Unsupervised NLP

Vsevolod Dyomkin
prj-nlp-2020

Contents

1. Overview
2. Word Embeddings
3. Document Embeddings
4. Topic Modeling
5. Visualization

0. Overview of Unsupervised NLP

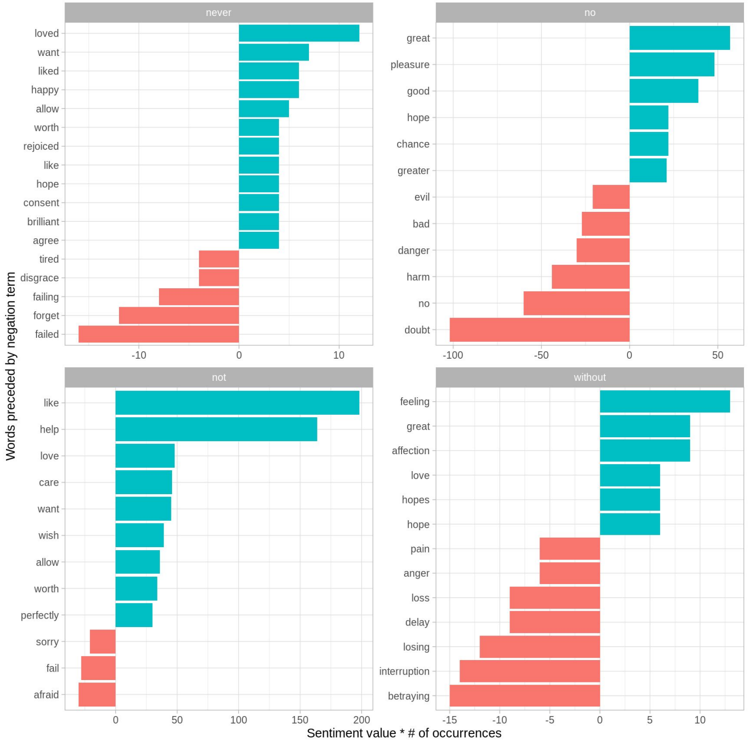
Potential:

- lots of unlabeled data
    - or data with weak supervision
- not everything may be labeled
- examples:
    - web
    - books
    - parallel corpora for MT

Major Unsupervised Approaches

- counting
- matrix factorization
- expectation maximization
- clustering

Counting Example: Ngrams



1. Word Similarity

Question 1: How words are related?

Question 2: how to measure word similarity?

1. Word Relations

Question 1: How words are related?

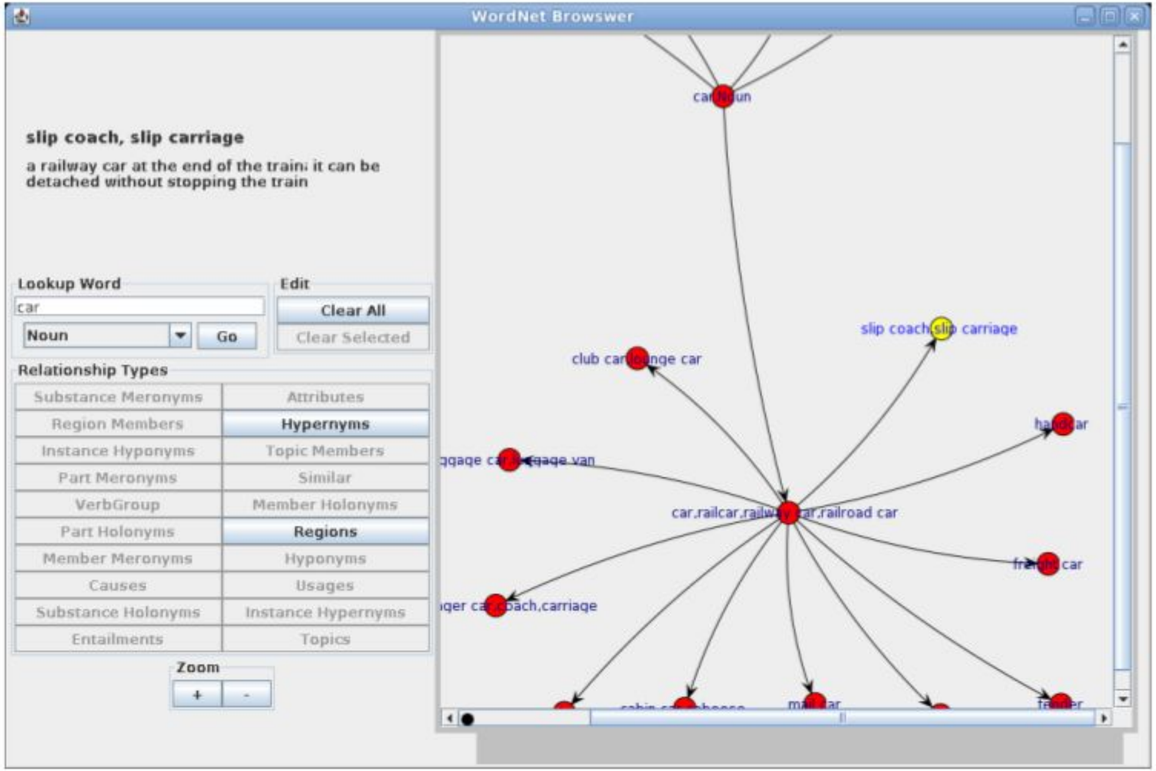
Question 2: how to measure word similarity?

Many faces of similarity:

- dog -- cat
- dog -- poodle
- dog -- animal
- dog -- bark
- dog -- leash

- dog -- chair — same POS
- dog -- dig — edit distance
- dog -- god — same letters
- dog -- fog — rhyme
- dog -- 6op — shape

Graph-based Approach



Wordnet Similarity Measures

$$Sim(C1, C2) = 2 \bullet Max(C1, C2) - SP$$

$$Sim_{Rod}(C1^p, C2^q) = W_w S_w(C1^p, C2^q) + W_u S_u(C1^p, C2^q) + W_n S_n(C1^p, C2^q)\ )$$

$$Sim_{Resnik}(C1, C2) = \frac{2 \bullet \ln\left((p_{mis}(C1,C2)\right)}{\ln(p(c1)) + \ln\left(p(c2)\right)}$$

$$Sim_{Knappe}(C1, C2) = p * \frac{|Ans(C1) \cap Ans(C2)|}{|Ans(C1)|} + (1-p) * \frac{|Ans(C1) \cap Ans(C2)|}{|Ans(C2)|}$$

$$Sim_{Zhou}(C1, C2) = 1 - k\left(\frac{\ln(len(C1,C2)+1}{\ln(2 \bullet (deep_{max}-1))}\right) - (1-k) * ((IC(C1) + IC(C2) - 2 * IC(lso(C1, C2))/2)$$

$$Sim_{Resnik}(C1, C2) = -ln(p_{mis}(C1,C2))$$

$$Sim_{tvsk}(C1, C2) = \frac{|C1 \cap C2|}{|C1 \cap C2| + \propto |C1 - C2| + (\propto - 1)|C2 - C1|}$$

$$Sim_{LC}(C1, C2) = -\log\left(\frac{length}{2.D}\right)$$

$$Sim_{HSO}(C1, C2) = C - SP - k \bullet d$$

$$Sim_{wup}(C1, C2) = \frac{2 \bullet N}{N1 + N2 + 2 \bullet N}$$

Wordnet Similarity Measures

$$Sim(C1,C2) = 2 \cdot Max(C1,C2) - SP$$

$$Sim_{Rod}(C1^p,C2^q) = W_w S_w(C1^p,C2^q) + W_u S_u(C1^p,C2^q) + W_n S_n(C1^p,C2^q)\ )$$

$$Sim_{Resnik}(C1,C2) = \frac{2 \cdot \ln((p_{mis}(C1,C2))}{\ln(p(c1)) + \ln(p(c2))}$$

$$Sim_{Knappe}(C \ldots \frac{|Ans(C1) \cap Ans(C2)|}{} \ldots (1 \ldots p) * \frac{|Ans(C1) \cap Ans(C2)|}{|Ans(C2)|}$$

