Lecture 14: Distributed Word Representations for Information Retrieval

Section 9.2.2 Query Expansion: How can we robustly match a user's search intent?

- *Synonymy*: In most collections, the same concept may be referred to using different words.
  - Has an impact on the *recall* of most IR systems
  - Users often attempt to manually refine their queries
  - How could an IR system help with query refinement?
  - We want to understand a query, rather than simply matching keywords. We want to better understand *when* query and docouments match
- *Query expansion*: Users give *additional* input on query words or phrases, possibly suggesting additional query terms
  - The users opting to use one of the alternative query suggestions
- How to generate alternative or expanded queries for the user?
  - Global analysis: For each term in a query, automatically expand the query using synonyms and related words from thesaurus
  - Local analysis: Analyze the documents in the current results set
    * Feedback on documents or on query terms
- How to build a theasaurus?
  - Use of a controlled vocabulary maintained by human editors: Canonical terms for each concepts
  - Manual theasarus: Synonymous names for concepts, without designating a canonical term.
  - Automatically derived thesaurus
    * Using word co-occurrence statistics: words co-occurring in a document or paragraph are likely to be in some sense *similar or related in meaning*
    * Exploiting grammatical relations or dependencies: less robust than co-occurrence statistics, but more accurate
    * Quality of the resulting terms often not so good
    * Since those terms are highly correlated in documents anyway, this method may not retrieve that many additional documents.
  - Query reformulations based on query log mining: Exploit the *manual query reformulations* of other users
- Use of query expansion generally increases recall
  - A domain-specific thesaurus is required.
  - May significantly decrease precision, particularly when the query contains ambiguous terms.

How can we represent term relations?

- Under the standard symbolic encoding of terms, different terms have no direct way of representing their similarities.

- Basic IR is scoring on $q^T d$. Can we learn parameters W to rank via $q^T W d$?
  - Berger and Lafferty 1999, Query translation model
  - W is huge ($> 10^{10}$): Sparsity is the problem
- We could learn a dense low-dimensional representation of a word in $\mathcal{R}^d$, such that dot products $u^T v$ express word similarity.
- Supervised Semantic Indexing shows successful use of learning W
- This lecture will however consider *direct similarity*
- Traditional way: Latent Semantic Indexing/Analysis - Use SVD; Results were somewhat iffy

Neural Embeddings

- Build a dense vector for each unique word, chosen so that it is good at predicting *other words appearing in its context*
- To do that, build a NN model that predict between a *center word* w_t and its set of context words
  - Directly learn *low-dimensional* word vectors, based on ability to predict
  - Learned weights of this model would be the *word embeddings*
- **Word2Vec**:
  - Two Algorithms (Word embedding models)
    1. Skip-grams: Predict *context words* given target
    2. Continuous Bag of Words: Predict *target word* from bag-of-words context
  - Two(Three?) Training Methods
    1. Hierarchical softmax
    2. Negative sampling
    3. Naive softmax
  - Skip-grams overview
    * For each position $t = 1, \cdots, T$, predict context words within a window of fixed size m, given center word w_j.
    * Likelihood: $L(\Theta) = \prod_{t=1}^{T} \prod_{-m \leq j \leq m, j \neq 0} P(w_{t+j} \mid w_t)$
    * Objective is the average negative log likelihood: $J(\Theta) = -\frac{1}{T} \log L(\Theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{-m \leq j \leq m, j \neq 0} \log P(w_{t+j} \mid w_t)$
- These representations are very good at encoding *similarity* and *dimensions of similarity*.

Dual Embedding Space Model (DESM)

- Word2Vec CBOW models learn 2 different word embeddings
  - One for target word (IN)
  - One for context word (OUT)
- We usually retain just one of the two, depending on whether we use CBOW or SG.
- But *interactions* of two seperate embedding spaces capture additional distributional semantics of words. **Let's combine the two**

  - The CBOW model pushes the IN vectors (representing context words for the missing target word) closer to the OUT vector of other words that *they commonly co-occur with.*
  - The IN-IN or OUT-OUT cosine similarities are higher for words that are similar in terms of **type or funtion**
  - The IN-OUT (or very likely, OUT-IN) cosine similarities are higher for words that **co-occur often in the training corpus**
- In the setting of ranked retrieval,
  - Represent a document as a centroid of its word vectors in OUT space
  - Represent each query term as a vector in IN space
  - Then calculate the cosine similarities between the centroid and each term, and average them.
- This allows us to capture *aboutness*: words that appear with this word
- Using DESM solely to rank documents on the entire collection generates too many *false positives*
  - Example: Given the query `cambridge`, the documents about `oxford` get high scores, because those documents would have similar context words
- However, DESM is effective at finding subtler similarities/aboutness
  - Allows ranking documents that is actually about the query terms higher, than the ones merely mentions the terms
  - Example: Get the document about `giraffe`, and replace all occurrence of `giraffe` with `cambridge` $\rightarrow$ This document still scores low
- Good results were shown when it was used as a re-ranking method for a smaller set of documents, given by other document ranking features such as TF-IDF