Outline

Outline

Unsupervised semantic matching with pre-training

Word embeddings have recently gained popularity for their ability to encode semantic and syntactic relations amongst words.

How can we use word embeddings for information retrieval tasks?

Word Embedding

Distributional Semantic Model (DSM): A model for associating words with vectors that can capture their meaning. DSM relies on the distributional hypothesis.

Distributional Hypothesis: Words that occur in the same contexts tend to have similar meanings [Harris, 1954].

Statistics on observed contexts of words in a corpus is quantified to derive word vectors.

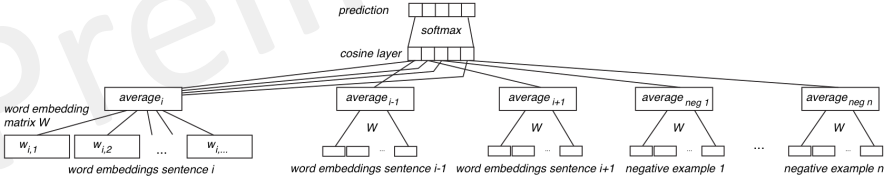
- The most common choice of context: The set of words that co-occur in a context window.
- Context-counting VS. Context-predicting [Baroni et al., 2014]

From Word Embedding to Query/Document Embedding

Obtaining representations of compound units of text (in comparison to the atomic words).

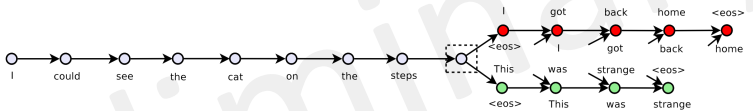
Bag of embedded words: sum or average of word vectors.

- Averaging the word representations of query terms has been extensively explored in different settings. [Vulić and Moens, 2015, Zamani and Croft, 2016b]
  - Effective but for small units of text, e.g. query [Mitra, 2015].
- Training word embeddings directly for the purpose of being averaged [Kenter et al., 2016].



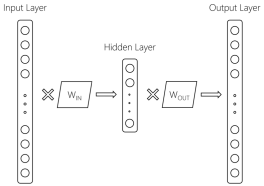
From Word Embedding to Query/Document Embedding

- Skip-Thought Vectors
  - Conceptually similar to distributional semantics: a units representation is a function of its neighbouring units, except units are sentences instead of words.



  - Similar to auto-encoding objective: encode sentence, but decode neighboring sentences.
  - Pair of LSTM-based seq2seq models with share encoder.
- Doc2vec (Paragraph2vec) [Le and Mikolov, 2014].
- You'll hear more later about it on "Learning unsupervised representations from scratch". (Also you might want to take a look at Deep Learning for Semantic Composition)

Dual Embedding Space Model (DESM) [Nalisnick et al., 2016]



Word2vec optimizes IN-OUT dot product which captures the co-occurrence statistics of words from the training corpus:

- We can gain by using these two embeddings differently

yale			**seahawks**			**eminem**		
IN-IN	OUT-OUT	IN-OUT	IN-IN	OUT-OUT	IN-OUT	IN-IN	OUT-OUT	IN-OUT
yale	yale	yale	seahawks	seahawks	seahawks	eminem	eminem	eminem
harvard	uconn	faculty	49ers	broncos	highlights	rihanna	rihanna	rap
nyu	harvard	alumni	broncos	49ers	jerseys	ludacris	dre	featuring
cornell	tulane	orientation	packers	nfl	tshirts	kanye	kanye	tracklist
tulane	nyu	haven	nfl	packers	seattle	beyonce	beyonce	diss
tufts	tufts	graduate	steelers	steelers	hats	2pac	tupac	performs

- IN-IN and OUT-OUT cosine similarities are high for words that are similar by function or type (typical) and the
- IN-OUT cosine similarities are high between words that often co-occur in the same query or document (topical).

Pre-trained word embedding for document retrieval and ranking

DESM [Nalisnick et al., 2016]: Using IN-OUT similarity to model document aboutness.