Don’t work :(

$$Sim_{Zhou}(C1,C2) = 1 - k \ldots \ln(2 \cdot (deep_{max} - 1))) \ldots (1-k) * ((IC(C1) + IC(C2) - 2 * IC(lso(C1,C2))/2)$$

$$Sim_{Resnik}(C1,C2) = -ln(p_{mis}(C1,C2))$$

$$Sim_{tvsk}(C1,C2) = \frac{|C1 \cap C2|}{|C1 \cap C2| + \alpha|C1 - C2| + (\alpha - 1)|C2 - C1|}$$

$$Sim_{LC}(C1,C2) = -\log\left(\frac{length}{2.D}\right)$$

$$Sim_{HSO}(C1,C2) = C - SP - k \cdot d$$

$$Sim_{wup}(C1,C2) = \frac{2 \cdot N}{N1 + N2 + 2 \cdot N}$$

https://arxiv.org/pdf/1310.8059.pdf

Distributional Hypothesis

You shall know a word by the company it keeps.

John Rupert Firth, 1957

Co-occurrence Matrix

- Counting FTW :)

- I like deep learning.
- I like NLP.
- I enjoy flying.

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

Co-occurrence Matrix

- An explicit word representation
- Number of nonzero dimensions (in one experiment):
    - max: 474234
    - min: 3
    - mean: 1595
    - median: 415

Co-occurrence Matrix Issues

- Sparse
- Non-normalized
- Spurious relations (noise)

Pointwise Mutual Information

Dan Jurafsky

$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i*}p_{*j}}$$

p(w,context)

	computer	data	pinch	result	sugar	p(w)
apricot	0.00	0.00	0.05	0.00	0.05	0.11
pineapple	0.00	0.00	0.05	0.00	0.05	0.11
digital	0.11	0.05	0.00	0.05	0.00	0.21
information	0.05	0.32	0.00	0.21	0.00	0.58
p(context)	0.16	0.37	0.11	0.26	0.11	

- pmi(information,data) = $\log_2$ (.32 / (.37*.58)) = .58

(.57 using full precision)

PPMI(w,context)

	computer	data	pinch	result	sugar
apricot	-	-	2.25	-	2.25
pineapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.57	-	0.47	-

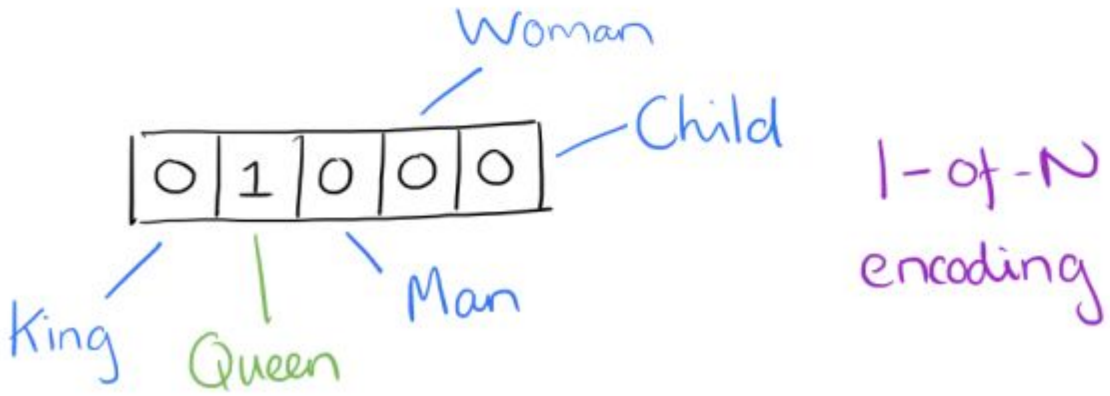
Positive PMI (PPMI)

PPMI = max(PMI, 0)

Negative PMI carries no useful information.

Word Representations

- 1-hot (BoW-style)



Word Representations

- 1-hot
- Feature-template based

  “Queen” -> [capitalized, NN, singular, nominal, feminitive, ...]

Word Representations

- 1-hot
- Feature-template based
- Distributed

Word Vectors

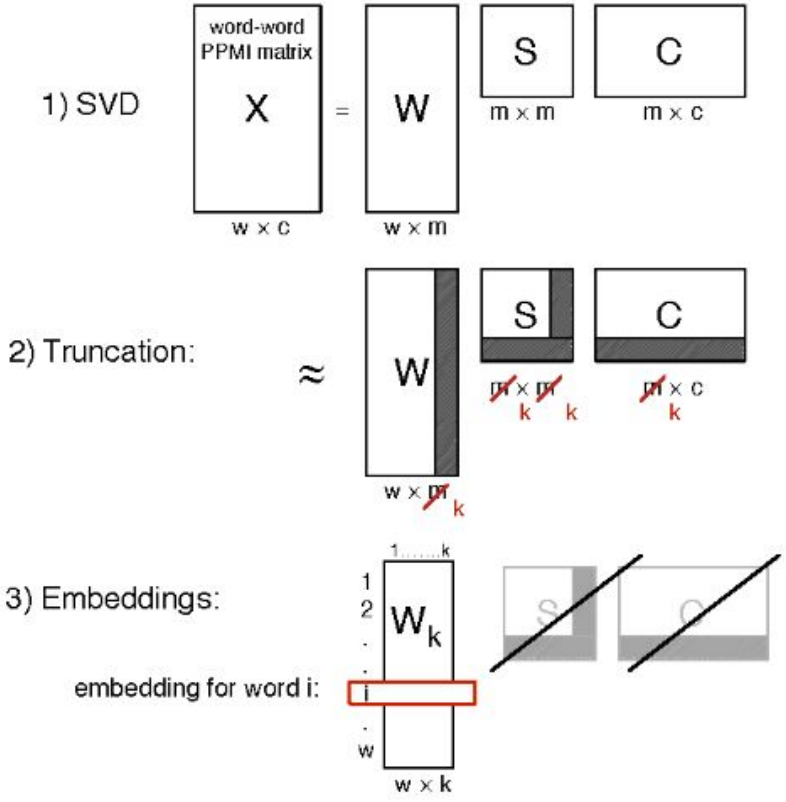
Another name for Distributed Word Representations

- Explicit (sparse)
- Dense

Dense Word Vectors with SVD

Matrix Factorization

to the rescue



Dense Word Vectors with NNSE

Non-Negative Sparse Embedding - an alternative approach to matrix factorization

- using non-negative matrix factorization
- and sparse coding

$$C(\mathbf{A}, \mathbf{S}) = \frac{1}{2}\|\mathbf{X} - \mathbf{AS}\|^2 + \lambda \sum_{ij} f(S_{ij}), \quad (1)$$

where the squared matrix norm is simply the summed squared value of the elements, i.e. $\|\mathbf{X}-\mathbf{AS}\|^2 = \sum_{ij}[\mathbf{X}_{ij}-(\mathbf{AS})_{ij}]^2$. The tradeoff between sparseness and accurate reconstruction is controlled by the parameter λ, whereas the form of f defines how sparseness is measured. To achieve a sparse code, the form of f must be chosen correctly: A typical choice is $f(s) = |s|$, although often similar functions that exhibit smoother behaviour at zero are chosen for numerical stability.

http://talukdar.net/papers/nnse_coling12.pdf

NNSE Sparsity

	SVD_{300}	$NNSE_{50}$	$NNSE_{300}$	$NNSE_{1000}$
Sparsity level (% of zeros)	0	81.94	90.39	99.95
Average number of words per dimension	35560.0	6422.4	3418.5	1818.2
Average number of dimensions per word	300.0	9.0	28.8	51.1

NMF – Why does it work?

