WSDM 2017 Tutorial

Neural Text Embeddings for IR

Bhaskar Mitra and Nick Craswell

Microsoft

Download slides from: http://bit.ly/NeuIRTutorial-WSDM2017

Check out the full tutorial:

https://arxiv.org/abs/1705.01509

Deep Learning

Amazingly successful on many hard applied problems.

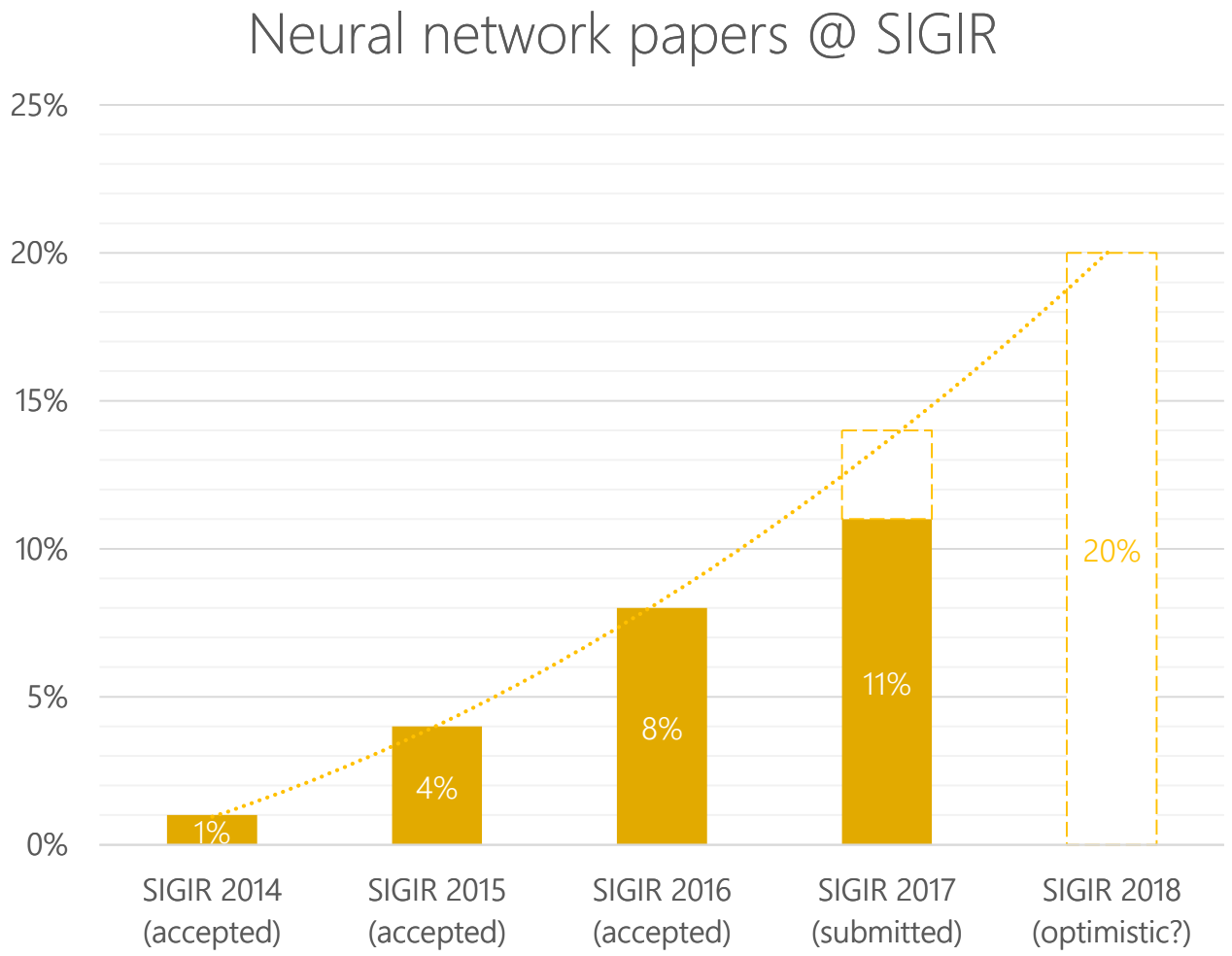
Dominating multiple fields:

2011	2013	2015	2017
speech	vision	NLP	IR

(But also beware the hype.)

Christopher Manning.
[Understanding Human Language: Can NLP and Deep Learning Help?]
Keynote SIGIR 2016 (slides 71,72)

SIGIR papers with title words: Neural, Embedding, Convolution, Recurrent, LSTM



Microsoft

Neural methods for information retrieval

This tutorial mainly focuses on:

- Ranked retrieval of short and long texts, given a text query
- Shallow and deep neural networks
- Representation learning

For broader topics (multimedia, knowledge) see:

- Craswell, Croft, Guo, Mitra, and de Rijke. Neu-IR: Workshop on Neural Information Retrieval. SIGIR 2016 workshop
- Hang Li and Zhengdong Lu. Deep Learning for Information Retrieval. SIGIR 2016 tutorial

Today's agenda

1. IR Fundamentals
2. Word embeddings
3. Word embeddings for IR
4. Deep neural nets
5. Deep neural nets for IR
6. Other applications

bit.ly/NeuIRTutorial-WSDM2017

slides are available here for download

We cover key ideas, rather than exhaustively describing all NN IR ranking papers.

For a more complete overview see:

- Zhang et al. Neural information retrieval: A literature review. 2016
- Onal et al. Getting started with neural models for semantic matching in web search. 2016

Microsoft

bit.ly/
NeuIRTutorial-WSDM2017

send us your questions and feedback during or after the tutorial

Bhaskar Mitra

@UnderdogGeek

bmitra@microsoft.com

Nick Craswell

@nick_craswell

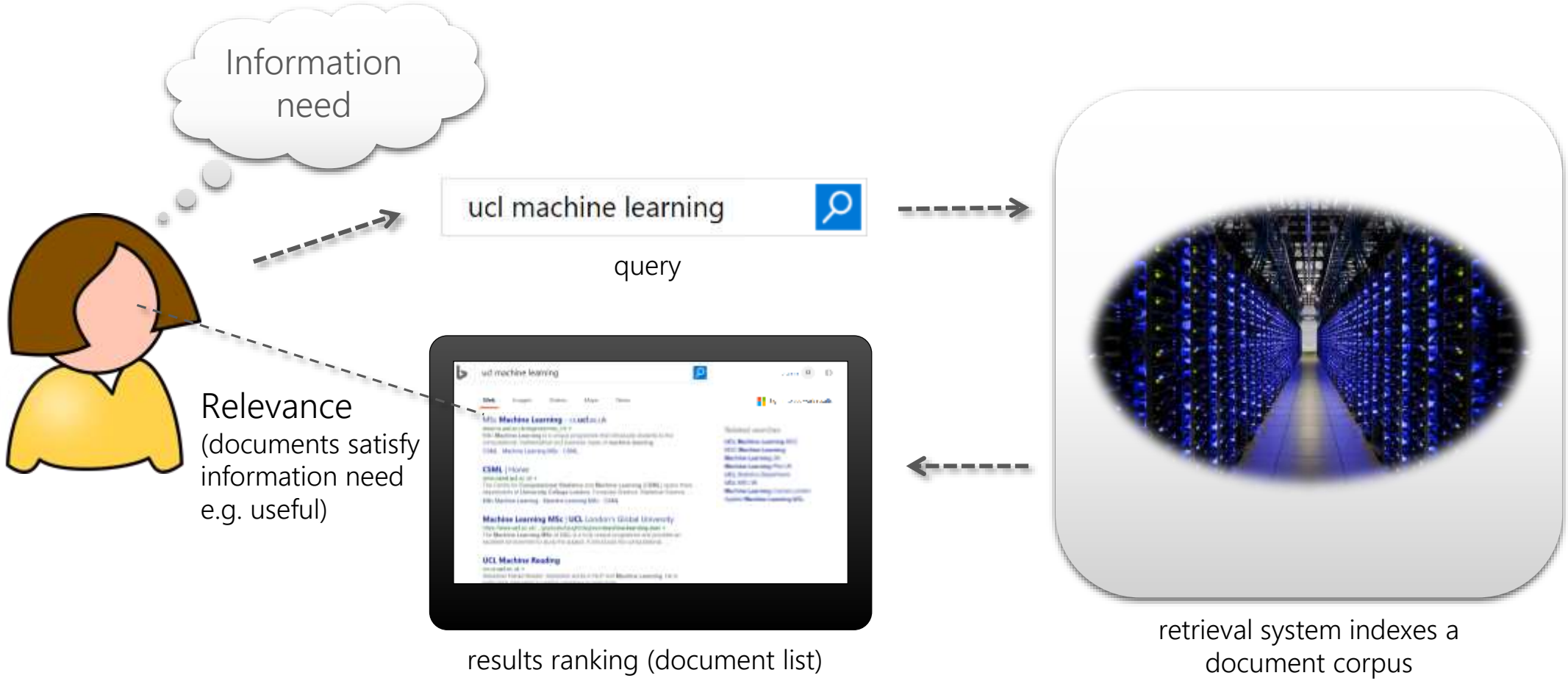
nickcr@microsoft.com

Microsoft

Chapter 1

Fundamentals of Retrieval

Information retrieval (IR) terminology



This tutorial: Using neural networks in the retrieval system to improve relevance. For text queries and documents.

IR applications

	Document ranking	Factoid question answering
Query	Keywords	Natural language question
Document	Web page, news article	A fact and supporting passage
TREC experiments	TREC ad hoc	TREC question answering
Evaluation metric	Average precision, NDCG	Mean reciprocal rank
Research solution	Modern TREC rankers BM25, query expansion, learning to rank, links, clicks+likes (if available)	IBM@TREC-QA Answer type detection, passage retrieval, relation retrieval, answer processing and ranking
In products	• Document rankers at: Google, Bing, Baidu, Yandex, …	• Watson@Jeopardy • Voice search
This NN tutorial	Long text ranking	Short text ranking

Challenges in (neural) IR [slide 1/3]

- Vocabulary mismatch

  Q: How many people live in Sydney?

