Word Representation Learning

Sebastian Hofstätter

sebastian.hofstaetter@tuwien.ac.at
/s_hofstaetter

Today

Word Representation Learning

1 Word embeddings

- From characters to vectors
- Word2Vec

2 A glimpse at Word Embeddings in IR

- Unsupervised Query Expansion
- Mitigating Topic Shifting

Differentiable Matrices & Transforms

- Gradient descent based neural networks operate only in continuous spaces: Tensors of floating point numbers and continuous transformation functions
  - Without continuous values and functions -> no gradient
- This means we can’t just input the character-values of words in a linear algebra “network” and expect it to work
- We need to map chars or word-pieces or words to some sort of vector
  - A lot of options have been developed to do that – we’ll look at some of them

If you haven’t, follow https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html

First Pre-training then Fine-tuning

- A common problem in all ML projects is missing labeled data
  - Especially for language: a problem to get good representations for each word
  - Low-frequency terms, might be important indicators for your task,
    but are even less likely to appear in training data
- Pre-training word representations on a general text corpus is enormously helpful
  - This means that most words get good “starting” representations
- Fine-tuning: Train starting with the pre-trained representations with your (limited) data
- Major factor in ubiquity of word embeddings / language models in NLP

Take Pre-trained then Stack together

- Everything in a neural network can be combined like LEGO blocks
  - Like computer vision models
  - Also possible in the text domain (and it's getting better)
- Fine-tuning can be:
  - Take the pre-trained network 1:1 and just change the input & label data
  - Take the pre-trained module (e.g. word embedding) and put it in a bigger model; where the rest of the model is randomly initialized
- Word Embeddings are often used as the first building block in a network
  - Except for large pre-trained language models: BERT etc..
    (that are purposefully multi-task pre-trained)

Word Representation Learning

Words as vectors …

Working with Natural Language

- Inspiration for statistical methods from Ludwig Wittgenstein*:
  For a large class of cases-though not for all-in which we employ the word "meaning" it can be defined thus: the meaning of a word is its use in the language.

- Human language is highly ambiguous and context dependent
  *Example “love”: Love for people (parent, child), Love for other living creatures (cat), Love for things (food), Love for less concrete things (travelling)***

- Language is symbolic -> Word Embeddings

*Ludwig Wittgenstein - Philosophical investigations §43

**Thomas McNally 2017, Wittgenstein and the Philosophy of Language: The Legacy of the Philosophical Investigations.

Representation Learning

- Learning representations of data to make it easier to extract useful information when building classifiers or other predictors
    - Representation learners are a module, part of a larger network

- Goal is to create more abstract—and ultimately more useful—representations

- A reasonably sized learned representation should capture a huge number of possible input configurations

Bengio, Y., Courville, A. and Vincent, P., 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*

Word Embeddings

- Provide a dense vector representation for words
  - Typically 100-300 dimension
  - The dimensions are abstract
  - The vector space allows for math operations
    - For example nearest neighbors: semantic related words are close together in the space
- Can be unsupervised pre-trained on huge text data sets
  - Wikipedia, CommonCrawl, or more domain specific
  - And fine-tuned inside a model (end-to-end trained)
- Super simple data structure: `Dictionary<string, float[]>`
  - A major factor for their success: ease of use

Word Embeddings

- There are many unsupervised (creation) methods
  - 1-Word-1Vector:
    - *Word2Vec (Skip-Gram & CBOW)*
    - *Glove*
    - *A lot of specialized variants of the two*
  - *1-Word-1-Vector+Char-n-grams*
    - *FastText (based on Word2Vec)*
  - *Contextualized / context dependent / complex structure (char or word piece based)*
    - *ELMo*
    - *Transformers a la BERT and its variants (State of the Art - Neural IR 2 lecture)*

Fast + easy to use library (training & pre-trained): Gensim https://github.com/RaRe-Technologies/gensim