The reason why NMF has become so popular is because of its ability to automatically extract sparse and easily interpretable factors.

Also grounded in fMRI research.

Compositional NNSE (CNNSE)

Training with additional constraints to impose compositionality for certain relations (e.g. noun-adjective)

http://www.aclweb.org/anthology/N15-1004

2. word2vec

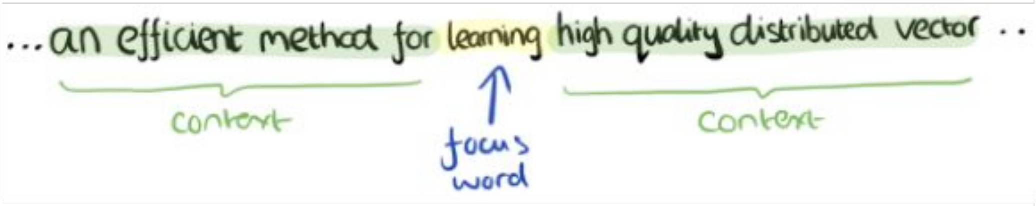
Train a Neural network model to predict the word in context

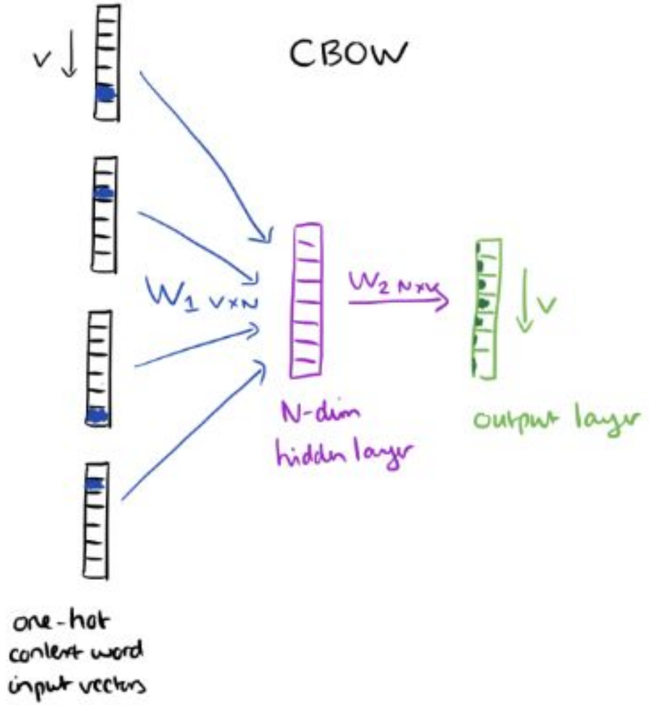
The result is, in theory, equivalent to SVD of the PPMI matrix

But with a more nuanced implementation, which is, basically, an instance of **Expectation Maximization**

word2vec - CBOW

Continuous bag-of-words (CBOW)

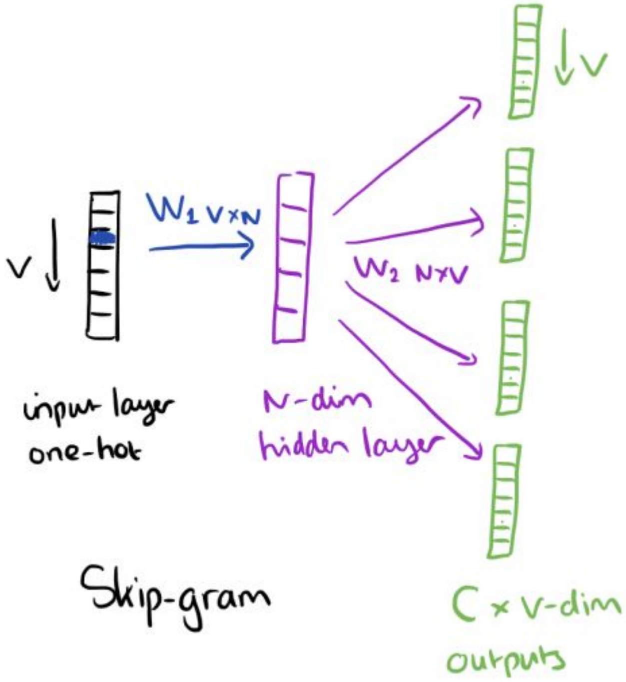




http://u.cs.biu.ac.il/~yogo/cvsc2015.pdf

word2vec - SGNS

Skip-gram negativa sampling



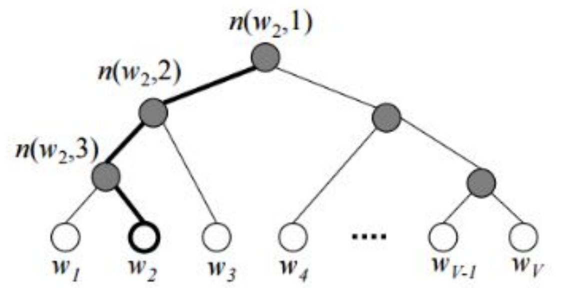
word2vec Loss Functions

- Softmax
  - Hard to compute

$$p(w_O|w_I) = \frac{\exp({v'_{w_O}}^\top v_{w_I})}{\sum_{i=1}^{V} \exp({v'_{w_i}}^\top v_{w_I})}$$

word2vec Loss Functions

- Softmax
- Hierarchical Softmax
    - Use a binary tree to get to the word probability



$$p(w_O|w_I) = \prod_{k=1}^{L(w_O)} \sigma(\mathbb{I}_{\text{turn}}(n(w_O,k), n(w_O,k+1)) \cdot {v'_{n(w_O,k)}}^\top v_{w_I})$$

word2vec Loss Functions

- Softmax
- Hierarchical Softmax
- Cross Entropy
  - Loss function:

$$\mathcal{L}_\theta = -\sum_{i=1}^{V} y_i \log p(w_i|w_I) = -\log p(w_O|w_I)$$

$$\mathcal{L}_\theta = -\log \frac{\exp({v'_{w_O}}^\top v_{w_I})}{\sum_{i=1}^{V} \exp({v'_{w_i}}^\top v_{w_I})} = -{v'_{w_O}}^\top v_{w_I} + \log \sum_{i=1}^{V} \exp({v'_{w_i}}^\top v_{w_I})$$

word2vec Loss Functions

- Softmax
- Hierarchical Softmax
- Cross Entropy
- Noise-contrastive estimation
  - differentiate the target word from noise samples using a logistic regression classifier
  - the probability of a noise word in logarithm is reversely proportional to its rank

$$\mathcal{L}_\theta = -[\log \frac{\exp({v'_w}^\top v_{w_I})}{\exp({v'_w}^\top v_{w_I}) + Nq(\tilde{w})} + \sum_{\substack{i=1 \\ \tilde{w}_i \sim Q}}^{N} \log \frac{Nq(\tilde{w}_i)}{\exp({v'_w}^\top v_{w_I}) + Nq(\tilde{w}_i)}]$$

word2vec Loss Functions

- Softmax
- Hierarchical Softmax
- Cross Entropy
- Noise-contrastive estimation
- Negative sampling
  - simplified variation of NCE

$$\mathcal{L}_\theta = -\left[\log \sigma({v'_w}^\top v_{w_I}) + \sum_{\substack{i=1 \\ \tilde{w}_i \sim Q}}^{N} \log \sigma(-{v'_{\tilde{w}_i}}^\top v_{w_I})\right]$$

word2vec Training Tricks

- Normalization
- Soft sliding window:
  - assign less weight to more distant words
  - the actual window size is randomly sampled
- Subsampling frequent words:
  - discard words w with probability (1–t)/f(w)
- Learn phrases (collocations) first

GloVe

- aka "Global" Vectors
- combine the count-based matrix factorization and the context-based skip-gram model together

$$\mathcal{L}_\theta = \sum_{i=1,j=1}^{V} f(C(w_i, w_j))(w_i{}^\top \tilde{w}_j + b_i + \tilde{b}_j - \log C(w_i, \tilde{w}_j))^2$$