  ➢ Sydney's population is 4.9 million
  [relevant, but missing 'people' and 'live']

  ➢ Hundreds of people queueing for live music in Sydney
  [irrelevant, and matching 'people' and 'live']

- Need to interpret words based on context (e.g., temporal)

Vocab mismatch:
- Worse for short texts
- Still an issue for long texts

query:
"uk prime minister"

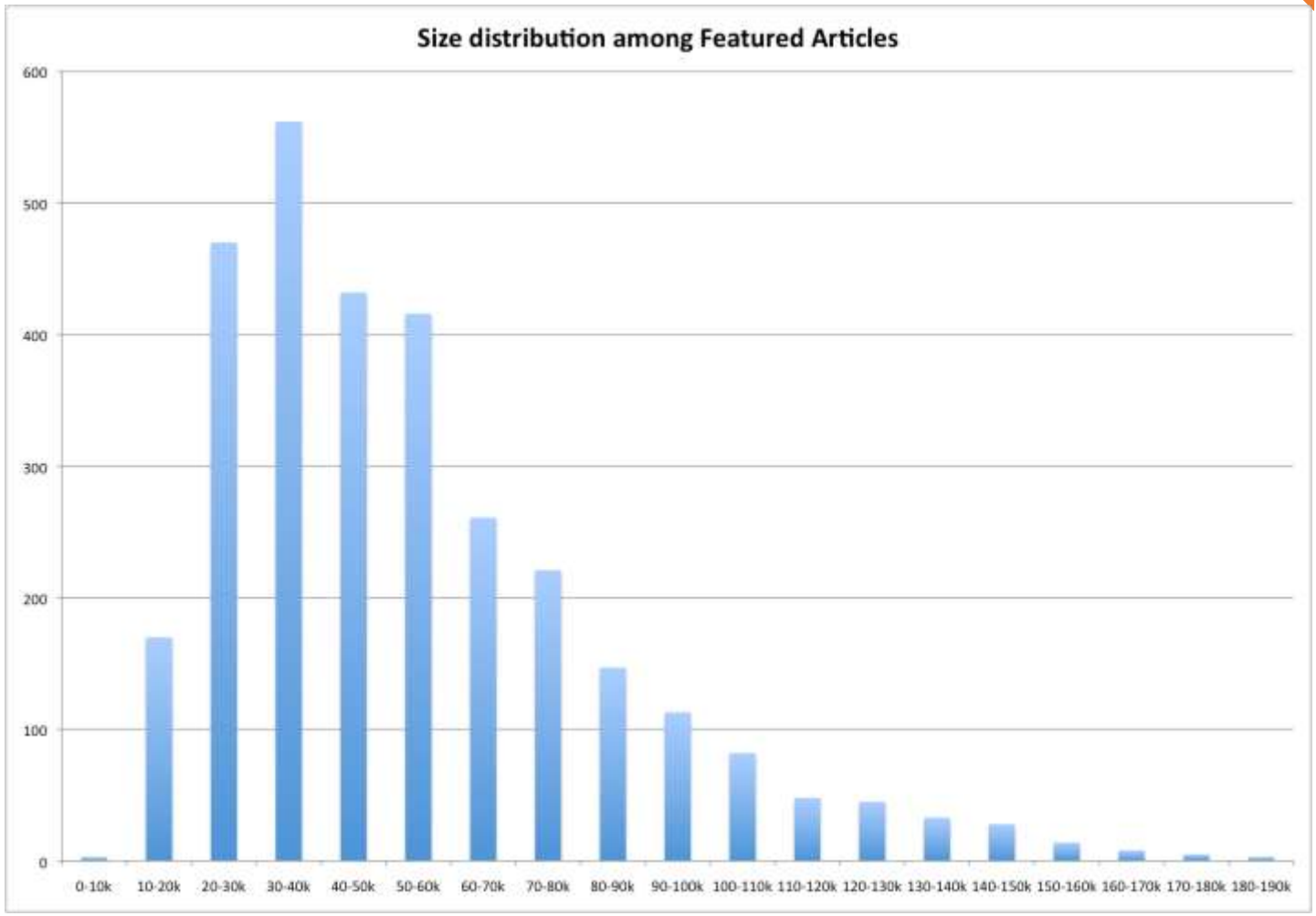

Today


Recent


In older (1990s) TREC data

Challenges in (neural) IR [slide 2/3]

- Q and D vary in length
  - Models must handle variable length input
  - Relevant docs have irrelevant sections



https://en.wikipedia.org/wiki/File:Size_distribution_among_Featured_Articles.png

Challenges in (neural) IR [slide 3/3]

- Need to learn Q-D relationship that generalizes to the tail
  - Unseen Q
  - Unseen D
  - Unseen information needs
  - Unseen vocabulary
- Efficient retrieval over many documents
  - KD-Tree, LSH, inverted files, …

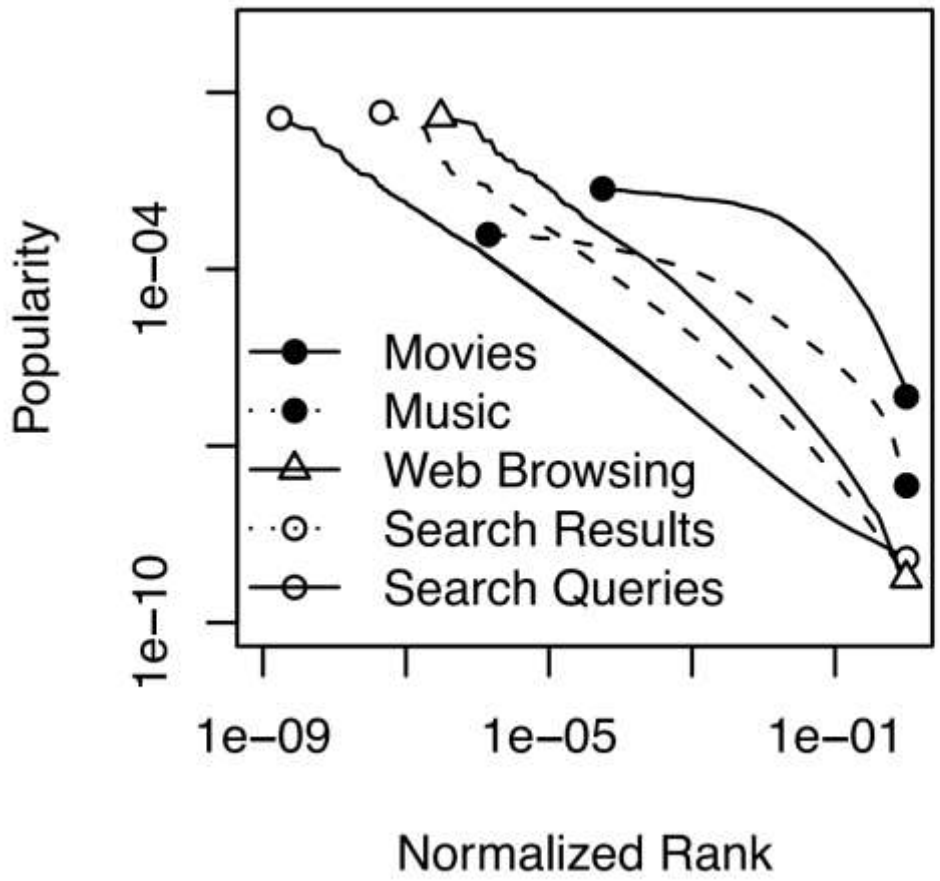


Figure from: Goel, Broder, Gabrilovich, and Pang. Anatomy of the long tail: ordinary people with extraordinary tastes. WWW Conference 2010

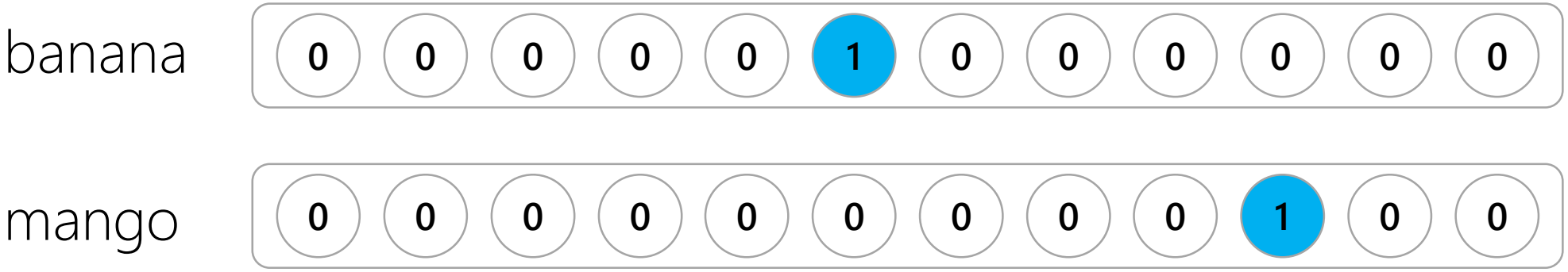
Chapter 2

Representation of Words

One-hot representation (local)

Dim = |V|

sim(banana,mango) = 0



Notes: 1) Popular sim() is cosine, 2) Words/tokens come from some tokenization and transformation

DISTRIBUTED REPRESENTATIONS [1]

Geoffrey E. Hinton
Computer Science Department
Carnegie-Mellon University
Pittsburgh PA 15213

October 1984

Abstract

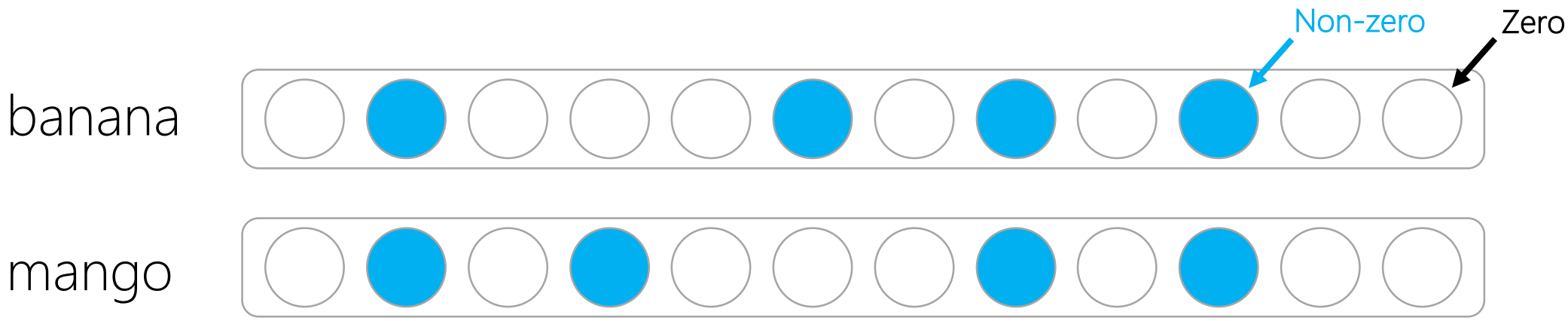
Given a network of simple computing elements and some entities to be represented, the most straightforward scheme is to use one computing element for each entity. This is called a *local* representation. It is easy to understand and easy to implement because the structure of the physical network mirrors the structure of the knowledge it contains. This report describes a different type of representation that is less familiar and harder to think about than local representations. Each entity is represented by a pattern of activity distributed over many computing elements, and each computing element is involved in representing many different entities. The strength of this more complicated kind of representation does not lie in its notational convenience or its ease of implementation in a conventional computer, but rather in the efficiency with which it makes use of the processing abilities of networks of simple, neuron-like computing elements.