- A document is represented by the centroid of its word OUT_vectors:

$$\vec{v}_{d,\text{OUT}} = \frac{1}{|d|} \sum_{t_d, \in d} \frac{\vec{v}_{t_{d,\text{OUT}}}}{||\vec{v}_{t_{d,\text{OUT}}}||}$$

- Query-document similarity is average of cosine similarity over query words:

$$\text{DESM}_{\text{IN-OUT}}(q, d) = \frac{1}{q} \sum_{t_q \in q} \frac{\vec{v}_{t_{q,\text{IN}}}^{\top} \vec{v}_{t_{d,\text{OUT}}}}{||\vec{v}_{t_{q,\text{IN}}}|| \; ||\vec{v}_{t_{d,\text{OUT}}}||}$$

- IN-OUT captures more Topical notion of similarity than IN-IN and OUT-OUT.
- DESM is effective at, but only at, ranking at least somewhat relevant documents.

Pre-trained word embedding for document retrieval and ranking

- NTLM [Zuccon et al., 2015]: Neural Translation Language Model
  - Translation Language Model: extending query likelihood:

$$p(d|q) \sim p(q|d)p(d)$$
$$p(q|d) = \prod_{t_q \in q} p(t_q|d)$$
$$p(t_q|d) = \sum_{t_d \in d} p(t_q|t_d)p(t_d|d)$$

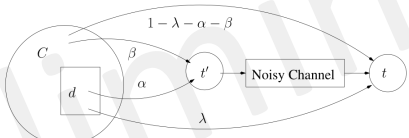
  - Uses the similarity between term embeddings as a measure for term-term translation probability $p(t_q|t_d)$.

$$p(t_q|t_d) = \frac{\cos(\vec{v}_{t_q}, \vec{v}_{t_d})}{\sum_{t \in V} \cos(\vec{v}_t, \vec{v}_{t_d})}$$

Pre-trained word embedding for document retrieval and ranking

GLM [Ganguly et al., 2015]: Generalize Language Model

- Terms in a query are generated by sampling them independently from either the document or the collection.
- The noisy channel may transform (mutate) a term t into a term t'.



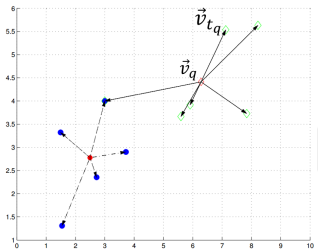
$$p(t_q|d) = \lambda p(t_q|d) + \alpha \sum_{t_d \in d} p(t_q, t_d|d) p(t_d) + \beta \sum_{t' \in N_t} p(t_q, t'|C) p(t') + 1 - \lambda - \alpha - \beta) p(t_q|C)$$

N_t is the set of nearest-neighbours of term t.

$$p(t', t|d) = \frac{\mathsf{sim}(\vec{v}_{t'}, \vec{v}_t).\mathsf{tf}(t', d)}{\sum_{t_1 \in d} \sum_{t_2 \in d} \mathsf{sim}(\vec{v}_{t_1}, \vec{v}_{t_2}).|d|}$$

Pre-trained word embedding for query term weighting

Term re-weighting using word embeddings [Zheng and Callan, 2015].
- Learning to map query terms to query term weights.



- Constructing the feature vector $\vec{x}_{t_q}$ for term t_q using its embedding and embeddings of other terms in the same query q as:

$$\vec{x}_{t_q} = \vec{v}_{t_q} - \frac{1}{|q|} \sum_{t'_q \in q} \vec{v}_{t'_q}$$

- $\vec{x}_{t_q}$ measures the semantic difference of a term to the whole query.
- Learn a model to map the feature vectors the defined target term weights.

Pre-trained word embedding for query expansion

- Identify expansion terms using word2vec cosine similarity [Roy et al., 2016].
  - pre-retrieval:
    - Taking nearest neighbors of query terms as the expansion terms.
  - post-retrieval:
    - Using a set of pseudo-relevant documents to restrict the search domain for the candidate expansion terms.
  - pre-retrieval incremental:
    - Using an iterative process of reordering and pruning terms from the nearest neighbors list.
      - Reorder the terms in decreasing order of similarity with the previously selected term.
- Works better than having no query expansion, but does not beat non-neural query expansion methods.