- The weighting schema f(c) is a function of the co-occurrence of wi and wj and it is an adjustable model configuration:

$$f(c) = \begin{cases} (\frac{c}{c_{\max}})^\alpha & \text{if } c < c_{\max}, c_{\max} \text{ is adjustable.} \\ 1 & \text{if otherwise} \end{cases}$$

GloVe

- aka "Global" Vectors
- combine the count-based matrix factorization and the context-based skip-gram model together

$$\mathcal{L}_\theta = \sum_{i=1,j=1}^{V} f(C(w_i, w_j))(w_i^\top \tilde{w}_j + b_i + \tilde{b}_j - \log C(w_i, \tilde{w}_j))^2$$

- The weighting schema f(c) is a function of the co-occurrence of wi and wj and it is an adjustable model configuration:

$$f(c) = \begin{cases} (\frac{c}{c_{\max}})^\alpha & \text{if } c < c_{\max},\ c_{\max} \text{ is adjustable.} \\ 1 & \text{if otherwise} \end{cases}$$

word2vec problems

- OOVs
- Polysemy
- Non-explicit dimensions
- Noise for low-frequency words

fasttext & LexVec

Extension to SGNS to take into account subword information (character ngrams):

The word “where” is represented as an “embedding bag” - a sum of representations of the words “<where>”, “<wh”, “whe”, “her”, “ere”, “re>”, “<whe”, “wher”, “here”, “ere>”, “<wher”, “where”, “here>”, “<where”, “where>”

https://arxiv.org/pdf/1607.04606.pdf

word2gauss

Each word is represented as a multivariate Gaussian:

a probability P[i] – a K-dimensional Gaussian parameterized by

mean mu and covariance matrix Sigma:

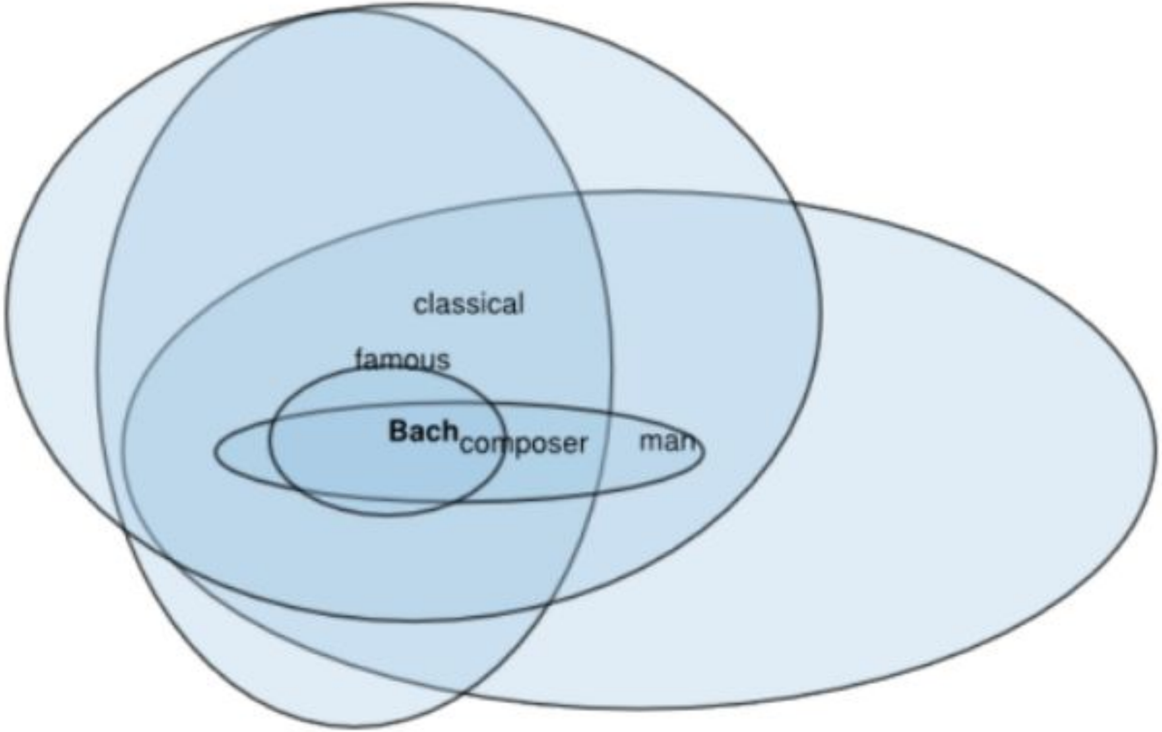
P[i] ~ N(x; mu[i], Sigma[i])

The mean is a vector of length K and in the most general case Sigma[i] is a (K, K) matrix. 2 approximations to simplify Sigma:

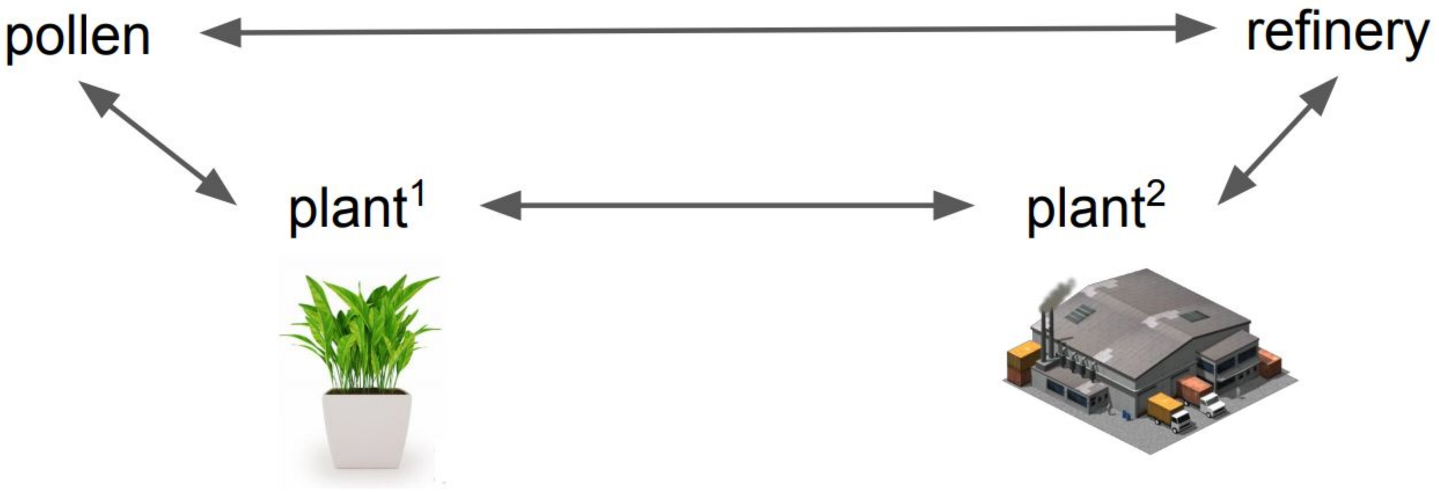
- diagonal - a vector length K
- spherical - a float