One way of thinking about distributed memories is in terms of a very large set of plausible inference rules. Each active unit represents a "micro-feature" of an item, and the connection strengths stand for plausible inferences between micro-features. Any particular pattern of activity of the units will satisfy some of the "micro-inferences" and violate others. A stable pattern of activity is one that violates the plausible micro-inferences less than any of the neighboring patterns. A new stable pattern can be created by changing the

Microsoft

Hinton, Geoffrey E. Distributed representations. Technical Report CMU-CS-84-157, 1984

Context-based distributed representation



sim(banana,mango) > 0

- Appear in same documents
- Appear near same words

"You shall know a word by the company it keeps"

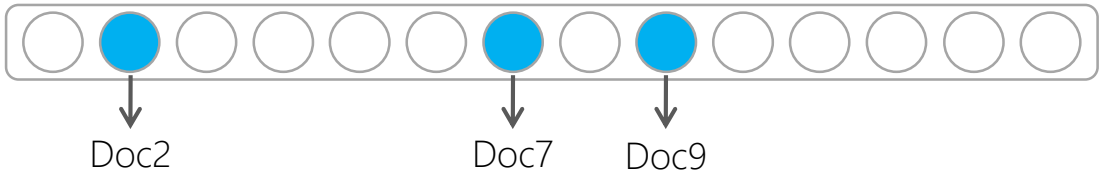
Firth, J. R. (1957). A synopsis of linguistic theory 1930–1955. In Studies in Linguistic Analysis, p. 11. Blackwell, Oxford.

Turney and Pantel. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research* 2010

Distributional Semantics

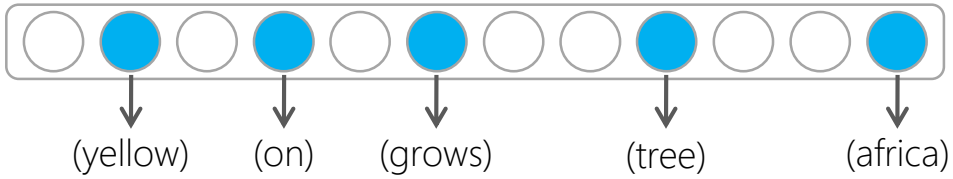
Word-Document

banana



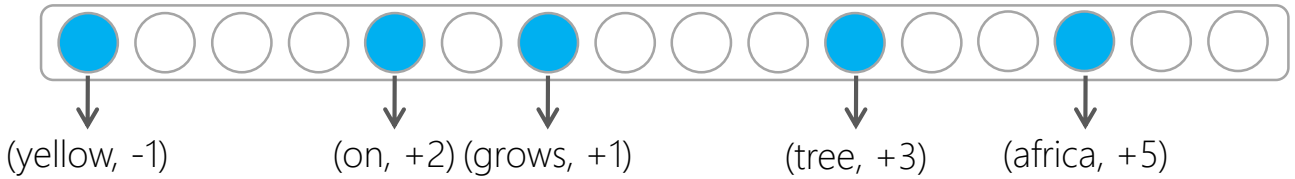
Word-Word

banana



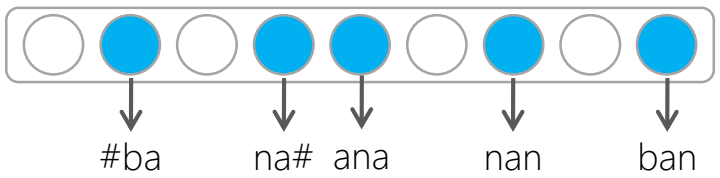
Word-WordDist

banana



Word hash
(not context-based)

banana



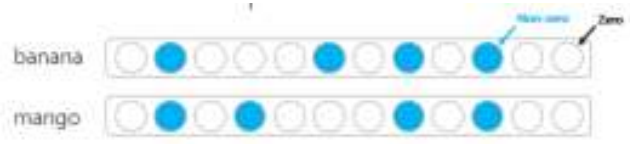
"You shall know a word
by the company it keeps"

Distributed and distributional

- **Distributed representation**: Vector represents a concept as a pattern, rather than 1-hot

- **Distributional semantics**: Linguistic items with similar distributions (e.g. context words) have similar meanings

Distributional methods use distributed representations



"You shall know a word by the company it keeps"

http://www.marekrei.com/blog/26-things-i-learned-in-the-deep-learning-summer-school/

Vector Space Models

PPMI weighting for Word-Word matrix
TF-IDF weighting for Word-Document matrix

For a given task: Choose matrix, choose s_{ij} weighting.

Could be binary, could be raw counts.

Example weighting:

Positive Pointwise Mutual Information (Word-Word matrix)

V: vocabulary, C: set of contexts, S: sparse matrix $|V|$ x $|C|$

$$S_{ij} = PPMI(w_i, c_j)$$

$$PPMI(w,c) = \begin{cases} 0 & PMI(w,c) < 0 \\ PMI(w,c) & otherwise \end{cases}$$

$$PMI(w,c) = \log \frac{P(w,c)}{P(w)P(c)} = \log \frac{freq(w,c)|corpus|}{freq(w)freq(c)}$$

	c_0	c_1	c_2	...	c_j	...	$c_{|C|}$
w_0							
w_1							
w_2							
...							
w_i					s_{ij}		
...							
$w_{|V|}$							

Microsoft

Turney and Pantel. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research* 2010

Distributed word representation overview

- Next we cover lower-dimensional dense representations
  - Including word2vec. Questions on advantage*
- But first we consider the effect of context choice, in the data itself

context ↕

Data	Counts matrix (sparse)	Learning from counts matrix	Learning from individual instances
Word-Document	Vector space models	LSA	Paragraph2Vec PV-DBOW
Word-Word	Vector space models	GloVe	Word2vec
Word-WordDist	Vector space models		

* Baroni, Dinu and Kruszewski. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. ACL 2014
Levy, Goldberg and Dagan. Improving distributional similarity with lessons learned from word embeddings. Transactions of the ACL 3:211-225

Let's consider the following example…

We have four (tiny) documents,

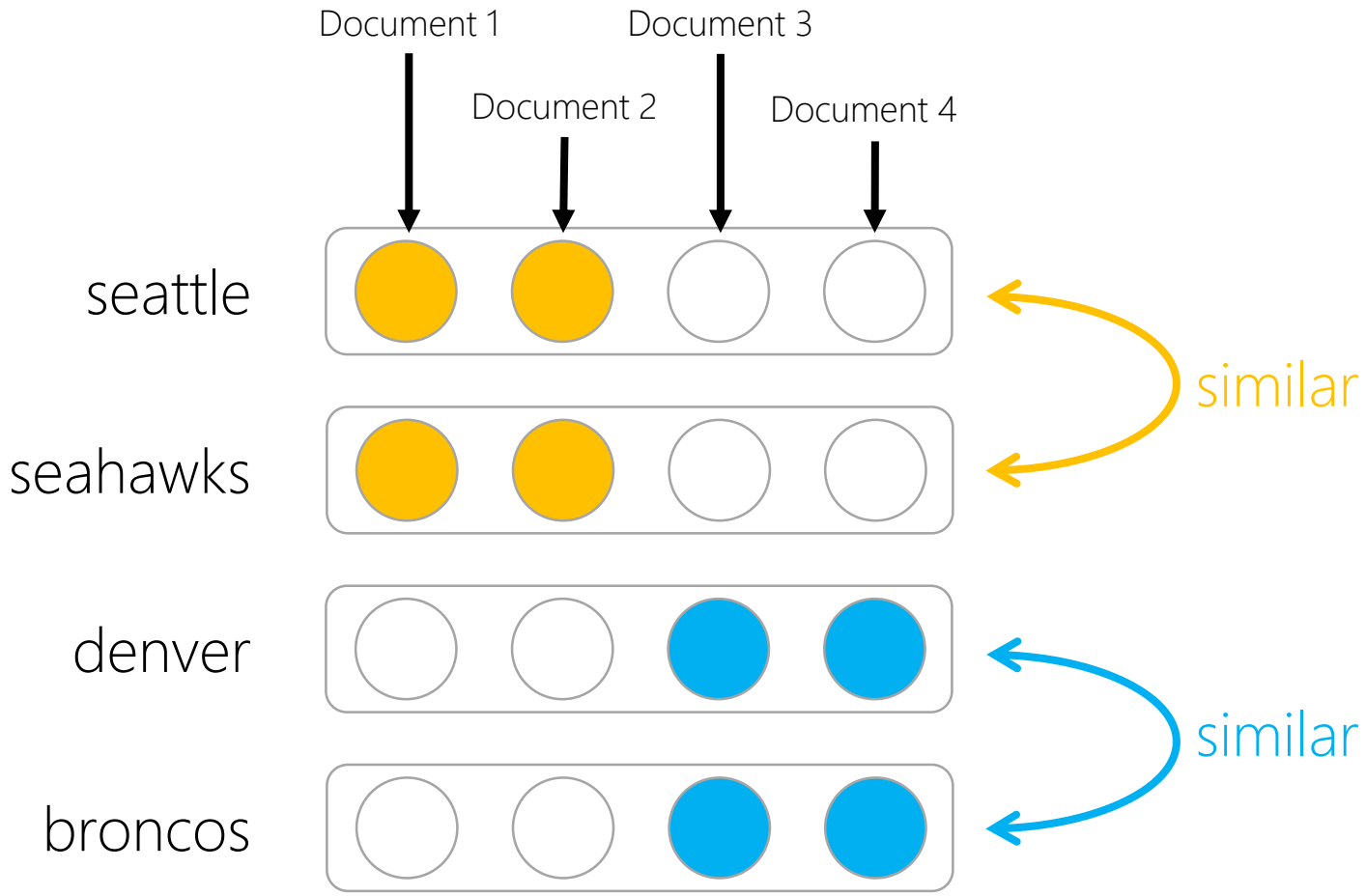
Document 1 : "seattle seahawks jerseys"

