Similarity and Vectors

Adam Meyers
New York University
2016
Summary

• Vectors representing Documents
  – IR and Document Classification
• Similarity between vectors
• Vectors representing Words
  – Word Similarity, Word Sense Disambiguation, Paraphrase/Entailment
• Reducing Dimensions of Large Vectors
• Neural Networks, aka, Deep Learning
Term Document Matrix: Information Retrieval

- Lecture 6 & Homework 5
- Matrix of documents and words
  - Columns → documents
  - Rows → words
  - Rows are vectors and columns are dimensions of the vectors
- Scores in matrix = TF-IDF scores
  - How significant is word $t$ at row for document at column?
  - $\text{TFIDF}(t) = \text{TF}(t) \times \text{IDF}(t)$
    - $\text{TF}(t)$ = measure of frequency of $t$ in document
    - $\text{IDF}(t)$ = measure of how few documents contain $t$
      \[
      \text{IDF}(t) = \log \left( \frac{\text{NumberOfDocuments}}{\text{NumberOfDocumentsContaining}(t)} \right)
      \]
Example: *coconut milk* vs. *tablespoon*

- **coconut milk**
  - occurs ~ 3 times in chicken and coconut soup recipe
    - Term frequency = 3
  - occurs in 4 out of 10,000 documents in collection
  - inverse document frequency = \( \log(10000/4) = \log(2500) = 7.82 \)
  - TFIDF = \( 3 \times 7.82 = 23.46 \)

- **tablespoon**
  - occurs 4 times in chicken and coconut soup recipe
    - Term frequency = 4
  - occurs in 1200 out of 10,000 documents in corpus
  - inverse document frequency = \( \log(10000/1200) = \log(8.33) = 2.12 \)
  - TFIDF = \( 4 \times 2.12 = 8.48 \)

- *coconut milk* is more highly weighted for Thai Soup recipes than *tablespoon*

- Note: Suitability of query term may depend on the nature of the collection
  - Is this a collection of recipes? – *tablespoon* not good search term
  - Is collection diverse: instructions, news, …? – *tablespoon* may be good search term
**Cosine Similarity: Similarity Between Vectors**

\[
\text{Similarity}(A, B) = \frac{\sum_i a_i \times b_i}{\sqrt{\sum_i a_i^2 \times \sum_i b_i^2}}
\]

- Cosine of the Angle Between the Vectors
- Cosine similarity high
  - if values of a and b are similar
  - If angle between vectors is small
- Used for all kinds of vectors
  - We applied these to Information Retrieval
  - But also apply to Word Sense Disambiguation, Sentiment Analysis, Paraphrase/Entailment, …
- Other similarity metrics: Jaccard, Dice, KL divergence, etc.
Information Retrieval Example

- Vectors have values corresponding to terms:
  - potato chip, chicken, sesame seed, coconut milk, ground beef

- 2 Queries
  - Q1 chicken, coconut milk: (0,5,0,5,0)
  - Q2 ground beef, potato chip: (4,0,0,0,7)

- 2 Documents
  - D1 Chicken and Coconut Soup Recipe: (0,7,0,9,0)
  - D2 Hamburger Recipe: (3,0,2,0,9)

- Cosign similarities

<table>
<thead>
<tr>
<th></th>
<th>Q1</th>
<th>Q2</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>99.2</td>
<td>0</td>
</tr>
<tr>
<td>D2</td>
<td>0</td>
<td>95.9</td>
</tr>
</tbody>
</table>
Other Uses of Document Vectors

• Document Classification
  – Given sets of documents with known classifications
  – Computer average vectors for each class
  – Create vector for unclassified document
  – Place new document in class with the highest similarity

• Sentiment Analysis
  – Like Document classification, but classes are sentiments
    • But may need different vectors for different domains/types of products/etc.
  – Words relevant to sentiment are selected for dimensions of vectors
    • part of challenge = choice of words (great, terrible, ....)
    • maybe domain specific (low interest: loans vs. investments)
  – Adjustments to account for negation
    • combine negative words with nearby sentiment words, e.g., “don't like” → not_like
Word Word Matrix Using Pointwise Mutual Information

- **Word Word Matrix (aka *word embedding*)**
  - Rows represent word \( R \)
  - Columns (aka *dimensions*) represent words co-occurring with word \( C \)
  - Can be generalized to multi-words (n-grams, phrases, …)
    - word to multi-word
    - multi-word to multi-word
  - Context can be defined other ways, e.g., proximity in syntactic tree

- **Approximation of meaning:**
  - Words in the same contexts tend to have similar meanings (Harris, 1954)
  - You shall know a word by the company it keeps (Firth, 1957)