https://arxiv.org/pdf/1412.6623.pdf

word2gauss

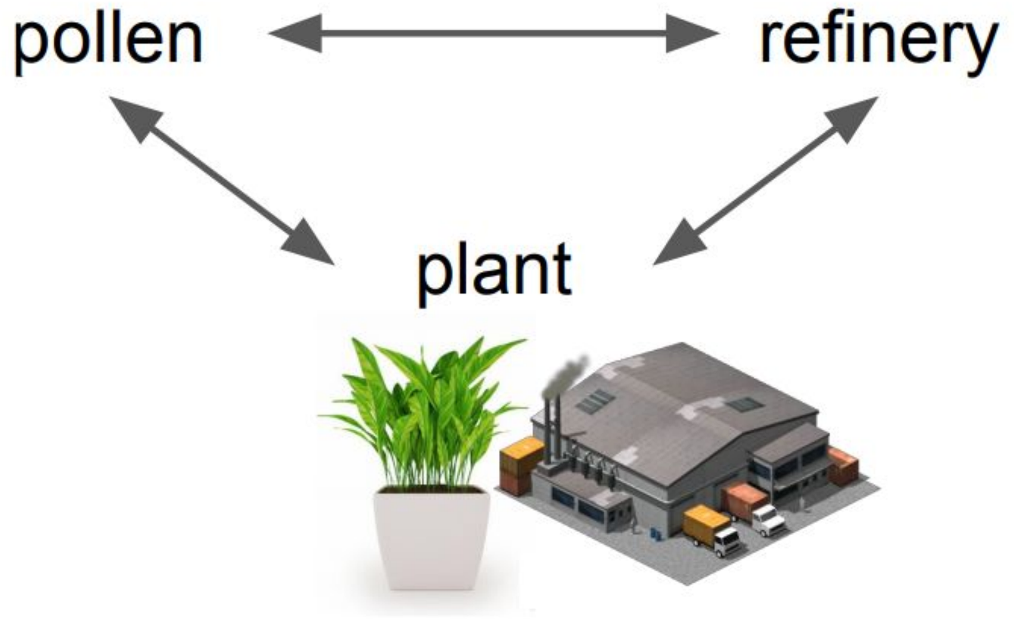


3. Sense Embeddings



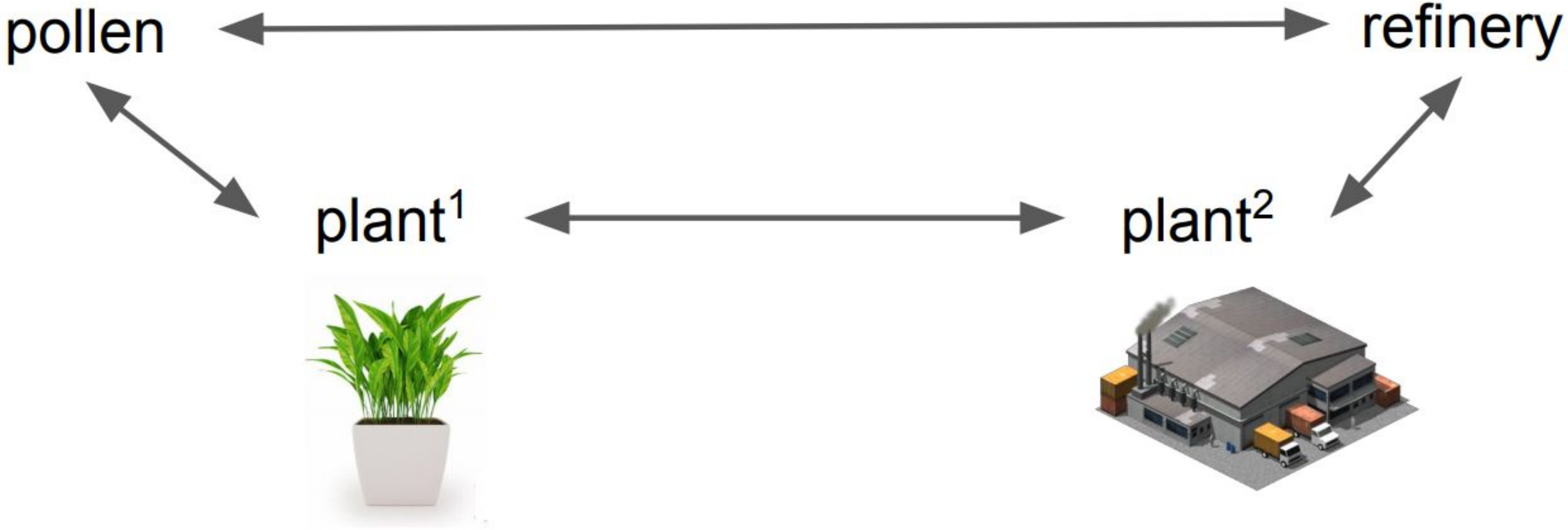
What does it mean for NLP?

Triangle inequality in word embeddings.



What does it mean for NLP?

Word embeddings => sense embeddings



How to get sense embeddings?

- Train on a sense-annotated corpus
    - *also: add related words from the KB to the context vector*
- Derive from word embeddings using ontologies

SensEmbed vectors

- use *BabelNet* as a sense inventory
- use a dump of the *English Wikipedia* for corpus
- do disambiguation with *Babelfy*
- train *word2vec CBOW* to obtain sense embeddings
  (window = 5, size = 400)

SensEmbed vectors

$\textit{bank}_1^n$ (geographical)	$\textit{bank}_2^n$ (financial)	$\textit{number}_4^n$ (phone)	$\textit{number}_3^n$ (acting)
upstream_1^r	$\text{commercial\_bank}_1^n$	calls_1^n	appearing_6^v
downstream_1^r	$\text{financial\_institution}_1^n$	dialled_1^v	minor_roles_1^n
runs_6^v	national_bank_1^n	operator_{20}^n	$\text{stage\_production}_1^n$
confluence_1^n	trust_company_1^n	$\text{telephone\_network}_1^n$	$\text{supporting\_roles}_1^n$
river_1^n	savings_bank_1^n	telephony_1^n	leading_roles_1^n
stream_1^n	banking_1^n	subscriber_2^n	stage_shows_1^n

Example from Iacobacci et al. (2015)

Nasari vectors

Bank (financial institution)		
English	**French**	**Spanish**
bank	banque	banco
banking	bancaire	bancario
deposit	crédit	banca
credit	financier	financiero
money	postal	préstamo
loan	client	entidad
commercial_bank	dépôt	depósito
central_bank	billet	crédito

Bank (geography)		
English	**French**	**Spanish**
river	eau	banco
stream	castor	limnología
bank	berge	ecología
riparian	canal	barrera
creek	barrage	estuarios
flow	zone	isla
water	perchlorate	interés
watershed	humide	laguna

Example from Camacho-Collados (2016)

How to get sense embeddings?

- Train on a sense-annotated corpus
  - *also: add related words from the KB to the context vector*
- Derive from word embeddings using ontologies

Retrofitting

```python
''' Retrofit word vectors to a lexicon '''
def retrofit(wordVecs, lexicon, numIters):
  newWordVecs = deepcopy(wordVecs)
  wvVocab = set(newWordVecs.keys())
  loopVocab = wvVocab.intersection(set(lexicon.keys()))
  for it in range(numIters):
    # loop through every node also in ontology (else just use data estimate)
    for word in loopVocab:
      wordNeighbours = set(lexicon[word]).intersection(wvVocab)
      numNeighbours = len(wordNeighbours)
      #no neighbours, pass - use data estimate
      if numNeighbours == 0:
        continue
      # the weight of the data estimate if the number of neighbours
      newVec = numNeighbours * wordVecs[word]
      # loop over neighbours and add to new vector (currently with weight 1)
      for ppWord in wordNeighbours:
        newVec += newWordVecs[ppWord]
      newWordVecs[word] = newVec/(2*numNeighbours)
  return newWordVecs
```

Retrofit - pull the words closer to their relations in the KB.