Document 2 : "seattle seahawks highlights"

Document 3 : "denver broncos jerseys"

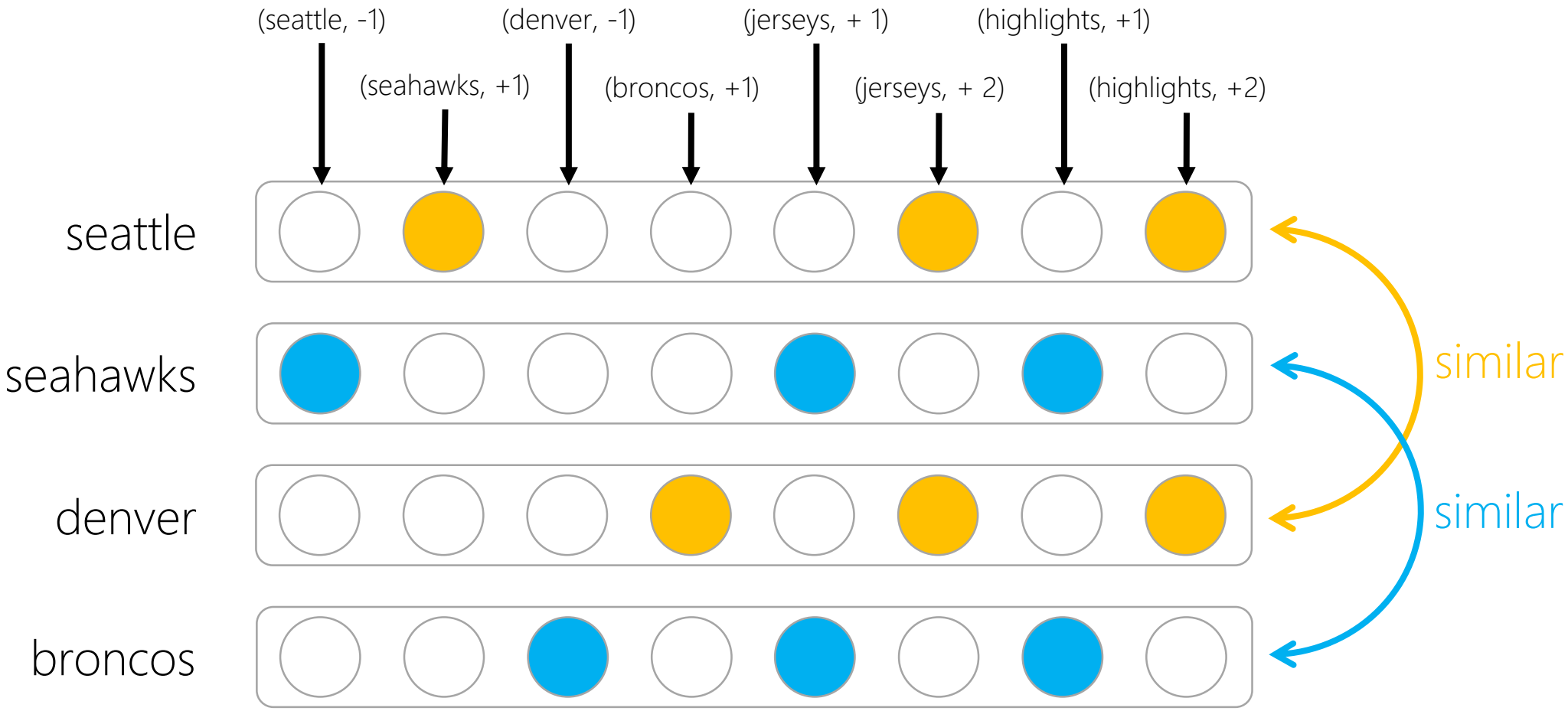
Document 4 : "denver broncos highlights"

Using Word-Document context



This is *Topical* or *Syntagmatic* similarity.

Using Word-WordDist context



This is *Typical* or *Paradigmatic* Similarity.

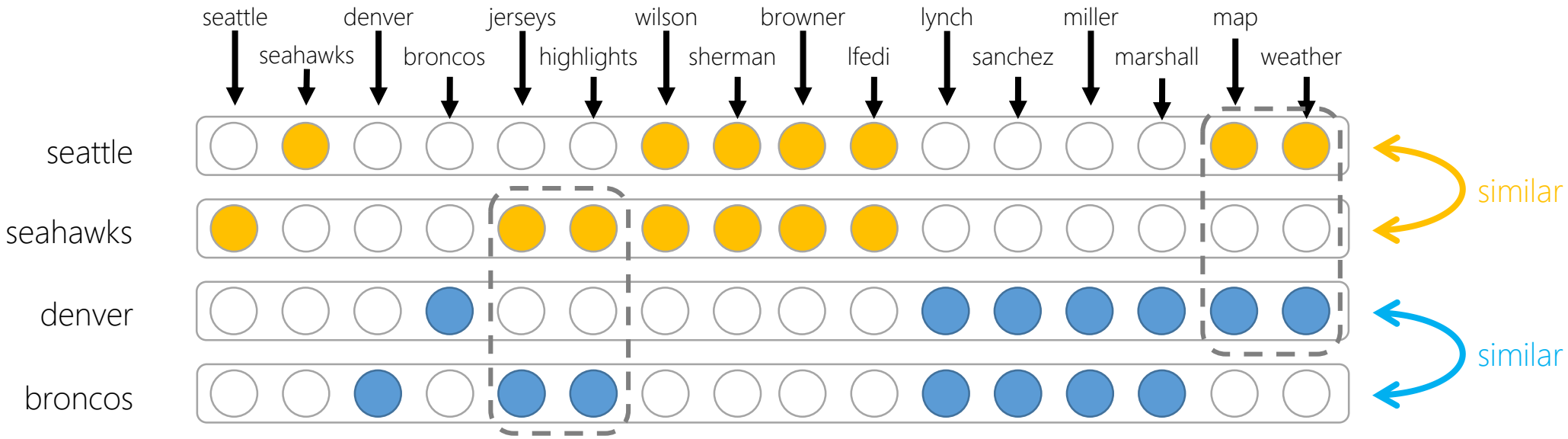
Let's consider another example…

With more (tiny) documents…

"seattle map"
"seattle weather"
"seahawks jerseys"
"seahawks highlights"
"seattle seahawks wilson"
"seattle seahawks sherman"
"seattle seahawks browner"
"seattle seahawks lfedi"

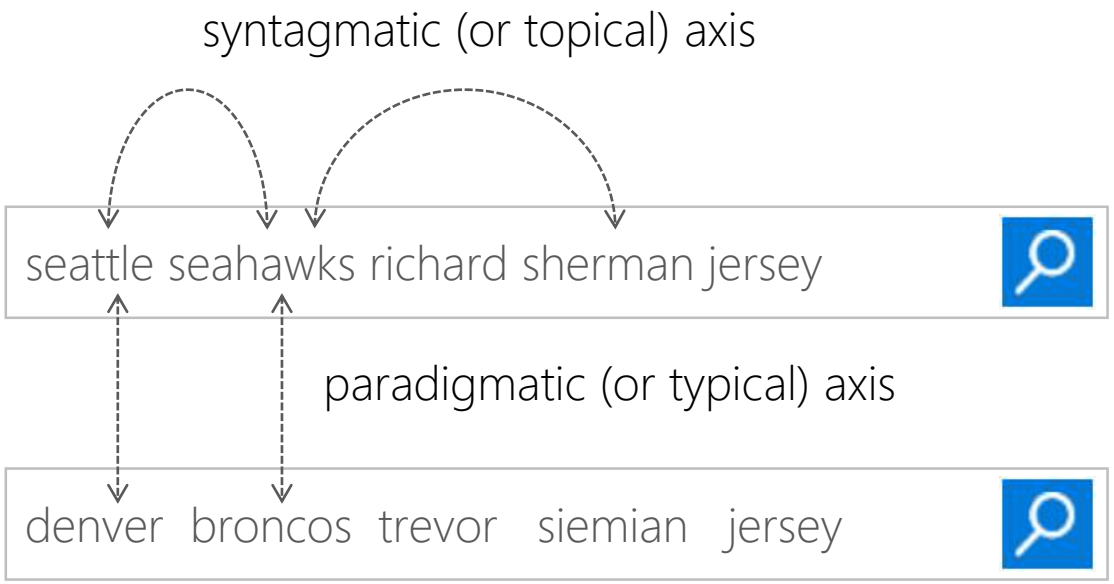
"denver map"
"denver weather"
"broncos jerseys"
"broncos highlights"
"denver broncos lynch"
"denver broncos sanchez"
"denver broncos miller"
"denver broncos marshall"

Using Word-Word context



1. Word-Word is less sparse than Word-Document (Yan et al., 2013)
2. A mix of topical and typical similarity (function of window size)

Paradigmatic vs Syntagmatic



Do we choose one axis or the other, or both, or something else? Can we learn a general-purpose word representation that solves all our IR problems? Or do we need several?