Pre-trained word embedding for query expansion

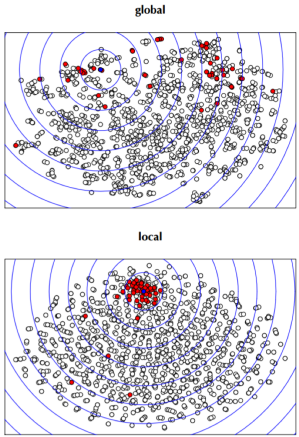
- Embedding-based Query Expansion [Zamani and Croft, 2016a]
  Main goal: Estimating a better language model for the query using embeddings.
  - Two models with different assumptions:
    - Conditional independence of query terms.
    - Query-independent term similarities.
    Different calculation of the probability of expansion terms given the query.
  - Choosing top-k probable terms as expansion terms.
- Embedding-based Relevance Model:
  Main goal: Semantic similarity in addition to term matching for PRF.

$$P(t|\theta_F) = \sum_{d \in F} p(t, q, d)$$
$$= \sum_{t \in V} p(q|t, d) p(t|d) p(d)$$

$$p(q|t, d) = \beta p_{\mathsf{tm}}(q|t, d) + (1 - \beta) p_{\mathsf{sm}}(q|t, d)$$

Pre-trained word embedding for query expanssion

Query expansion with locally-trained word embeddings [Diaz et al., 2016].



- Main idea: Embeddings be learned on topically-constrained corpora, instead of large topically-unconstrained corpora.
- Training word2vec on documents from first round of retrieval.
- Fine-grained word sense disambiguation.
- A large number of embedding spaces can be cached in practice

Outline

Semi-supervised semantic matching

Using unsupervised pre-trained word embeddings, we have vector space of words that we have to put to good use to create query and document representations.

However, in information retrieval, there is the concept of pseudo relevance that gives us a supervised signal that was obtained from unsupervised data collections.

Outline

Pseudo test/training collections

Given a source of pseudo relevance, we can build pseudo training or test collections. We can

- use the pseudo training collections to train a model and then test on a non-pseudo test collection, or
- use the pseudo test collections to verify models in a domain where human judgments are lacking or incomplete.

History of pseudo test collections

Problems in the simulation of bibliographic retrieval systems [Tague et al., 1980]

"If tests are carried out with large operational systems, there are difficulties in experimentally controlling and modifying the variables [of bibliographic retrieval systems]. [...] An alternative approach [...] is computer simulation."

Use simulation to investigate the **complexity** (data structures) and **effectiveness** (query/document representation) of retrieval systems.

How to determine query/document relevance?

Synthesize a separate set of relevant documents for a query [Tague et al., 1980] or sample judgments for every query and all documents from a probabilistic model [Tague and Nelson, 1981].

Modern pseudo test collections for evaluating effectiveness (1/2)

Research focused on validating pseudo relevance with non-pseudo judgments.

Web search [Beitzel et al., 2003]

Find sets of pseudo-relevant documents using the Open Directory Project. Queries are editor-entered document titles (document with exact title is relevant) and category names (leaf-level documents are relevant).

Known-item search [Azzopardi et al., 2007]

Compare manual query/judgments with pseudo query/judgments using a Kolmogorov-Smirnov (KG) test on multi-lingual documents from government websites.

Desktop search [Kim and Croft, 2009]

Building upon Azzopardi et al. [2007], construct a pseudo test collection for enterprise search and verify its validity using a KG test.

Modern pseudo test collections for evaluating effectiveness (2/2)

Archive search [Huurnink et al., 2010a,b]

Generate queries and judgments using the strategy of Azzopardi et al. [2007] and validate using transactions logs of an audiovisual archive.

Product search [Van Gysel et al., 2016]

Construct queries from product category hierarchies and estimate product relevance by category membership [Beitzel et al., 2003], grounded in observations from e-commerce research. Pseudo test collections are then used to evaluate unsupervised neural representation learning algorithms (see Slide 94).

From testing to training using pseudo relevance

At some point, pseudo relevance began to be used to train retrieval functions. Learning To Rank models (see later) trained using pseudo relevance outperform non-supervised retrieval functions (e.g., BM25) on TREC collections.

Web search using anchor texts [Asadi et al., 2011]

Construct a pseudo relevance collection from anchor texts in a web corpus and use it to train Learning To Rank (LTR) models. LTR models trained using pseudo relevance outperform BM25 on TREC collections.