Example from Faruqui et al. (2015)

ConceptNet Numberbatch

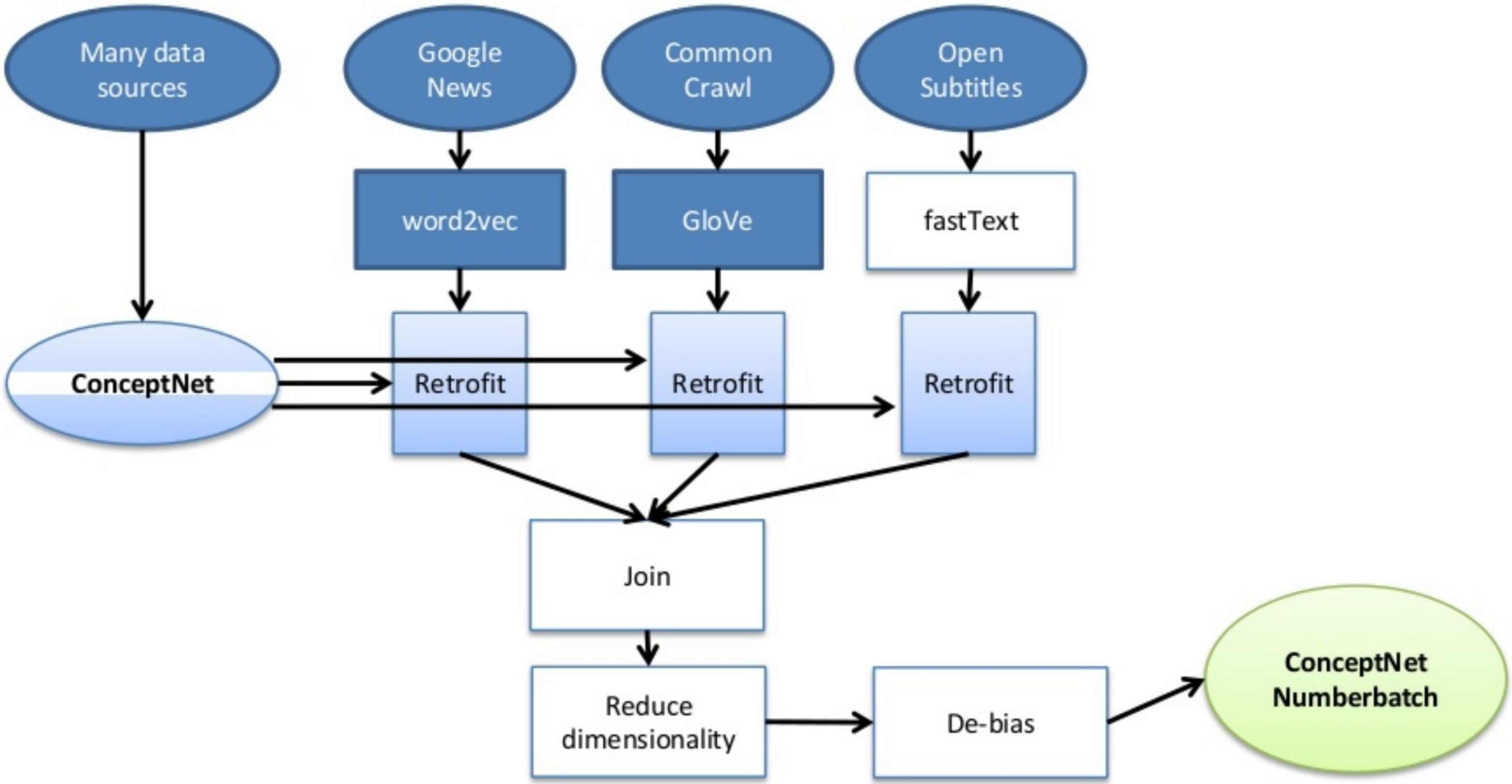
The current SOTA vectors due to:

- vector ensemble using ConceptNet to merge vectors
- OOV handling

https://blog.conceptnet.io/2016/05/25/conceptnet-numberbatch-a-new-name-for-the-best-wOrd-embeddings-you-can-download/

https://blog.conceptnet.io/2017/03/02/how-luminoso-made-conceptnet-into-the-best-word-vectors-and-won-at-semeval/

ConceptNet Numberbatch

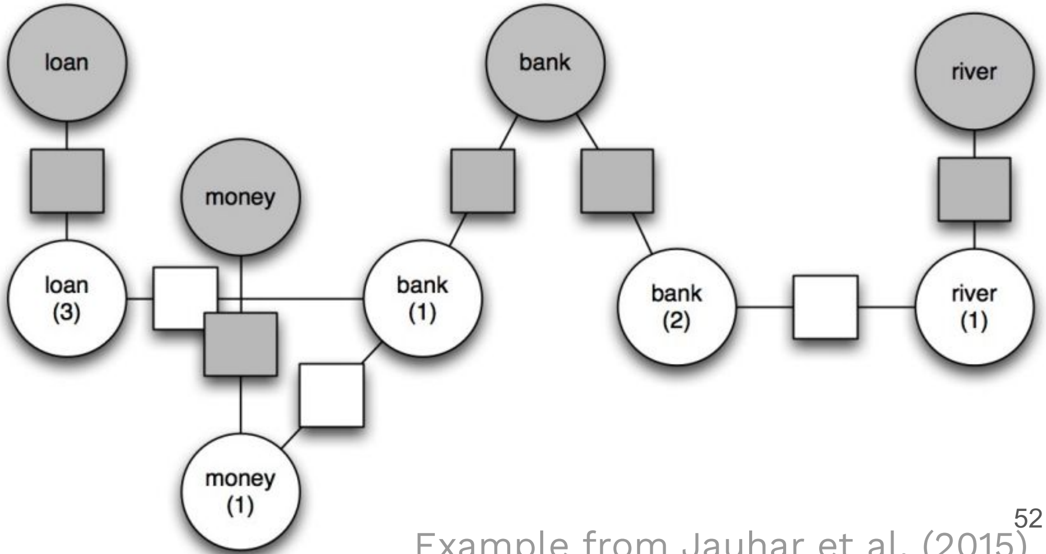


Sense vectors by retrofitting

- Assign each sense the vector of the word.
- Pull vectors of senses closer to vectors of words they relate to in the KB.

Observed sense-agnostic word vectors

Latent sense-specific sense vectors



Example from Jauhar et al. (2015)

4. Document Embeddings

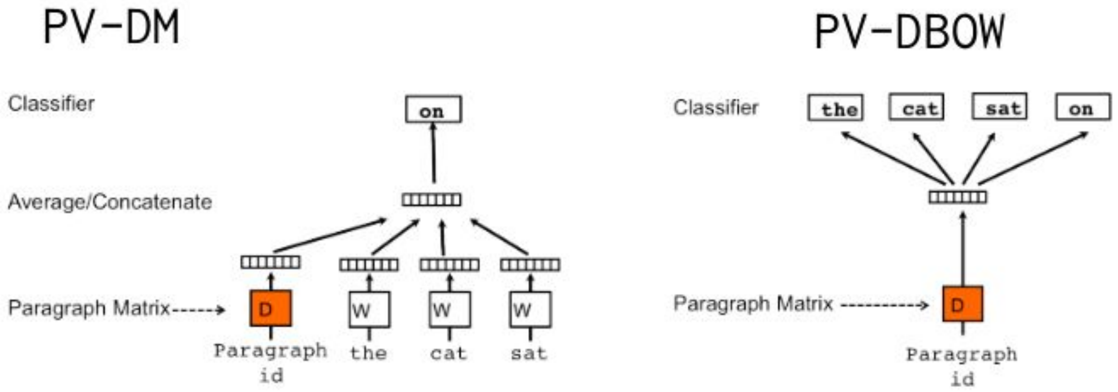
Question: how to represent phrases/sentences/paragraphs/documents with dense vectors?

Default answer: average the word vectors (aka “embedding bag”)

doc2vec

Alternative: paragraph vectors - apply the same approaches as with word2vec to paragraphs.

Caveat: paragraphs are unique and will not repeat



https://cs.stanford.edu/~quocle/paragraph_vector.pdf

Other Document Embeddings

- Skip-thoughts
- Universal Sentence Encoder
- ElMo

5. Graph Embeddings

- graph2vec
- DeepWalk (https://arxiv.org/pdf/1403.6652.pdf)

Dense Representations Recap

Key idea: transition from sparse (BoW) to dense vectors and maximize the vectors' affinity to some relation in the process.