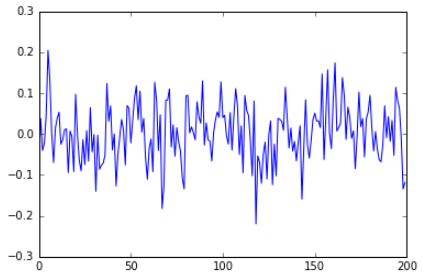
Embeddings (distributed)

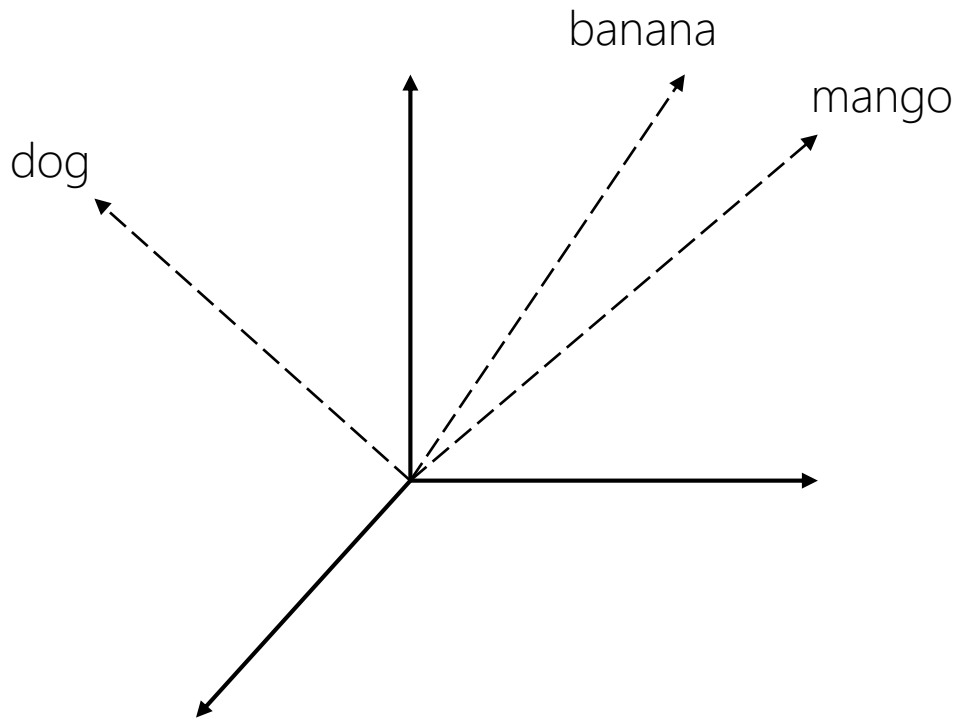
Dense. Dim = 200 (for example)

```
In [67]: print(vec['banana'])
         plt.plot(vec['banana'])
```

```
[-0.065091, 0.037847, -0.040299, -0.022862, 0.046481, 0.204306, 0.132157, 0.000275, -0.069716, 0.014626, 0.038425, 0.053029, -
0.024947, -0.013991, 0.010317, 0.012735, -0.094237, 0.007101, -0.007268, -0.091869, 0.097138, -0.002357, -0.065102, -0.089856,
 -0.013727, -0.074923, 0.007938, -0.066188, 0.064525, -0.0436, -0.001177, -0.140017, -0.003096, -0.086315, -0.0763, -0.071214,
 -0.051458, 0.123467, 0.031151, 0.068839, -0.039029, 4e-06, -0.127185, -0.049415, -0.007708, 0.035502, 0.009538, -0.075545, 0.0
69583, 0.062794, -0.021556, 0.031155, 0.087352, 0.117663, 0.034883, 0.104613, 0.004534, 0.037999, -0.058016, -0.110679, -0.0353
5, -0.012488, -0.0924, 0.126315, 0.080949, -0.040334, 0.047046, -0.182169, -0.1268, 0.082376, 0.082963, 0.110073, -0.031732, 0.
022219, -0.054332, 0.015394, -0.019853, -0.04169, -0.106969, -0.134253, 0.093094, 0.094716, 0.002643, 0.017417, 0.00309, -0.014
145, 0.078464, 0.041464, 0.026328, 0.12988, -0.02715, 0.027002, -0.014312, -0.017305, -0.066002, 0.002747, 0.033995, 0.053829,
 0.040628, 0.127369, 0.040216, 0.045803, -0.003395, -0.024843, 0.052411, -0.039267, 0.043378, 0.110868, 0.067947, -0.050505, 0.
019753, -0.094825, 0.094058, 0.057547, 0.045447, -0.016258, -0.102323, 0.080506, -0.219969, -0.053595, -0.069609, -0.120579, -
0.048799, -0.019837, -0.109987, -0.002571, 0.031825, -0.124037, -0.024646, -0.102276, 0.038512, 0.035166, 0.031713, 0.008979,
 0.114415, 0.0421, -0.034152, 0.014497, -0.04199, -0.018534, -0.065822, -0.020059, 0.019861, -0.159393, -0.03374, 0.083666, -0.
025234, -0.058921, -0.014924, 0.035292, 0.050979, 0.031609, 0.0322, 0.015638, 0.146793, -0.062475, 0.042192, 0.157084, 0.00237
1, -0.035507, 0.08275, 0.173776, 0.007175, 0.016044, 0.025942, 0.137863, 0.094541, -0.013125, 0.065621, 0.040823, -0.010574, 0.
007796, -0.085031, -0.003617, 0.102267, 0.018047, 0.037613, -0.056187, 0.036693, 0.053867, 0.094616, 0.015941, -0.041536, 0.005
796, -0.03694, -0.063241, -0.067796, -0.026023, 0.069142, -0.008786, 0.042428, -0.017718, 0.03318, -0.052277, 0.114012, 0.08154
2, 0.063282, -0.012149, -0.134274, -0.118431]
```

```
Out[67]: [<matplotlib.lines.Line2D at 0x12a60774e48>]
```





Microsoft

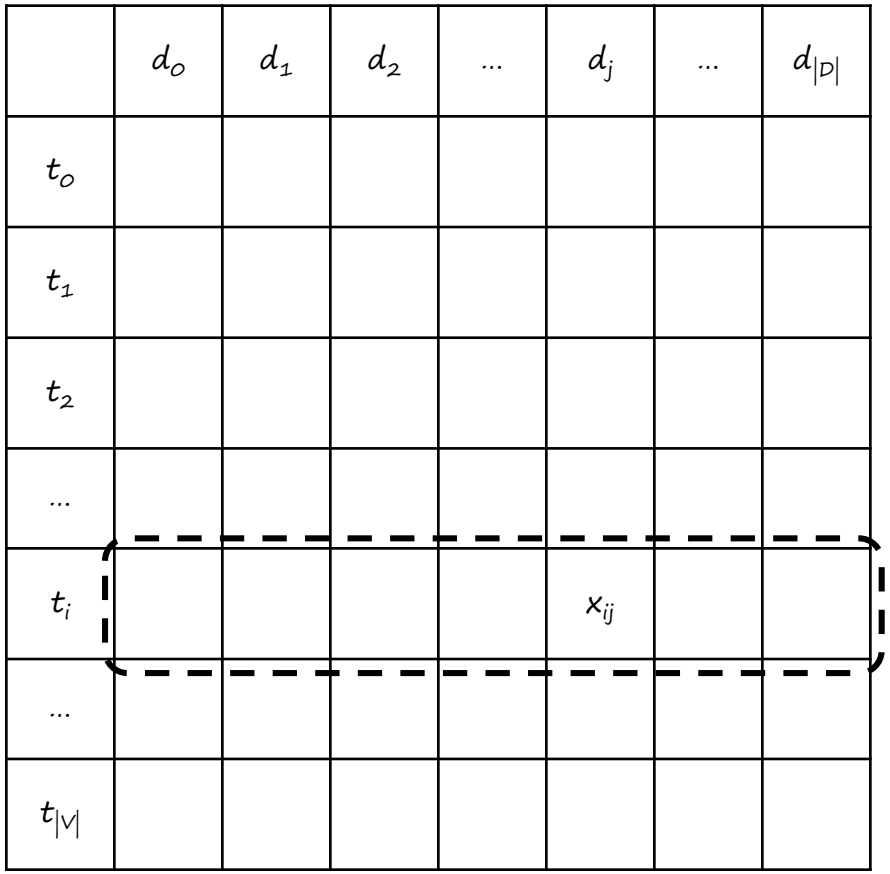
Latent Semantic Analysis

V: vocabulary, D: set of documents, X: sparse matrix $|V| \times |D|$

$$x_{ij} = \text{TF-IDF}$$

Singular value decomposition of X:

$$X = U\Sigma V^T$$



Microsoft

Deerwester, Dumais, Furnas, Landauer and Harshman. Indexing by latent semantic analysis. JASIS 41, no. 6, 1990

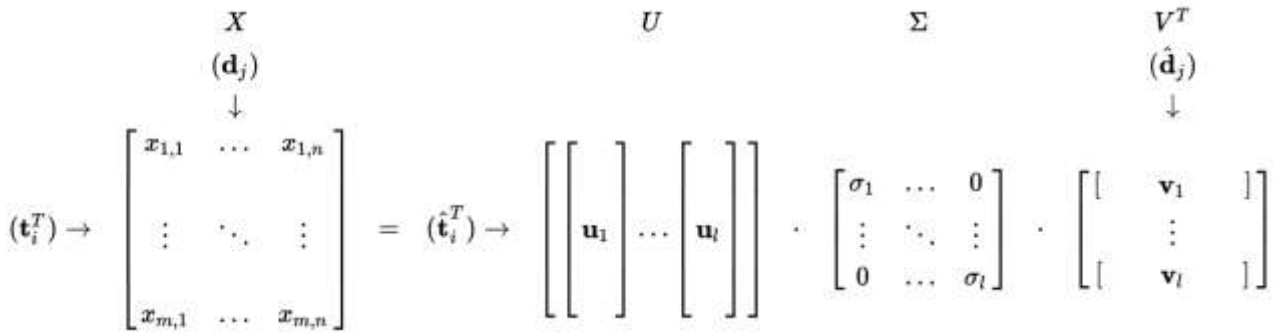
Latent Semantic Analysis

$\sigma_1, \ldots, \sigma_l$: singular values
$u_1, \ldots, u_l$: left singular vectors
$v_1, \ldots, v_l$: right singular vectors

The k largest singular values, and corresponding singular vectors from U and V, is the rank k approximation of X

$$X_k = U_k \Sigma_k V_k^T$$

Embedding of i^{th} term = $\Sigma_k \hat{t_i}$



Source: en.wikipedia.org/wiki/Latent_semantic_analysis

Dumais. Latent semantic indexing (LSI): TREC-3 report. NIST Special Publication SP (1995): 219-219.