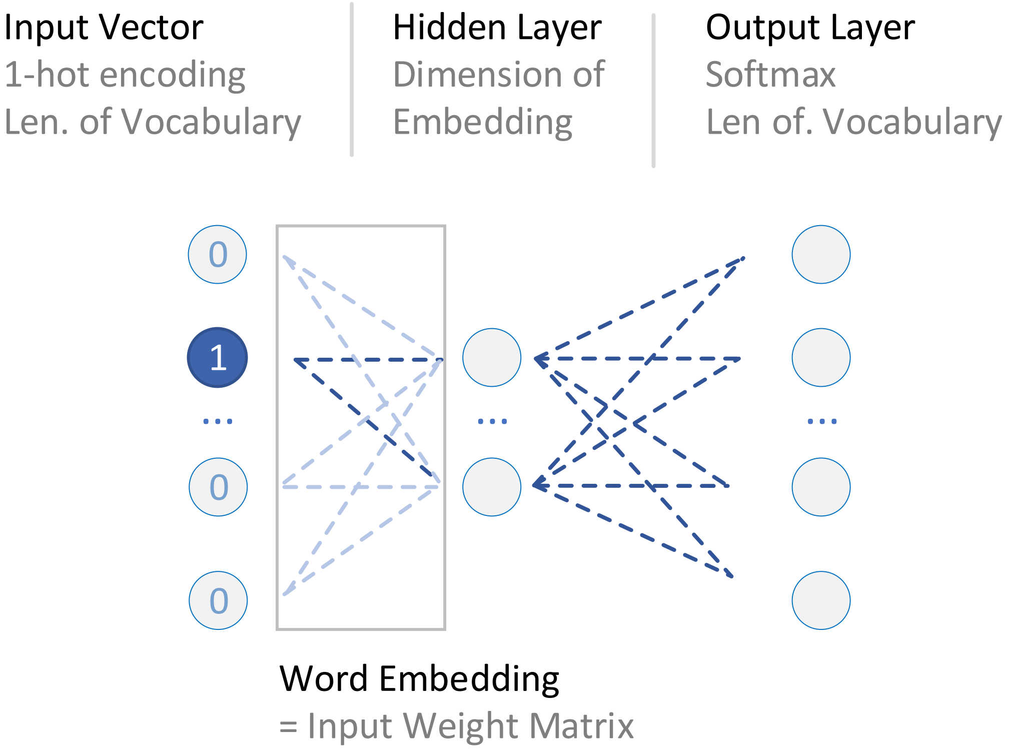
Unsupervised Training: Language Modelling

- We don’t have explicit labels = unsupervised
- But we have real text, how people use language – we model that
- Task: Predict next word given a sequence of words
  - Allows us to compute loss based on probability over a vocabulary
- Main technique for text pre-training
- Many variants exist:
  - Predict context words -2,-1,+1,+2 ..
  - Predict masked words in sequence (Masked Language model)

More about MLM in the Transformers lecture

Word2Vec

- Train a 1 hidden layer network to predict context words
  - Language Modelling
- Target words via 1-hot encoding
- Harvest the word vectors from the network = take the matrix
  - Each row is now corresponding to the 1-hot position of a word
- Output matrix is ignored (mostly)



Input Vector
1-hot encoding
Len. of Vocabulary

Hidden Layer
Dimension of
Embedding

Output Layer
Softmax
Len of. Vocabulary

Word Embedding
= Input Weight Matrix

Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Proc. Of NeurIPS. 2013.