Pros:

- capture those relations
- easier to compute with (possible to use as input for neural nets)

Cons:

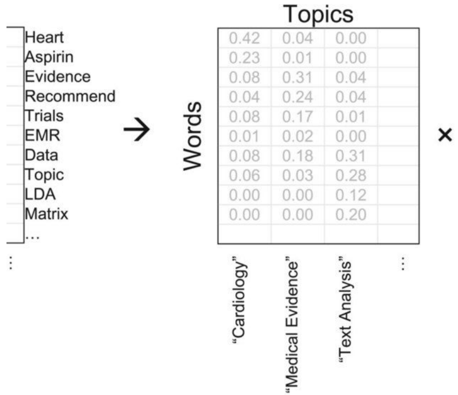
- expensive to compute the vectors themselves
- knowledge transfer?

6. Topic Modelling

May be framed as a multi-class whole-text classification/ranking problem.

But is mostly an unsupervised problem: the topics are not known beforehand.

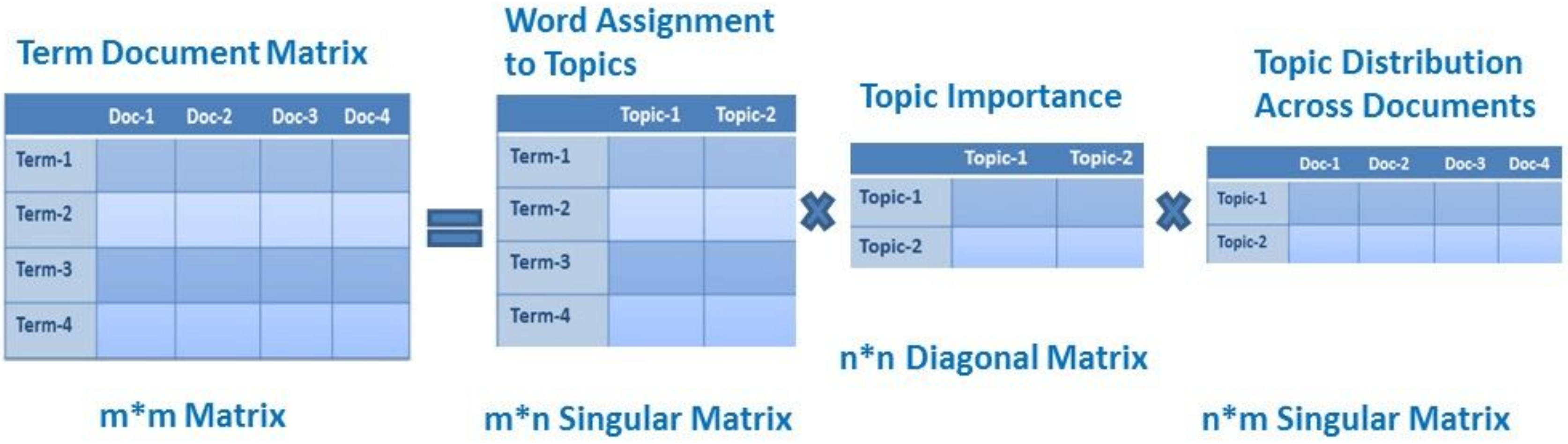
Same approaches as to word embeddings apply...



Latent Semantic Indexing

Factorization of the word-document matrix using SVD

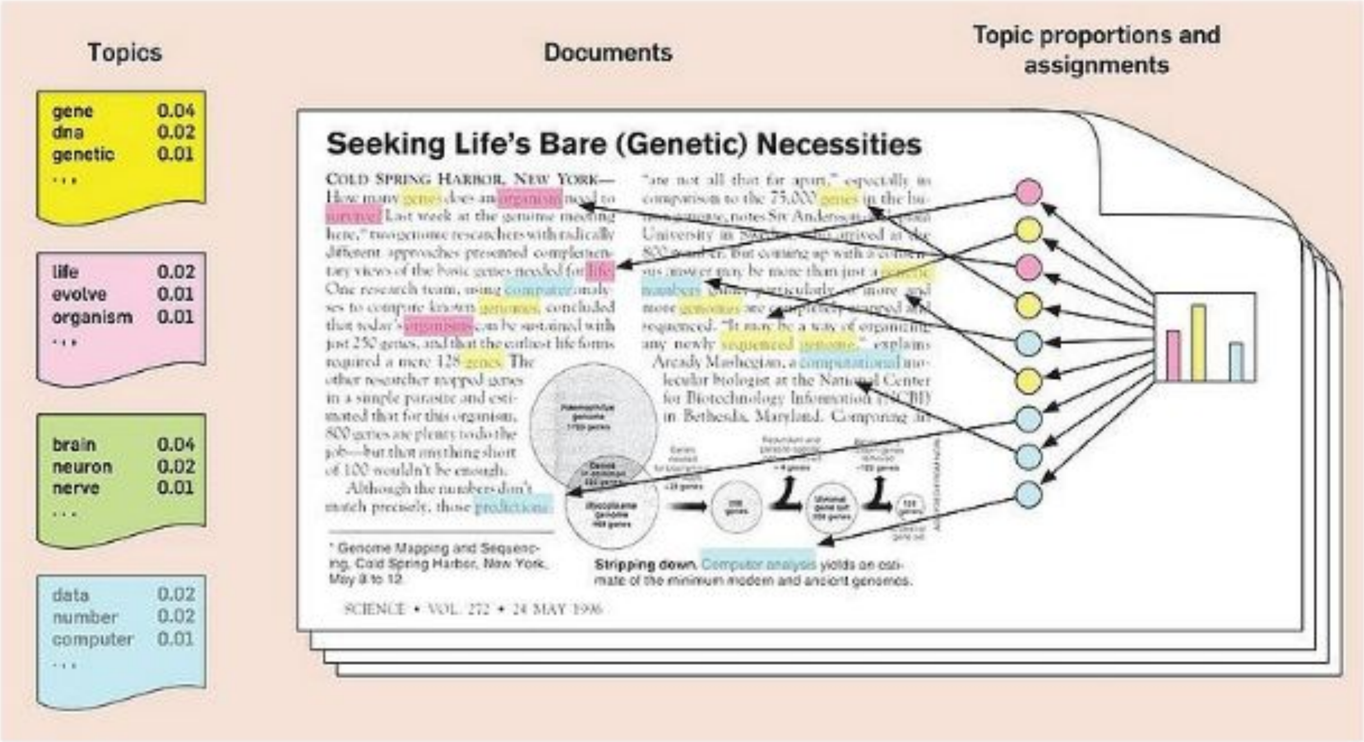
and selecting the top-N eigen values.



Latent Dirichlet Allocation

In LDA, each document may be viewed as a mixture of various topics. This is identical to probabilistic latent semantic analysis (pLSA), except that in LDA the topic distribution is assumed to have a sparse Dirichlet prior. The sparse Dirichlet priors encode the intuition that documents cover only a small set of topics and that topics use only a small set of words frequently. In practice, this results in a better disambiguation of words and a more precise assignment of documents to topics. LDA is a generalization of the pLSA model, which is equivalent to LDA under a uniform Dirichlet prior distribution.