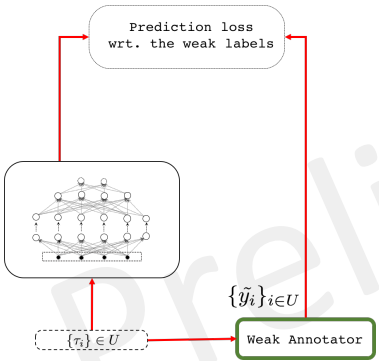
Microblog search using hashtags [Berendsen et al., 2013]

Tweets with a hashtag are relevant to the topic covered by the hashtag. Queries are constructed by sampling terms from tweets that discriminate the relevant set from the non-relevant set.

Outline

Training neural networks using pseudo relevance

Training a neural ranker using weak supervision [Dehghani et al., 2017].

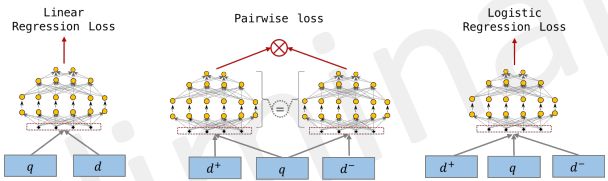


Main idea: Annotating a large amount of unlabeled data using a weak annotator (Pseudo-Labeling) and design a model which can be trained on weak supervision signal.

- Function approximation. (re-inventing BM25?)
- Beating BM25 using BM25!

Training neural networks using pseudo relevance

- Employed three different architectures: Score, Rank, and RankProb



- Employed Three different feeding paradigms:
  - Dense: $\psi(q,d) = [N||avg(l_d)_D||l_d||\{df(t_i)||tf(t_i,d)\}_{1\leq i\leq k}]$
  - Sparse: $\psi(q,d) = [tfv_c||tfv_q||tfv_d]$
  - Embedding-based: $\psi(q,d) = [\odot_{i=1}^{|q|}(\mathcal{E}(t_i^q),\mathcal{W}(t_i^q))||\odot_{i=1}^{|d|}(\mathcal{E}(t_i^d),\mathcal{W}(t_i^d))]$,

Training neural networks using pseudo relevance

Lesson Learned:

- Define an objective which enables your model to go beyond the imperfection of the weakly annotated data. (ranking instead of calibrated scoring)
- Let the network decide about the representation. Feeding the network with featurized input kills the model creativity!
- If you get enough data, you can learn embedding which is better fitted to your task by updating them just based on the objective of the downstream task.
- You can compensate the lack of enough training data by pertaining your network on weakly annotated data.

Training neural networks using pseudo relevance

Generating weak supervision training data for training neural IR model [MacAvaney et al., 2017].

- Using a news corpus with article headlines acting as pseudo-queries and article content as pseudo-documents.
- Problems:
  - **Hard-Negative**
  - **Mismatched-Interaction**: (example: "When Bird Flies In", a sports article about basketball player Larry Bird)
- Solutions:
  - **Ranking filter**:
    - top pseudo-documents are considered as negative samples.
    - only pseudo-queries that are able to retrieve their pseudo-relevant documents are used as positive samples.
  - **Interaction filter**:
    - building interaction embeddings for each pair.
    - filtering out based on similarity to the template query-document pairs.

Query expansion using neural word embeddings based on pseudo relevance

Locally trained word embeddings [Diaz et al., 2016]

- Performing topic-specific training, on a set of topic specific documents that are collected based on their relevance to a query.

Relevance-based Word Embedding [Zamani and Croft, 2017].

- Relevance is not necessarily equal to semantically or syntactically similarity:
  - "united state" as expansion terms for "Indian American museum".
- Main idea: Defining the "context"
  Using the relevance model distribution for the given query to define the context. So the objective is to predict the words observed in the documents relevant to a particular information need.
- The neural network will be constraint by the given weights from RM3 to learn word embeddings.

Outline

Learning unsupervised representations from scratch

- Pseudo relevance judgments allow the training of supervised models in the absence of human judgments or implicit relevance signals (e.g., clicks).
- Introduces a dependence on relevance models, hypertext, query lists, ...
- Unsupervised retrieval models (e.g., BM25, language models) operate without pseudo relevance.