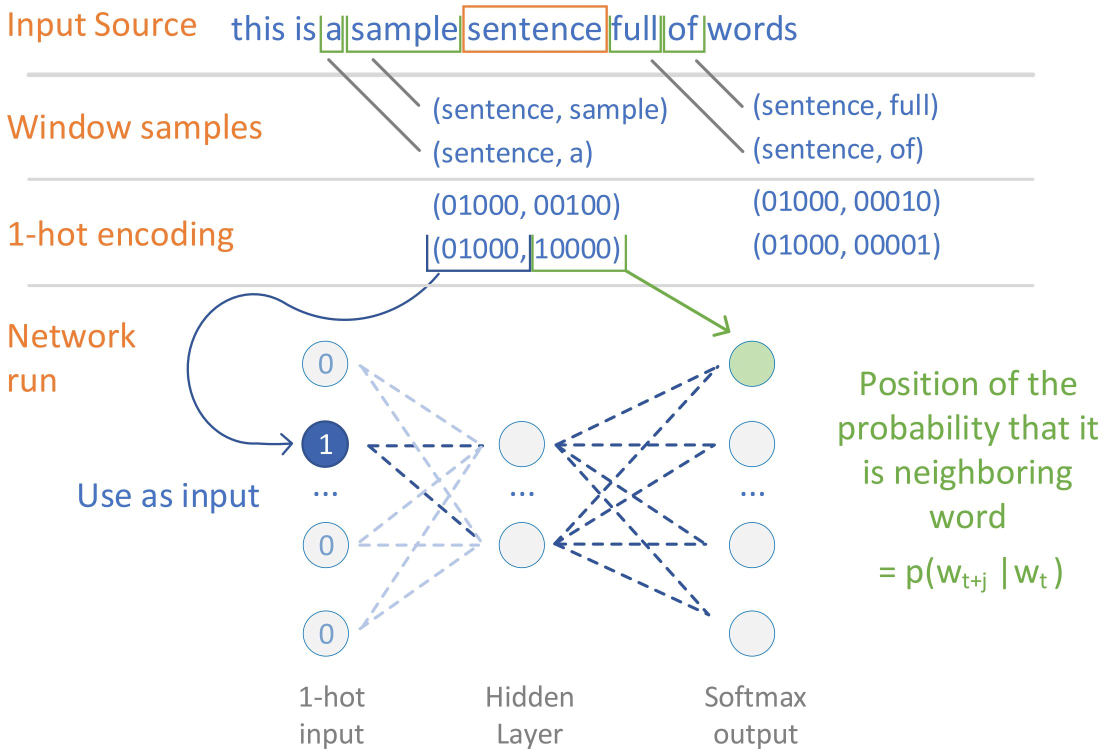
Word2Vec

- Two variants of word2vec
  - CBOW – predict word from context
  - Skip-gram – predict context from word
- Not the first “word vector” algorithm, but very influential
  - Fast to train – everyone can train on domain specific data from scratch
  - And of course cool marketing: word2vec sounds a lot better than LDA, SVD …
- Not state-of-the-art anymore, but good enough for practical use in many scenarios

More resources: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/
and http://jalammar.github.io/illustrated-word2vec/

Word2Vec - Training

- Train with a sliding window across our input text
  - That text sausage does not know about sentence or document boundaries (it does not matter)
- Compute negative log likelihood loss
  - But not over all terms in the vocabulary (too costly)
  - Do negative sampling of random terms



Input Source: this is a sample sentence full of words

Window samples: (sentence, sample) (sentence, a) (sentence, full) (sentence, of)

1-hot encoding: (01000, 00100) (01000, 10000) (01000, 00010) (01000, 00001)

Network run

Use as input

Position of the probability that it is neighboring word $= p(w_{t+j} \mid w_t)$

For detailed formulations & derivatives: Stanford NLP course https://www.youtube.com/watch?v=ERibwqs9p38

Offshoots based on the …2Vec Idea

- Unsupervised prediction of context in a sequence is a very generalizable idea
- Has been applied to:
  - Sentence / Paragraph / Document embeddings
  - Graph embeddings
  - Entity embeddings
  - Everything with sequences, such as playlists
- And of course endless small adjustments and improvements to the original word2vec word embedding task

More pointers at https://github.com/MaxwellRebo/awesome-2vec

FastText

- FastText improves on Word2Vec by using word-pieces (or char n-grams) as vocabulary
- Char n-gram vectors are added together to form a word
- No more out-of-vocabulary words or low-frequency words with almost random representations
- Very good for compound-heavy languages like German
- Offers similar performance (speed) & usability as word2vec
  - We can still generate 1 vector per term in our vocabulary
  - Analysis for word2vec can be applied here as well