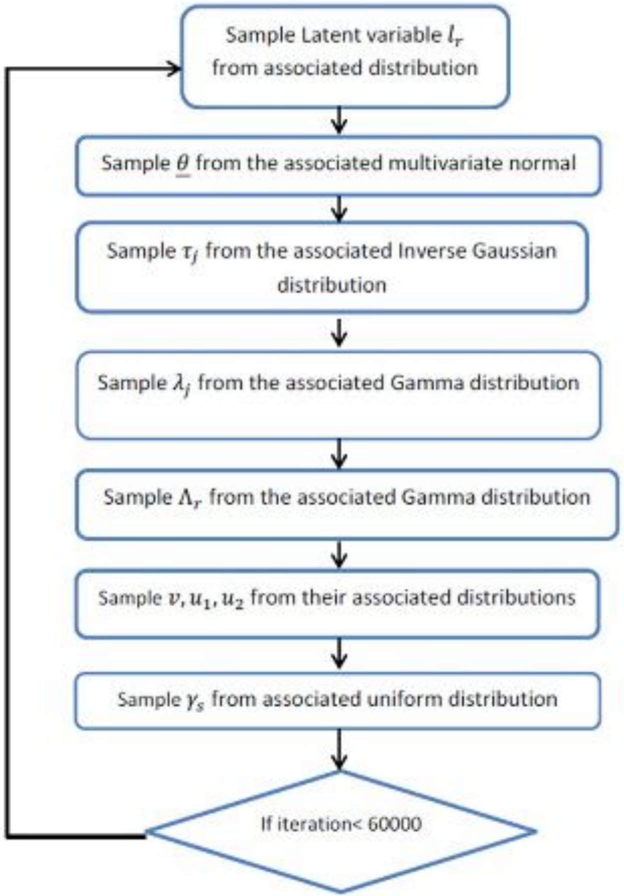
Latent Dirichlet Allocation



Gibbs Sampling

A computationally-heavy procedure used to estimate LDA models.

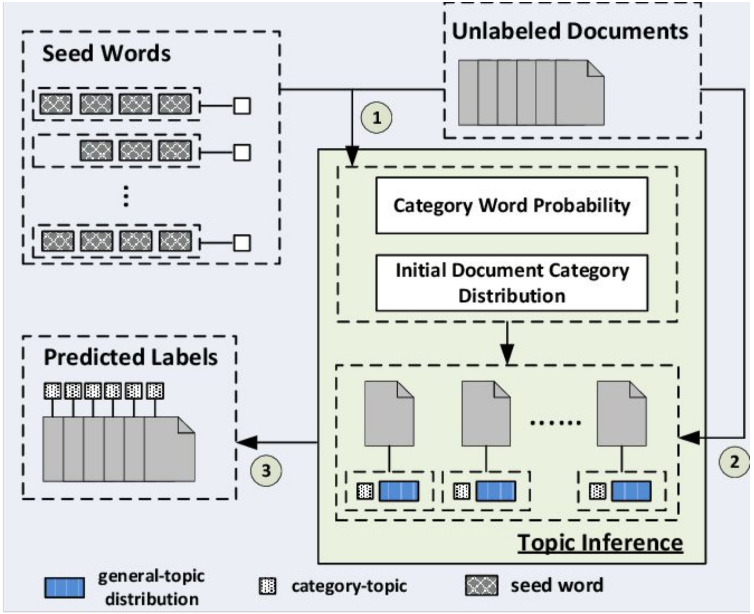
https://stats.stackexchange.com/questions/10213/can-someone-explain-gibbs-sampling-in-very-simple-words



Guided LDA

Common problem with LDA: some “expected” topics are overlapping and some are not present at all (due to the uneven distribution of documents).

Simple idea: assign topics to some “seed” words



Anchor Words

Problem of LSI/LDA: hard to interpret topics.

Recall an alternative factorization to SVD: NMF

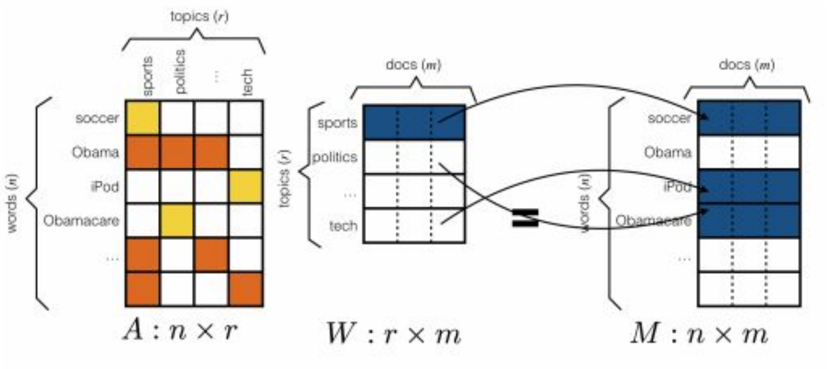
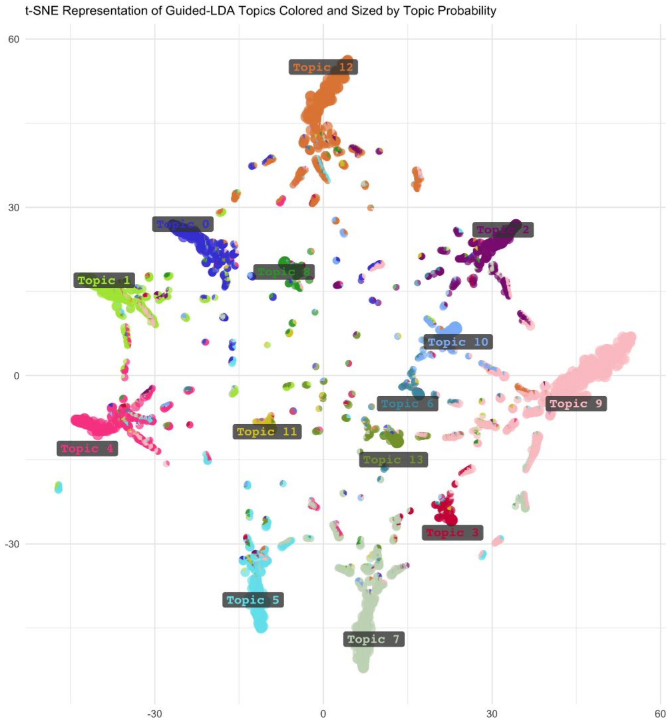


Figure 2: Consequences of the anchor word assumption: the r rows of W appear as scaled copies in M.

https://cs.stanford.edu/~rishig/courses/ref/l9b.pdf

7. Visualization

Problem: convert a high-dimensional representation into 2D (3D)



t-SNE

PCA is a linear algorithm. It will not be able to interpret complex polynomial relationship between features. On the other hand, t-SNE is based on probability distributions with random walk on neighborhood graphs to find the structure within the data.

t-Distributed Stochastic Neighbouring Entities minimizes the divergence between two distributions: a distribution that measures pairwise similarities of the input objects and a distribution that measures pairwise similarities of the corresponding low-dimensional points in the embedding.

Dependent on the variance hyperparameter.

Recap

counting - matrix factorization - expectation maximization

... but no clustering :(

Vectorize all the things!

Word Embeddings References

- https://lilianweng.github.io/lil-log/2017/10/15/learning-word-embedding.html
- https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/
- https://arxiv.org/pdf/1411.2738v3.pdf
- https://blog.acolyer.org/2016/06/01/distributed-representations-of-sentences-and-documents/
- https://github.com/RaRe-Technologies/gensim/blob/develop/docs/notebooks/doc2vec-IMDB.ipynb
- https://arxiv.org/pdf/1607.05368.pdf
- https://arxiv.org/pdf/1411.4166.pdf
- https://arxiv.org/pdf/1403.6652.pdf

Sense Embeddings References

- Iacobacci et al. (2015), SENSEMBED: Learning Sense Embeddings for Word and Relational Similarity
- Camacho-Collados et al. (2016), Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities
- Faruqui et al. (2015), Retrofitting Word Vectors to Semantic Lexicons
- Jauhar, Dyer et al. (2015), Ontologically Grounded Multi-sense Representation Learning for Semantic Vector Space Models

Topic Modeling References

- http://pages.cs.wisc.edu/~jerryzhu/cs769/latent.pdf
- https://www.youtube.com/watch?v=3mHy4OSyRf0
- https://cs.stanford.edu/~rishig/courses/ref/l9b.pdf
- https://www.quora.com/What-is-an-intuitive-explanation-of-the-Dirichlet-distribution