- **Scores in Matrix**
  - How related is word \( R \) to word \( C \) represented by column \( C \)
  - Pointwise Mutual Information
    \[
    PMI = \log \left( \frac{\text{prob}(\text{word}_R, \text{word}_C)}{\text{prob}(\text{word}_R) \times \text{prob}(\text{word}_C)} \right)
    \]
Modifications to PMI

- Negative values should be treated as 0
- PMI is high for low frequency words
  - *banana* occurs once in the corpus of 1K words
  - *face* occurs twice in that corpus
  - *Banana face* occurs once in that corpus
    - \( \text{PMI}(\text{banana}, \text{face}) = \log_2 \left( \frac{.5}{.001 \times .002} \right) = 12.42 \)
  - Smoothing – different methods that raise the denominator slightly which offset this effect
    - Example: La Place – add a small constant to all e.g., add 1(banana = 2, banana face = 2, face = 3
      - \( \text{PMI}(\text{banana}, \text{face}) = \log_2 \left( \frac{.667}{.002 \times .003} \right) = 11.6 \)
Sample Word Embedding 1

• Assume a “bag of words” approach
  – Order of words don't matter
  – Assume that words are stemmed

• Use words in a window of $K$ words before and $K$ words after word $w_i$

• Let's assume $K = 5$ (for this example)

• Eliminate stop words and high frequency (low IDF) words

• Use integers in vectors (scores usually between 0 and 1)
Sample Word Embedding 2
From Hypothetical Recipe Corpus

- Rows = words being classified
- Columns = words in context
- Numbers = arbitrary score ranking likelihood that column word +/- 5 words from row word (higher number → higher rank)

<table>
<thead>
<tr>
<th></th>
<th>cup</th>
<th>ounce</th>
<th>taste</th>
<th>chicken</th>
<th>stir</th>
<th>bake</th>
<th>chocolate</th>
</tr>
</thead>
<tbody>
<tr>
<td>beef</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>cabbage</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>lemon</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>parsley</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>pepper</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>salt</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>sugar</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>
Cosine similarity for Word Vectors from Previous Slide

<table>
<thead>
<tr>
<th></th>
<th>beef</th>
<th>cabbage</th>
<th>lemon</th>
<th>parsley</th>
<th>pepper</th>
<th>salt</th>
<th>sugar</th>
</tr>
</thead>
<tbody>
<tr>
<td>beef</td>
<td>1</td>
<td>.63</td>
<td>.54</td>
<td>.57</td>
<td>.72</td>
<td>.66</td>
<td>.41</td>
</tr>
<tr>
<td>cabbage</td>
<td>.63</td>
<td>1</td>
<td>.25</td>
<td>.51</td>
<td>.53</td>
<td>.58</td>
<td>.51</td>
</tr>
<tr>
<td>lemon</td>
<td>.54</td>
<td>.25</td>
<td>1</td>
<td>.86</td>
<td>.64</td>
<td>.68</td>
<td>.74</td>
</tr>
<tr>
<td>parsley</td>
<td>.57</td>
<td>.51</td>
<td>.86</td>
<td>1</td>
<td>.81</td>
<td>.86</td>
<td>.69</td>
</tr>
<tr>
<td>pepper</td>
<td>.72</td>
<td>.53</td>
<td>.64</td>
<td>.81</td>
<td>1</td>
<td>.97</td>
<td>.44</td>
</tr>
<tr>
<td>salt</td>
<td>.66</td>
<td>.58</td>
<td>.68</td>
<td>.86</td>
<td>.97</td>
<td>1</td>
<td>.56</td>
</tr>
<tr>
<td>sugar</td>
<td>.41</td>
<td>.51</td>
<td>.74</td>
<td>.69</td>
<td>.44</td>
<td>.56</td>
<td>1</td>
</tr>
</tbody>
</table>
Demo for find similar words

Word Sense Disambiguation

- Demo of A word sense diambiguator demo
  - http://www.ling.gu.se/~lager/Home/pwe_ui.html

- Shared tasks include Semcor
  - http://web.eecs.umich.edu/~mihalcea/downloads.html#semcor

- Using Word Vectors for Word Sense Disambiguation
  - Vectors represent word senses rather than words
    - Need sense annotated corpus
  - Create vectors for words in new text
  - Compute similarity of words in new text with sense vectors and choose most similar sense
Paraphrase and Entailment

- SemEval Text Similarity Task:
  - 2016 (Task 1)
  - [http://alt.qcri.org/semeval2014/task1/](http://alt.qcri.org/semeval2014/task1/) (webpage)

- Input pairs of text “snippets”
  - English/English (like previous year tasks)
  - Spanish/English pairs (innovation for 2016)
    - previous snippets, with one member of pair translated

