yellow</td>
<td>lemon</td>
<td><strong>banana</strong></td>
</tr>
</tbody>
</table>

Table 2: Unsuccessful try (3 trials, weather)

<table>
<thead>
<tr>
<th>Round</th>
<th>Narrator’s clue</th>
<th>Guesser 1</th>
<th>Guesser 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a</td>
<td>rain</td>
<td>sun</td>
<td>jacket</td>
</tr>
<tr>
<td>2a</td>
<td>sunny</td>
<td>cloudy</td>
<td>windy</td>
</tr>
<tr>
<td>3b</td>
<td>cloud</td>
<td>umbrella</td>
<td></td>
</tr>
</tbody>
</table>

[Parasca et al., 2016]
Discussion of the obligatory assignment

- Good news: everyone has passed :-)
- What was interesting?
- Won’t comment on purely pythonic issues, read the feedback.

- No need to include large data files in your submission
- Task 1: what is missing in Semantic Vectors web service?
- Some pointed they miss vector algebra (addition and subtraction)
- It’s already there: see the Calculator tab (http://ltr.uio.no/semvec/en/calculator)

- Task 2 (evaluation)
- Very frequent issue:
  - while calculating SimLex999 correlation, you ignore (skip) out-of-vocabulary words
  - Seems logical, but can be dangerous:
    - imagine the model doesn’t know 95% of the words from the dataset but is good in ranking the remaining 5%
  - Can we say this model is perfect?
  - Might be safer to produce similarity=0 for such word pairs (pretend the model thinks they are not related).

A good point: values of performance in Google Analogy test and in SimLex999 test are not directly comparable (64 > 34 means nothing).
Task 3 (document classification)
Everyone used semantic fingerprints (as expected).
*Gensim* model vector size can be retrieved with `model.vector_size`;
Word vectors are *Numpy* arrays;
Work with them using *Numpy* functions;
Try not to mix with other data types.

If you iteratively update your document vector (fingerprint):
create it as a *Numpy* array from the very beginning:
`numpy.zeros(model.vector_size)`
then successively add word vectors to this array.
Another way: first generate a zero matrix *(words number X vector size)*;
successively fill in the rows with word vectors;
Then do `numpy.sum()` by axis 0 and `numpy.average()`;
NB: do not try to expand the matrix (add new rows with new words)!
Array expansion is comparatively slow in *Numpy*.

Interesting issue with initialization, leading to *Infs*
You have a new document, you initialize the empty fingerprint variable with the vector of the first word:
`fingerprint = model[first_word]`
and continue updating it with the vectors of the next words
*Gensim* model is like a *Python* dictionary
`fingerprint` is linked to the same memory location as the word embedding in the model!
They essentially become one.
Thus, word embedding in the model (say, ‘today’) is summed up with the next vectors.

After some time, the same word occurs in the text.
Its vector is added to itself and is doubled!
`fingerprint` values grow fast and quickly reach *Inf*;
the model in RAM is corrupted;
things go crazy.

Remedy:
`fingerprint = numpy.zeros(model.vector_size)`
fingerprint += model[first_word]
fingerprint += model[second_word]
...
fingerprint += model[last_word]
Discussion of the obligatory assignment

Do we need averaging step at all?

- Only one student tried to use simple sum of word vectors instead of average.
- Classifier performance jumped from 0.68 to 0.75...
- ...with less computation time.
- Why so?

Average text length (in words)

- The Daily Mail 389
- 4Traders 327
- Individual.com 229
- Latest Nigerian News 97

Discussion of the obligatory assignment

Capturing non-semantic signals

- Classes differ in typical document length.
- Longer documents produce semantic fingerprints with larger magnitudes (values).
- Averaging normalizes the magnitudes by the number of words: eliminates length differences.
- Without averaging, document vectors remain different.
- Logistic regression happily employs this signal for classification...
- ...but it is not related to document semantics.

Can be considered a sort of overfitting: performance will severely drop if typical text length changes.
Still, a very interesting finding!

Contents

1 Hot topics in the distributional semantics world
2 Discussion of the obligatory assignment
3 The exam: what to expect?
The exam: what to expect?

Nothing extremely difficult at the exam
▶ Mostly simply answering questions
▶ ...related to general understanding of the basic concepts
▶ ...and to practical aspects of prediction-based distributional models.
▶ At most one problem requiring (simple) calculation.
▶ The only formula you have to remember by heart is cosine distance.

Most essential reading
1. Chapters from ‘Speech and Language Processing’ by Jurafsky and Martin
2. ‘From Frequency to Meaning: Vector Space Models of Semantics’ by Turney and Pantel
3. ‘Word2vec parameter learning explained’ by Rong (at least skim through)
4. ‘Distributed representations of words and phrases and their compositionality’ by Mikolov et al.
5. ‘Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change’ by Hamilton et al.

The links are at the Syllabus page.

Exam-like problems at Dec 1 group session
1. Draw the scheme of how CBOW and Continuous Skipgram algorithms train.
2. Briefly describe all key elements of the neural network in these algorithms.
3. Enumerate and briefly describe all ways of standardized extrinsic evaluation of word embedding models that you can think of.
4. How evaluation metrics are related to syntagmatic or paradigmatic relations between words?
5. How many values (parameters) a trained prediction-based model contain?
6. How to estimate its size (in MBytes), if all the values are 32-bit floats?
7. etc...

Questions?

INF5820
Distributional Semantics: Extracting Meaning from Data
Thanks for your attention!
Good luck at the exam!