Can we learn a model of relevance in the absence of any relevance judgments?

LSI, pLSI and LDA

History of latent document representations

Latent representations of documents that are learned from scratch have been around since the early 1990s.

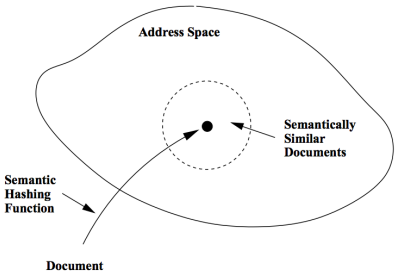
- Latent Semantic Indexing [Deerwester et al., 1990],
- Probabilistic Latent Semantic Indexing [Hofmann, 1999], and
- Latent Dirichlet Allocation [Blei et al., 2003].

These representations provide a semantic matching signal that is complementary to a lexical matching signal.

Semantic Hashing

Salakhutdinov and Hinton [2009] propose Semantic Hashing for document similarity.

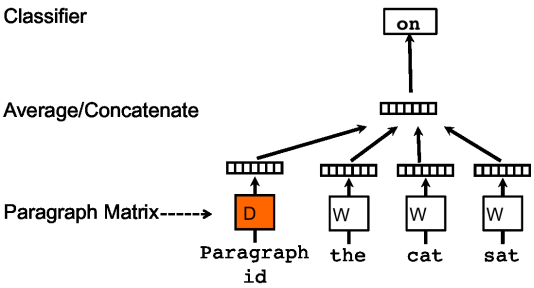
- Auto-encoder trained on frequency vectors.
- Documents are mapped to memory addresses in such a way that semantically similar documents are located at nearby bit addresses.
- Documents similar to a query document can then be found by accessing addresses that differ by only a few bits from the query document address.



Schematic representation of Semantic Hashing. Taken from Salakhutdinov and Hinton [2009].

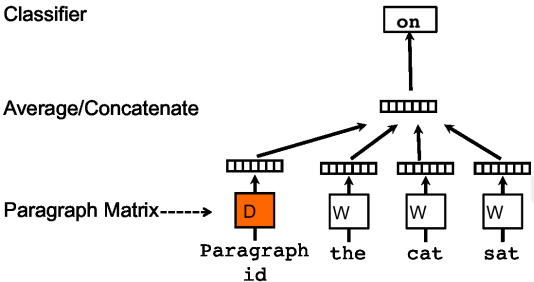
Distributed Representations of Documents [Le and Mikolov, 2014]

- Learn document representations based on the words contained within each document.
- Reported to work well on a document similarity task.
- Attempts to integrate learned representations into standard retrieval models [Ai et al., 2016a,b].

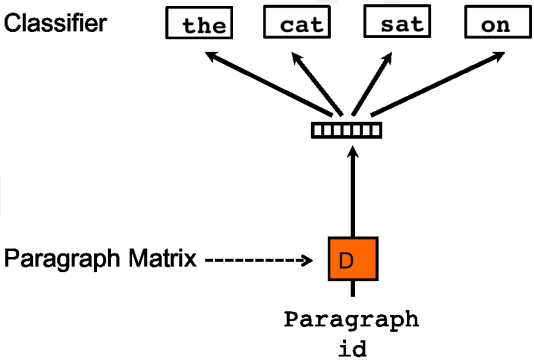


Overview of the Distributed Memory document vector model. Taken from Le and Mikolov [2014].

Two Doc2Vec Architectures [Le and Mikolov, 2014]



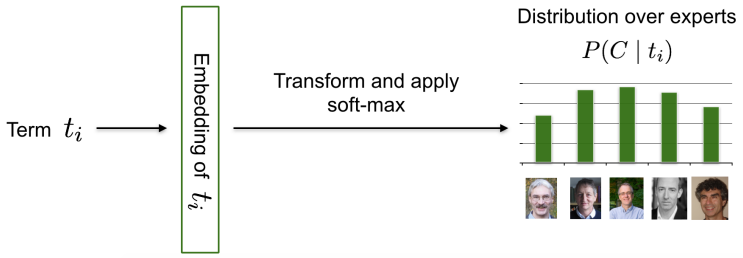
Overview of the Distributed Memory document vector model. Taken from Le and Mikolov [2014].