Comes with a nice library and pre-trained models: https://github.com/facebookresearch/fastText

Similarity Measurements

- A word embedding is a vector space – in which we can do math
- A common operation is to measure the distance between two points
  - In a word embedding, measuring the distance is like measuring similarity of words
- Standard measure is the cosine similarity
  - Measures only the direction not the magnitude of the two points
  - Implemented as the dot-product of normalized vectors
  - Hard to visualize for a 300-dimensional space
    - Here, we can have hub-vectors that are close to many others, but the others don't necessarily have to be close to each other

Again a shout out to Gensim https://github.com/RaRe-Technologies/gensim

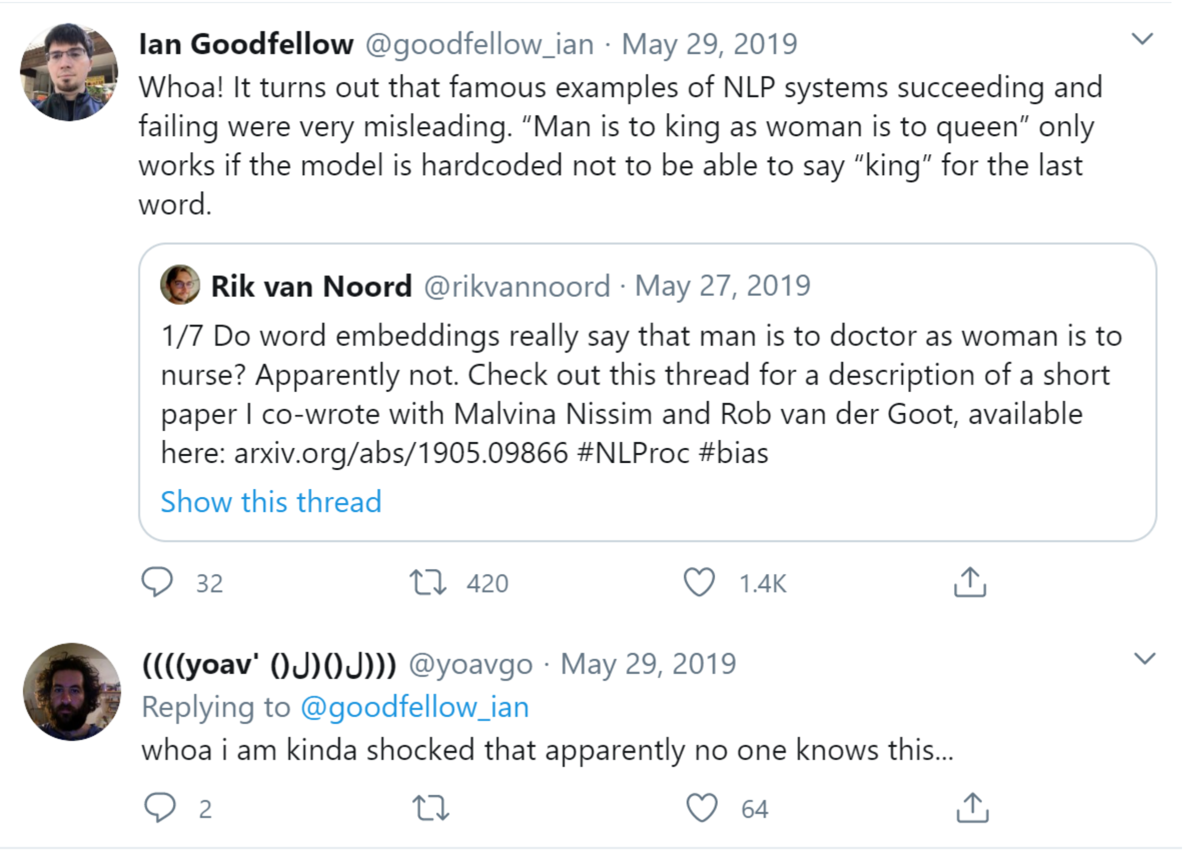
Analogies

- Probe relationships between more than 2 words
- A to B as C to D, can be reformulated as A – B + C = D
- Nice tool for PowerPoint presentations to non-tech audience
  - Easy to understand
  - Common example: King – Man + Woman = Queen
  - But also heads of state, capitals to countries …