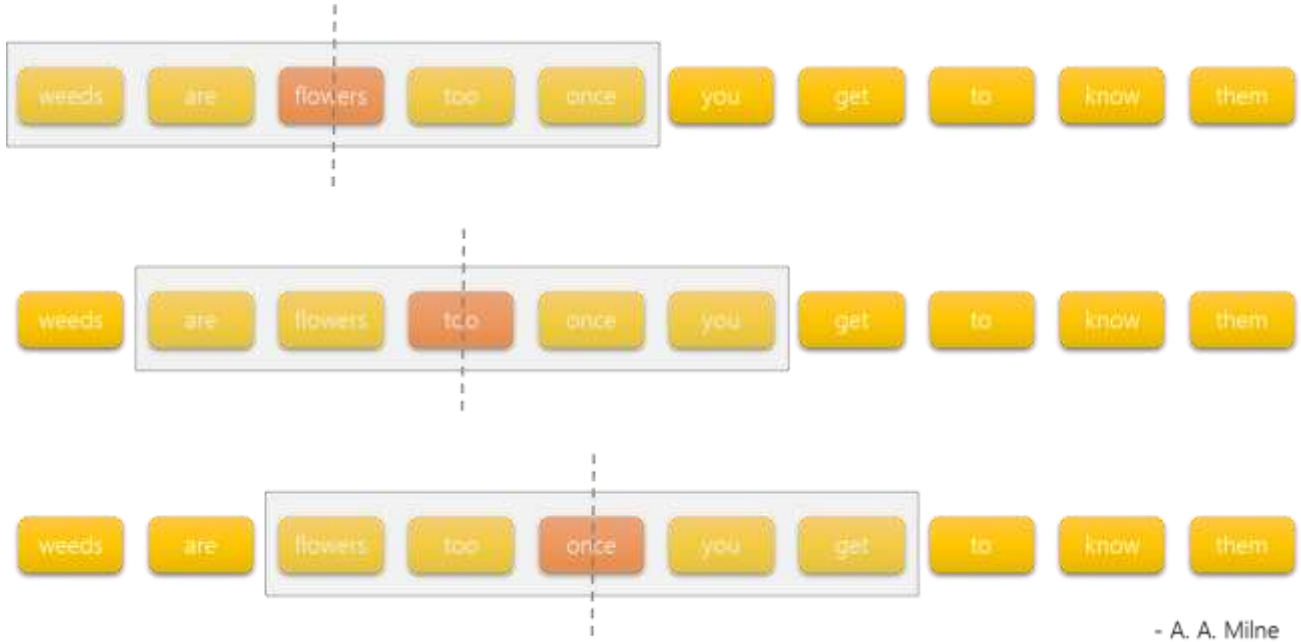
Word2vec

Goal: simple (shallow) neural model learning from billion words scale corpus

Predict middle word from neighbors within a fixed size context window

Two different architectures:

1. Skip-gram
2. CBOW



(Mikolov et al., 2013)

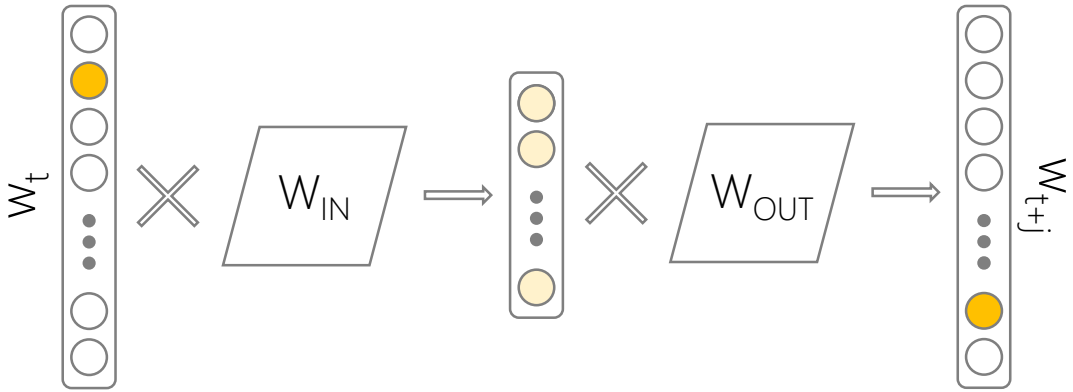
Skip-gram

Predict neighbor w_{t+j} given word w_t

Maximizes following average log prob.

$$\mathcal{L}_{skip-gram} = \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} \log p(w_{t+j}|w_t)$$

$$p(w_{t+j}|w_t) = \frac{exp\left(\left(W_{OUT}w_{t+j}\right)^{\intercal}(W_{IN}w_t)\right)}{\sum_{v=1}^{V} exp\left((W_{OUT}w_v)^{\intercal}(W_{IN}w_t)\right)}$$



Full softmax is computationally impractical. Word2vec uses hierarchical softmax or negative sampling instead.

(Mikolov et al., 2013)

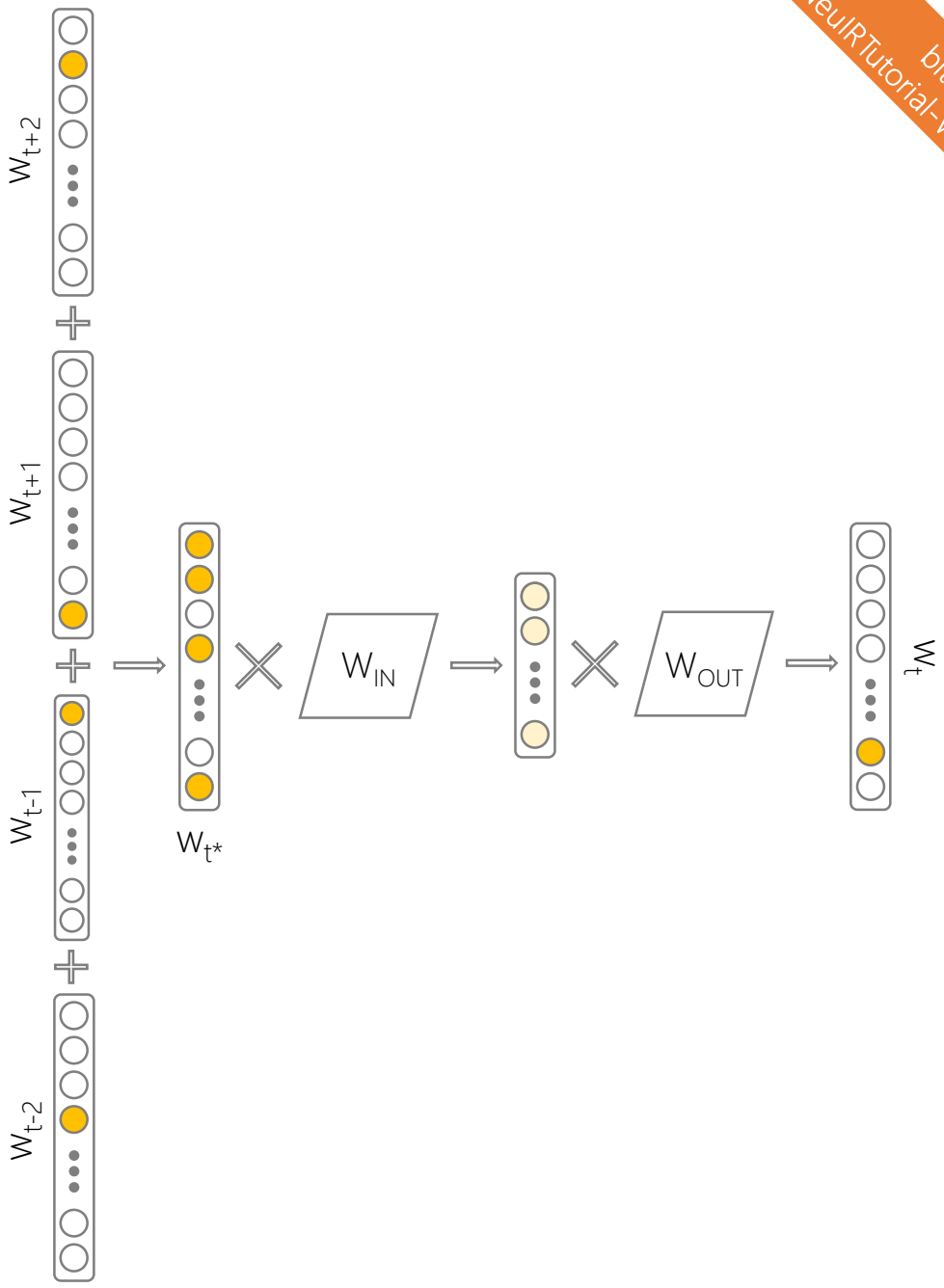
Continuous Bag-of-Words

Predict word given bag-of-neighbors

Modify the skip-gram loss function.

$$\mathcal{L}_{skip-gram} = \frac{1}{T}\sum_{t=1}^{T} \log p(w_t|w_{t*})$$

$$w_{t*} = \sum_{-c \le j \le c, j \ne 0} w_{t+j}$$

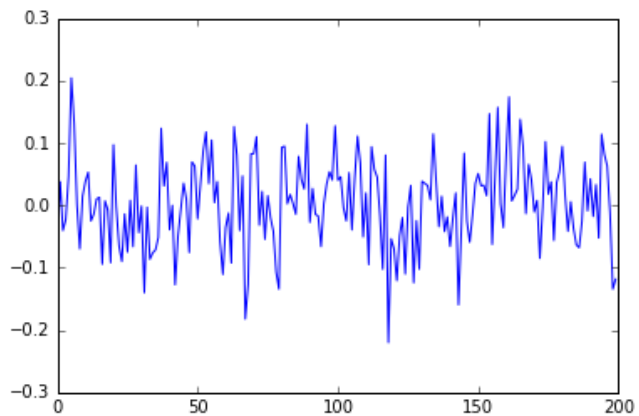


(Mikolov et al., 2013)