Overview of the Distributed Bag of Words document vector model. Taken from Le and Mikolov [2014].

Semantic Expertise Retrieval [Van Gysel et al., 2016]

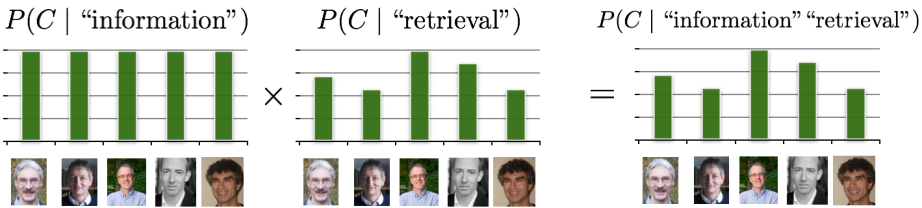
- Expert finding is a particular entity retrieval task where there is a lot of text.
- Learn representations of words and entities such that n-grams extracted from a document predicts the correct expert.



Taken from slides of Van Gysel et al. [2016].

Semantic Expertise Retrieval [Van Gysel et al., 2016] (cont'd)

- Expert finding is a particular entity retrieval task where there is a lot of text.
- Learn representations of words and entities such that n-grams extracted from a document predicts the correct expert.



Taken from slides of Van Gysel et al. [2016].

Regularities in Text-based Entity Vector Spaces [Van Gysel et al., 2017b]

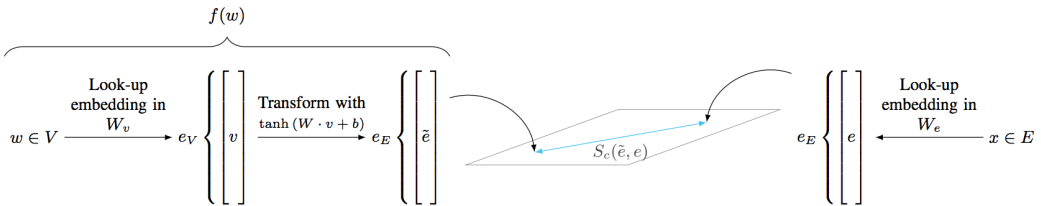
To what extent do entity representation models, trained only on text, encode structural regularities of the entity's domain?

Goal: give insight into learned entity representations.

- Clusterings of experts correlate somewhat with groups that exist in the real world.
- Some representation methods encode co-authorship information into their vector space.
- Rank within organizations is learned (e.g., Professors $>$ PhD students) as senior people typically have more published works.

Latent Semantic Entities [Van Gysel et al., 2016]

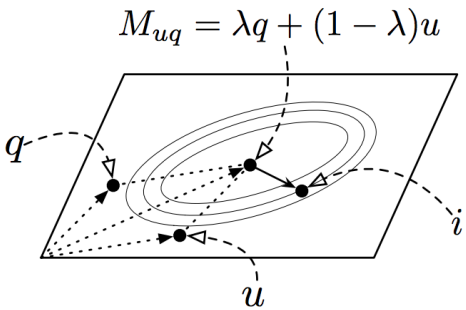
- Learn representations of e-commerce products and query terms for product search.
- Tackles learning objective scalability limitations from previous work.
- Useful as a semantic feature within a Learning To Rank model in addition to a lexical matching signal.



Taken from slides of Van Gysel et al. [2016].

Personalized Product Search [Ai et al., 2017]

- Learn representations of e-commerce products, query terms, and users for personalized e-commerce search.
- Mixes supervised (relevance triples of query, user and product) and unsupervised (language modeling) objectives.
- The query is represented as an interpolation of query term and user representations.



Personalized product search in a latent space with query $\vec{q}$, user $\vec{u}$ and product item $\vec{i}$. Taken from Ai et al. [2017].

Outline

Document & entity representation learning toolkits

gensim : `https://github.com/RaRe-Technologies/gensim` [Řehůřek and Sojka, 2010]

SERT : `http://www.github.com/cvangysel/SERT` [Van Gysel et al., 2017a]

HEM : `https://ciir.cs.umass.edu/downloads/HEM` [Ai et al., 2017]