- Beware, it seems they are an oversimplification:
  - Can be quite fragile between different runs (with different initializations, small changes to the algorithm)

Analogies – The Devil is in the Details

- May 2019: A surprise finding takes the Twittersphere by surprise:
  - Blindly re-using library code, that is crucial to your study is a bad idea
  - Vector similarity code of popular library hardcodes part of the analogy results:
  - King – man + woman = king ???



Ian Goodfellow @goodfellow_ian · May 29, 2019

Whoa! It turns out that famous examples of NLP systems succeeding and failing were very misleading. "Man is to king as woman is to queen" only works if the model is hardcoded not to be able to say "king" for the last word.

Rik van Noord @rikvannoord · May 27, 2019

1/7 Do word embeddings really say that man is to doctor as woman is to nurse? Apparently not. Check out this thread for a description of a short paper I co-wrote with Malvina Nissim and Rob van der Goot, available here: arxiv.org/abs/1905.09866 #NLProc #bias

Show this thread

(((yoav' ()J)()J))) @yoavgo · May 29, 2019

Replying to @goodfellow_ian

whoa i am kinda shocked that apparently no one knows this...

For more, here is the thread: https://twitter.com/goodfellow_ian/status/1133528189651677184

Limitations: Social Biases

- Taking biased text as input produces bias representations
- Bias can take many forms, such as gender or racial bias.
- Easily can affect downstream task, without anyone ever noticing:
  - Hiring decision
  - Predictive policing
  - Recommendation algorithms that marginalize minorities
- Various methods have been proposed to debias embeddings
  - Might not be truly effective and only cover up the bias

Gonen and Goldberg; Lipstick on a Pig: Debiasing Methods Cover up Systematic Gender Biases in Word Embeddings But do not Remove Them; NAACL 2019

Limitations: Word Ordering or N-Grams

“it was not good, it was actually quite bad”

== or !=

“it was not bad, it was actually quite good”

- The ordering & local context is important: “not good” vs. “not bad”
- Looking at N words at a time is called N-gram
- Creating bi-gram (2) or tri-gram (3) embeddings is not feasible
  - Sparsity problem
  - Not enough training data: no connection between “quite good” and “very good”

Limitations: Multiple senses per word

- Word2Vec, Glove & FastText always map 1 word to 1 vector
    - This is good for analysis purposes and constrained resource environments
- There is no contextualization in the vector after training
    - The vector of 1 word does not change based on other words in the sentence around it
- Words with multiple senses depending on context are squashed together in an average or most common sense in the training data
- But many words do have many senses based on the context
    - Missed opportunity for improved effectiveness