In [67]:
```
print(vec['banana'])
plt.plot(vec['banana'])
```

```
[-0.065091, 0.037847, -0.040299, -0.022862, 0.046481, 0.204306, 0.132157, 0.000275, -0.069716, 0.014626, 0.038425, 0.053029, -
0.024947, -0.013991, 0.010317, 0.012735, -0.094237, 0.007101, -0.007268, -0.091869, 0.097138, -0.002357, -0.065102, -0.089856,
 -0.013727, -0.074923, 0.007938, -0.066188, 0.064525, -0.0436, -0.001177, -0.140017, -0.003096, -0.086315, -0.0763, -0.071214,
 -0.051458, 0.123467, 0.031151, 0.068839, -0.039029, 4e-06, -0.127185, -0.049415, -0.007708, 0.035502, 0.009538, -0.075545, 0.0
69583, 0.062794, -0.021556, 0.031155, 0.087352, 0.117663, 0.034883, 0.104613, 0.004534, 0.037999, -0.058016, -0.110679, -0.0353
5, -0.012488, -0.0924, 0.126315, 0.080949, -0.040334, 0.047046, -0.182169, -0.1268, 0.082376, 0.082963, 0.110073, -0.031732, 0.
022219, -0.054332, 0.015394, -0.019853, -0.04169, -0.106969, -0.134253, 0.093094, 0.094716, 0.002643, 0.017417, 0.00309, -0.014
145, 0.078464, 0.041464, 0.026328, 0.12988, -0.02715, 0.027002, -0.014312, -0.017305, -0.066002, 0.002747, 0.033995, 0.053829,
 0.040628, 0.127369, 0.040216, 0.045803, -0.003395, -0.024843, 0.052411, -0.039267, 0.043378, 0.110868, 0.067947, -0.050505, 0.
019753, -0.094825, 0.094058, 0.057547, 0.045447, -0.016258, -0.102323, 0.080506, -0.219969, -0.053595, -0.069609, -0.120579, -
0.048799, -0.019837, -0.109987, -0.002571, 0.031825, -0.124037, -0.024646, -0.102276, 0.038512, 0.035166, 0.031713, 0.008979,
 0.114415, 0.0421, -0.034152, 0.014497, -0.04199, -0.018534, -0.065822, -0.020059, 0.019861, -0.159393, -0.03374, 0.083666, -0.
025234, -0.058921, -0.014924, 0.035292, 0.050979, 0.031609, 0.0322, 0.015638, 0.146793, -0.062475, 0.042192, 0.157084, 0.00237
1, -0.035507, 0.08275, 0.173776, 0.007175, 0.016044, 0.025942, 0.137863, 0.094541, -0.013125, 0.065621, 0.040823, -0.010574, 0.
007796, -0.085031, -0.003617, 0.102267, 0.018047, 0.037613, -0.056187, 0.036693, 0.053867, 0.094616, 0.015941, -0.041536, 0.005
796, -0.03694, -0.063241, -0.067796, -0.026023, 0.069142, -0.008786, 0.042428, -0.017718, 0.03318, -0.052277, 0.114012, 0.08154
2, 0.063282, -0.012149, -0.134274, -0.118431]
```

Out[67]:
```
[<matplotlib.lines.Line2D at 0x12a60774e48>]
```

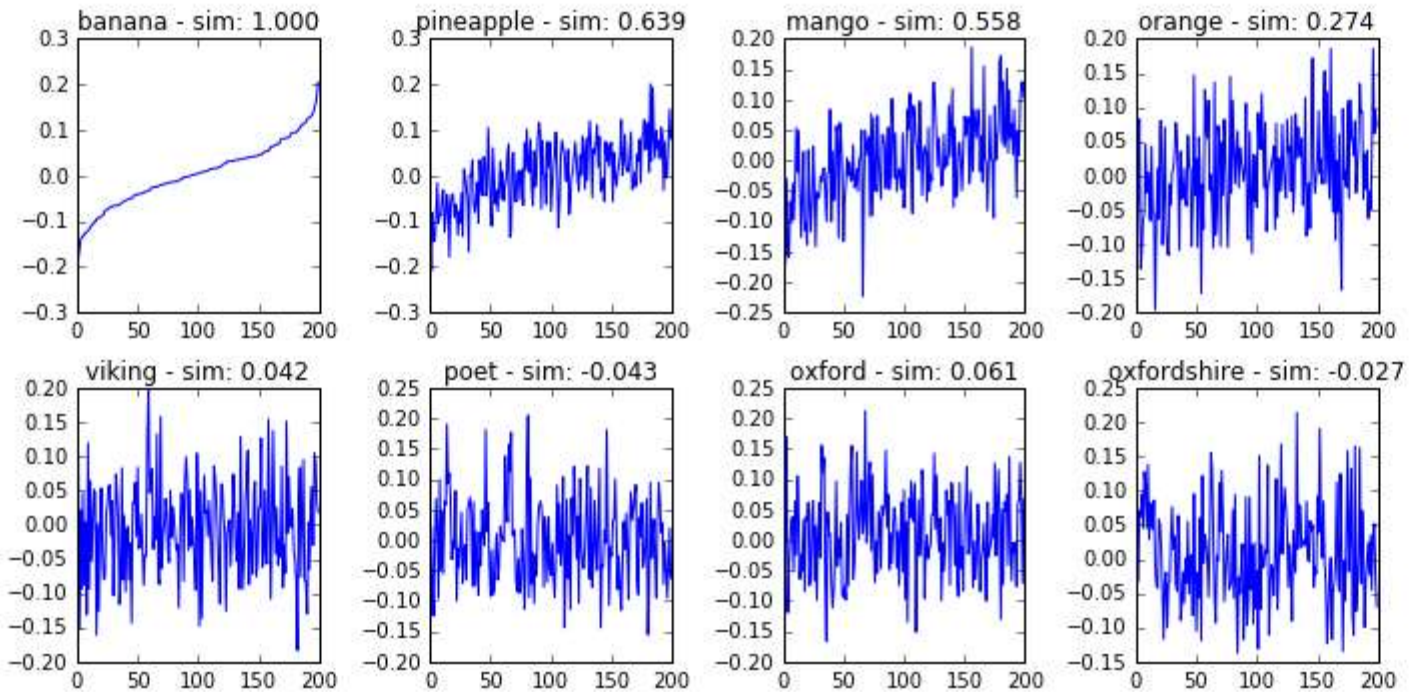


Microsoft

In [84]:
```
plt.figure(figsize=(10,5))

i=1
for k in ['banana','pineapple','mango','orange','viking','poet','oxford','oxfordshire']:
    plt.subplot(240+i)
    i += 1
    X = vec[k]
    Y = vec['banana']
    plt.plot([x for (y,x) in sorted(zip(Y,X))])
    plt.title((k + " - sim: %.3f" ) % (1-spatial.distance.cosine(X,Y)))

plt.tight_layout()
```

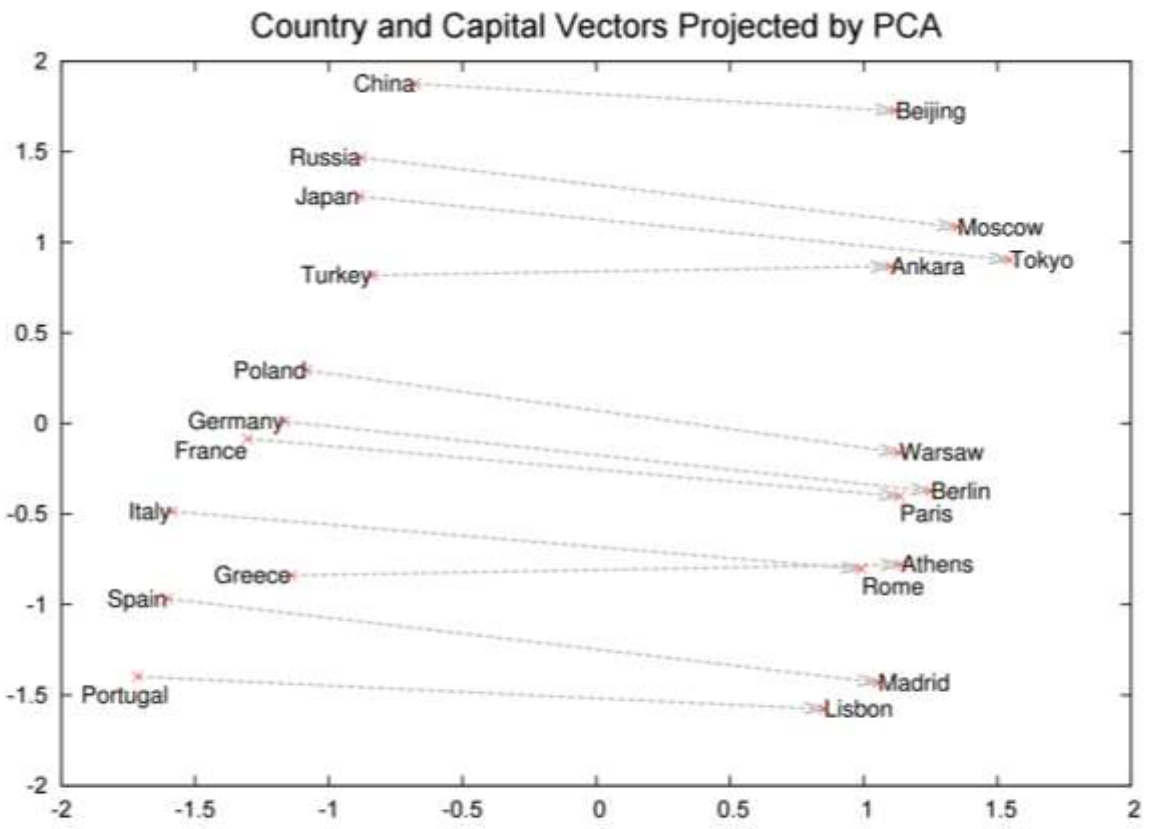


Word analogies with word2vec

[king] – [man] + [woman] ≈ [queen]

Table 1: *Examples of five types of semantic and nine types of syntactic questions in the Semantic-Syntactic Word Relationship test set.*