- System produces score from 0 to 5 indicating similarity
- Manually tagged data (test, dev, training sets)
- Data collection of snippets based on heuristics and manually annotate
  - One heuristic is based on word embedding similarity – embedding of sentence = sum of the embeddings of words

Similarity and Vectors
Computational Linguistics
2016
Human Judge similarity 0 to 5 (from Agirre et al 2016)

• 5 mean exactly the same thing
  – *The bird is in the sink* ↔ *Birdie is washing itself in the water basin*

• 4 mostly the same, differences unimportant
  – *In May 2010, the troops attempted to invade Kabul* ≈ *The US army invaded Kabul on May 7th last year, 2010*

• 3 roughly same with important differences/omissions
  – *John said he is considered a witness but not a suspect* → *“He is not a suspect anymore.” John said*

• 2 same topic, share some details
  – *They flew out of the nest in groups* ≠ *They flew out of the nest together*

• 1 same topic
  – *The woman is playing the violin* ≠ *The young lady enjoys listening to guitar*

• 0 disimilar
  – *John went horse back riding with a whole group of friends* ≠ *Sunrise at dawn is a magnificent view to take in if you wake up early enough for it*
Evaluation

- Systems scored by the Pearson correlation between their scores and the Manual Annotation
- Samsung's system got the highest score: .7781
- I looked at papers about the top 3 systems
  - All used word embeddings in one form or another
Top System (Samsung) used Word Embeddings

• Vectors contained words & multi-word phrases
  – Methods for combining embeddings of words into embeddings of sentences
• Used other features, e.g., from WordNet
• Used dependency parses of snippets
• Machine Learning Algorithms (e.g., SVM)
  – To predict 0 to 5 Textual Similarity Score
  – Features include cosine similarity of roots of parses
    • Similarity derived by combining children similarities according to an algorithm
• Most top systems used Word Embeddings
Real Vectors have Many Dimensions

• Preceding “toy” examples use few dimensions
• Vectors often have tens of thousands of dimensions
• More dimensions
  – Better output (higher recall and precision)
  – Slower speed (e.g., takes longer to compute similarity)
• Large Vectors are sparse (lots of zeros)
• Context: window of 3 to 17 (or the whole sentence)
• Reducing dimensions to make smaller, less sparse vectors
  – Capture Generalizations, more efficient processing, etc.
  – One such method is called Latent Semantic Analysis
  – Many other methods for refining vector-based analyses
Latent Semantic Analysis: Reducing Dimensions

Original 2-D Vector

Rotate/Move So Points Are Closer To The X and Y Axes

Eliminate One Dimension
Other factors

• Softmax functions: functions that normalize a range of values from 0 to 1, so they can be used as probabilities

• Eliminating dimensions that do not discriminate between vectors, high/low frequency words, words with low IDF, etc.

• Feature types
  – Bag of Words Feature (so far)
  – Features that include Relative positions
  – Features based on parser output, dictionaries, other databases, …
Deep Learning

- Initialize vectors with scores predicting words given neighboring words
  - Randomly initialize weights according to a prior distribution
  - Randomly initialize parameterized-length matrices (weights of the network)
    - these represent layers of the Neural Network
- Weights are tuned by running multiple times on different pieces of training corpus
  - On each batch, weights are adjusted to improve probabilities
  - For example, maximizing the average log of the probabilities that each (center) word is predicted by neighboring words
  - Training ends when probabilities converge or after maximum number of iterations
- Example Deep Learning (aka Neural Network) approaches
  - Word2Vec CBOW and Skip-gram
  - Convolutional Neural Networks
  - Recurrent Neural Networks
Deep Learning at NYU

• Machine Translation
  – Prof. Kyunghyun Cho (http://www.kyunghyuncho.me/)

• Natural Language Semantics
  – Prof. Sam Bowman (https://www.nyu.edu/projects/bowman/)

• ACE Event Detection
  – Thien Nguyen (http://www.cs.nyu.edu/~thien/)

• And Others
Documentation and Code

- Jurafsky and Martin 3rd Edition (Chapters 15 and 16)
- Word2Vec
  - https://www.tensorflow.org/versions/r0.12/tutorials/word2vec/index.html
  - https://deeplearning4j.org/word2vec
  - https://github.com/dav/word2vec
Summary

• Vector characterizations of documents
  – Dimensions represent terms relevant to classification
  – IR – dimensions represent query terms
  – Sentiment – dimensions represent opinion words
  – Topics – dimensions represent topic words

• Vector characterization of words (word embeddings)
  – Dimensions represent words in context within a window
  – Related words/word-senses/translations/etc. have similar embeddings

• Dimensions are weighted using TF-IDF, PMI and other metrics
• Similarity is calculated with Cosine Similarity, Jaccard similarity, …
• Real systems use large sparse vectors which are converted into smaller dense vectors, using various “deep learning” methods