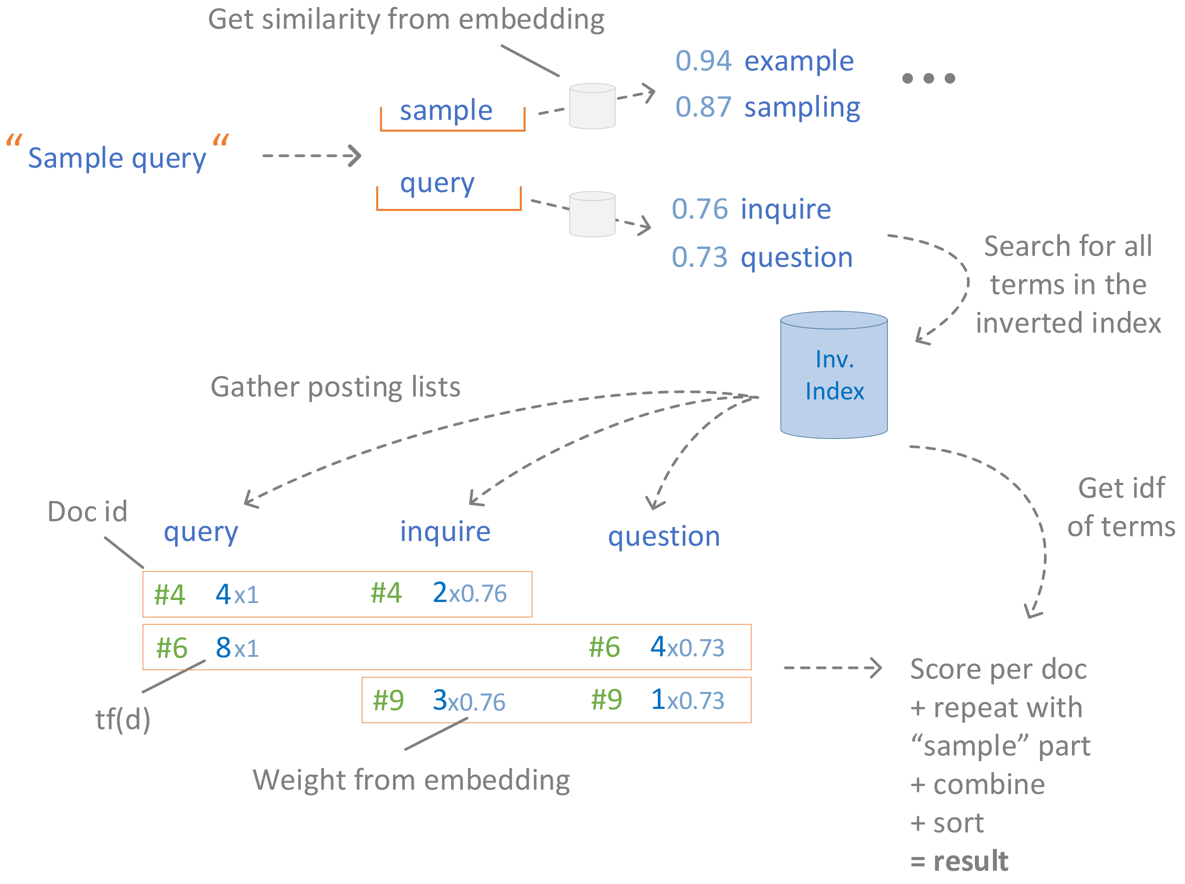
Word Embeddings in IR

A first glimpse …

Query Expansion with Word Embeddings

- Expand the search space of a search query with similar words
- Update collection statistics
- Adapted relevance model to score 1 document with multiple similar words together

N. Rekabsaz, M. Lupu, A. Hanbury, and G. Zuccon, "Generalizing Translation Models in the Probabilistic Relevance Framework," CIKM 2016
https://dl.acm.org/citation.cfm?id=2983833



"Sample query"

sample

query

Get similarity from embedding

0.94 example

0.87 sampling

…

0.76 inquire

0.73 question

Search for all terms in the inverted index

Inv. Index

Gather posting lists

Get idf of terms

Doc id

tf(d)

query	inquire	question
#4 4x1	#4 2x0.76	
#6 8x1		#6 4x0.73
	#9 3x0.76	#9 1x0.73

Weight from embedding

Score per doc
+ repeat with "sample" part
+ combine
+ sort
= result

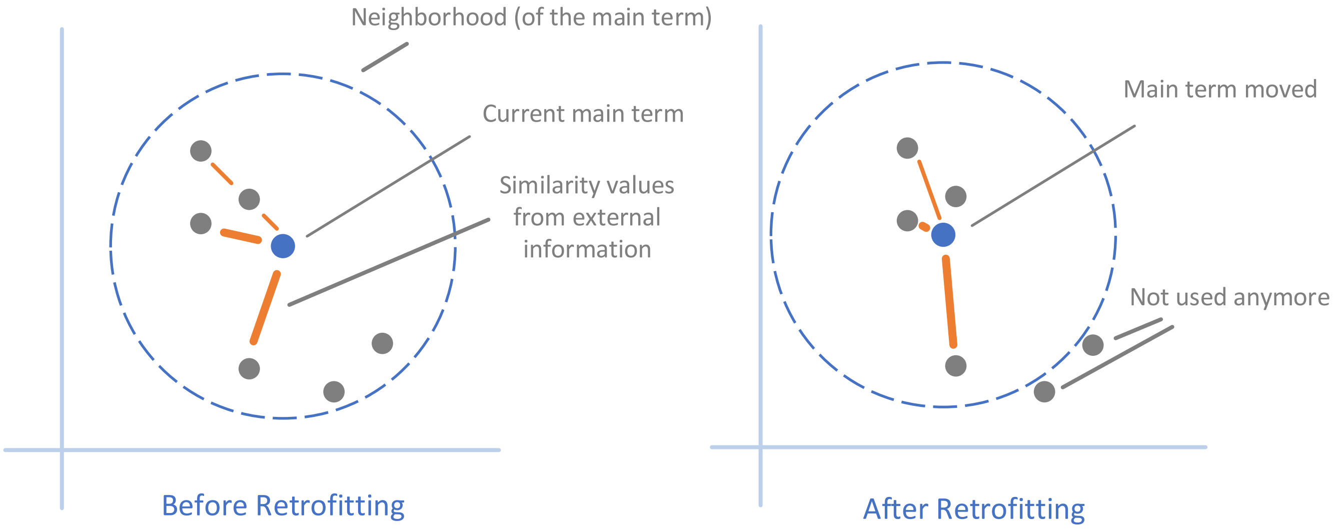
Problem: Topic Shifting

- Word Embeddings trained on Wikipedia context
- Words with similar context close together
  - Can be a different topic -> bad for IR

- Word “Austria” has close neighbors that correspond to physical neighboring countries
  - Germany, Hungary, …
  - Searching for “Hotels in Austria” -> you probably only want results located in “Austria” not the 25 best hotels in Germany

Retrofitting

- Incorporate external resources into an existing word embedding
- Moves the vectors of the existing word embedding
- Iterative update function



Retrofitting Word Vectors to Semantic Lexicons.
Faruqui M, Dodge J, Jauhar SK, Dyer C, Hovy E. NAACL 2015.

Topic Shifting – Examples from Wikipedia

austria

Original Skip-gram	
austrian	0.78
hungari	0.73
vienna	0.72
germani	0.7

LSI	
austrian	0.91

Retrofitted	
austrian	0.95
vienna	0.79
graz	0.74
wien	0.73

austria**n**

Original Skip-gram	
austria	0.78
vienna	0.73
austro	0.71
graz	0.6

LSI	
austria	0.91
habsburg	0.84
cisleithanian	0.83

Retrofitted	
austria	0.95
vienna	0.8
austro	0.78
viennes	0.76
habsburg	0.75
cisleithanian	0.74

Application to Patent Retrieval

- In patent retrieval the recall is super important
    - Single relevant patent missed can be bad (so I am told)
- Vocabulary mismatch is a tough problem, as there are many domain specific terms
- We expanded queries with retrofitted word embeddings
    - Skip-gram alone did not improve results
    - Combination of local skip-gram, global LSI via retrofitting

- Improved Recall significantly in a standard patent test collection

Enriching Word Embeddings for Patent Retrieval with Global Context
S. Hofstätter, N. Rekabsaz, M. Lupu, C. Eickhoff, A. Hanbury, ECIR 2019

Outlook

- Retrofitting was so 2018 & had many limitations
  - Unsupervised method
  - Limited effectiveness

- Ingredients for better approaches:
  - Supervised learning with large retrieval datasets
  - Way more parameters to learn
  - Contextualization
  - Operate on full text of query and documents

More on this in the coming lectures

Summary: Word Embeddings

1. Represent words as vectors instead of characters

2. Unsupervised pre-training is a major strength of word embeddings

3. Many potential applications in IR, such as query expansion

1. Represent words as vectors instead of characters
2. Unsupervised pre-training is a major strength of word embeddings
3. Many potential applications in IR, such as query expansion

Thank You