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

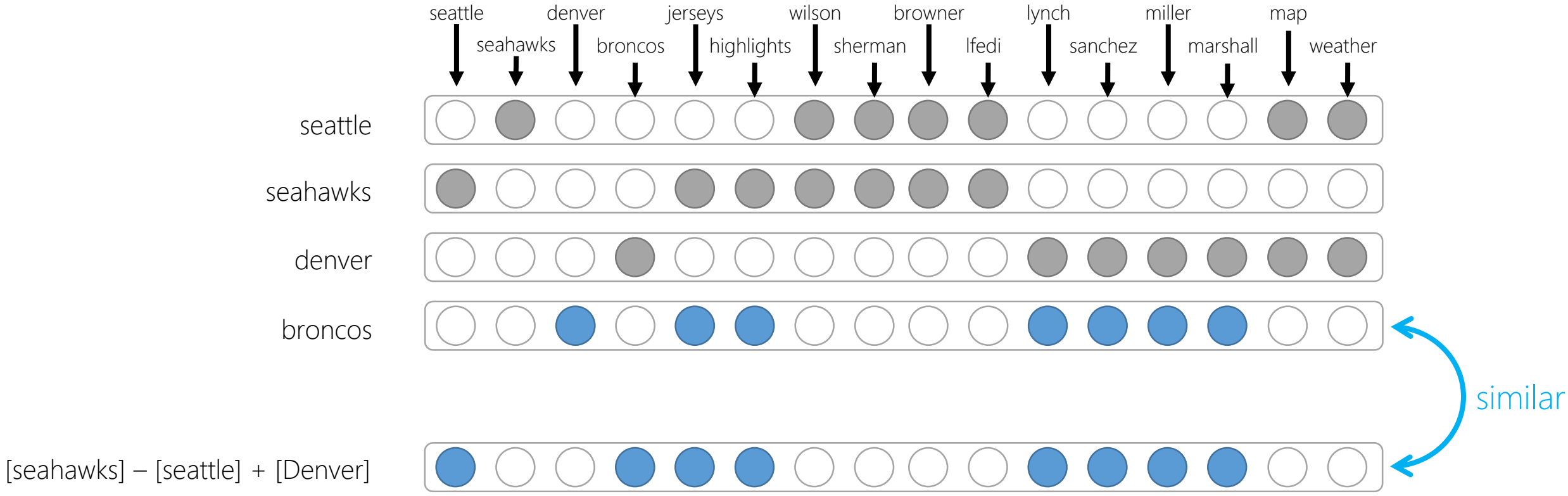


igure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and the
pital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitl
e relationships between them, as during the training we did not provide any supervised information abo
hat a capital city means.

(Mikolov et al., 2013)

Microsoft

Word analogies can work in underlying data too



Sparse vectors can work well for an analogy task:
Levy, Goldberg and Ramat-Gan. Linguistic Regularities in Sparse and Explicit Word Representations. *CoNLL* 2014

A Matrix Interpretation of word2vec

Skip-gram looks like this:

$$\mathcal{L}_{skip-gram} = -\sum_{A}\sum_{B} \log p(B|A)$$

If we aggregate over all training samples…

$$\mathcal{L}_{skip-gram} = -\sum_{i=1}^{V}\sum_{j=1}^{V} x_{i,j} \log p(w_i|w_j)$$

$$= -\sum_{i=1}^{V} x_i \sum_{j=1}^{V} \frac{X_{i,j}}{X_i} \log p(w_i|w_j)$$

$$= -\sum_{i=1}^{V} x_i \sum_{j=1}^{V} P(w_i|w_j) \log p(w_i|w_j)$$

$$= \sum_{i=1}^{V} x_i H\left(P(w_i|w_j), p(w_i|w_j)\right)$$

actual co-occurence probability

cross-entropy

co-occurence probability predicted by the model

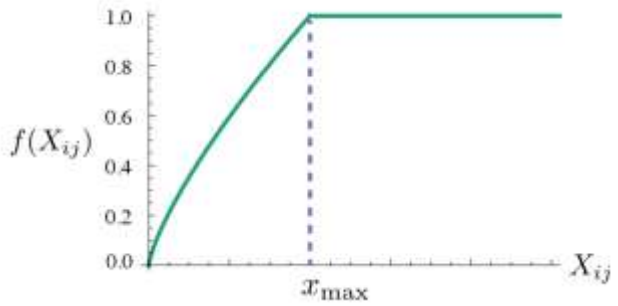
	w_0	w_1	w_2	…	w_j	…	$w_{\|V\|}$
w_0							
w_1							
w_2							
…							
w_i					x_{ij}		
…							
$w_{\|V\|}$							

(Pennington et al., 2014)

GloVe

Variety of windows sizes and weighting

AdaGrad



$$\mathcal{L}_{GloVe} = -\sum_{i=1}^{V}\sum_{j=1}^{V} f(X_{i,j})\left(logX_{i,j} - w_i^{\top} w_j\right)^2$$

weighting function

actual co-occurence probability`

squared error

co-occurence probability predicted by the model

	w_0	w_1	w_2	...	w_j	...	$w_{\|V\|}$
w_0							
w_1							
w_2							
...							
w_i					X_{ij}		
...							
$w_{\|V\|}$							

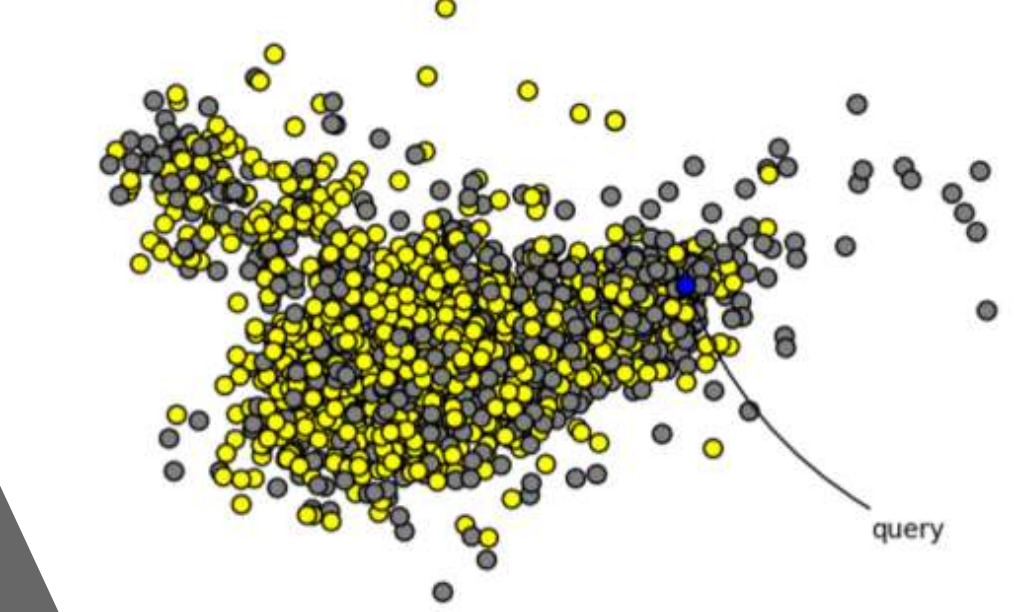
(Pennington et al., 2014)

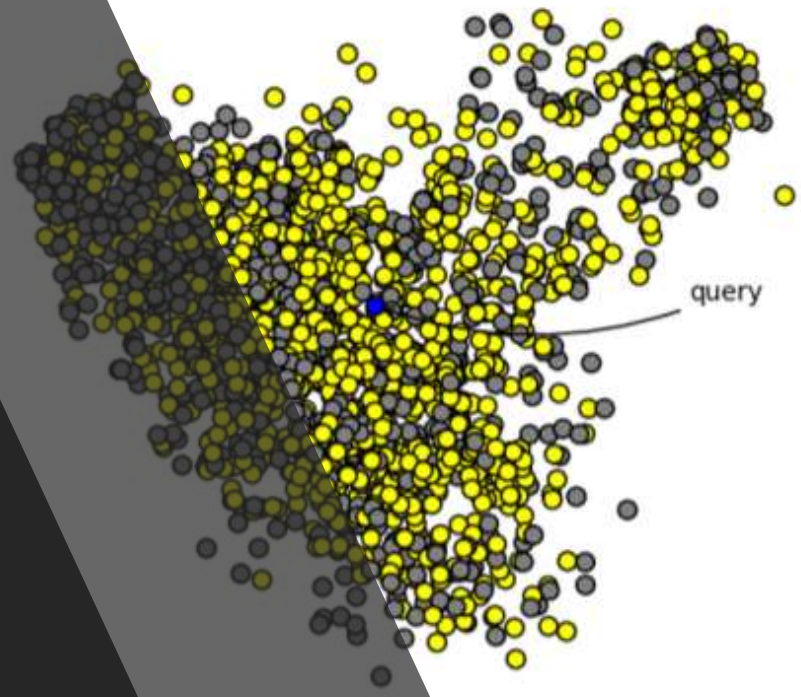
Discussion of word representations

- Representations: Weighted counts, SVD, neural embedding
  - Use similar data (W-D, W-W), capture similar semantics
  - For example, analogies using counts
  - Think sparse, act dense
- Stochastic gradient descent allows us to scale to larger data
  - On individual instances (w2v) or count matrix data (GloVe)
  - Scalability could be the key advantage for neural word embeddings
- Choice of data affects semantics
  - Paradigmatic vs Syntagmatic
  - Which is best for your application?

Chapter 3

Word Embeddings for IR



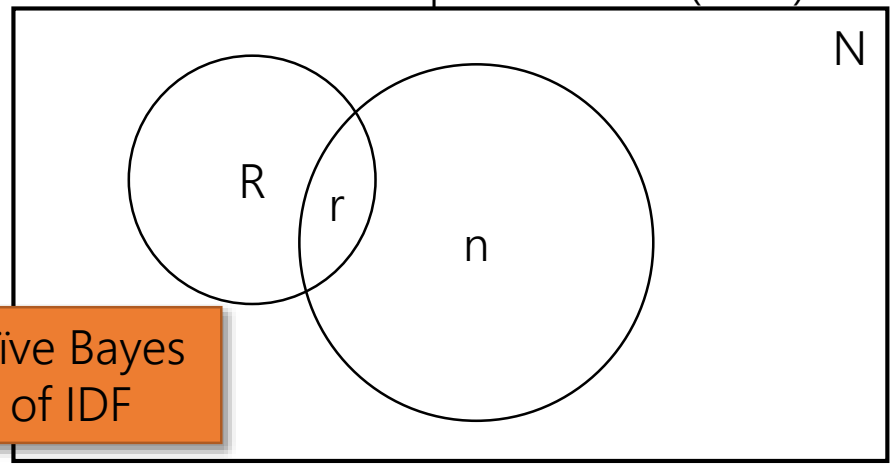


Robertson and Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, no. 4 (2009)

Traditional IR feature design

- **Inverse document frequency**

Robertson and Sparck-Jones (1977)



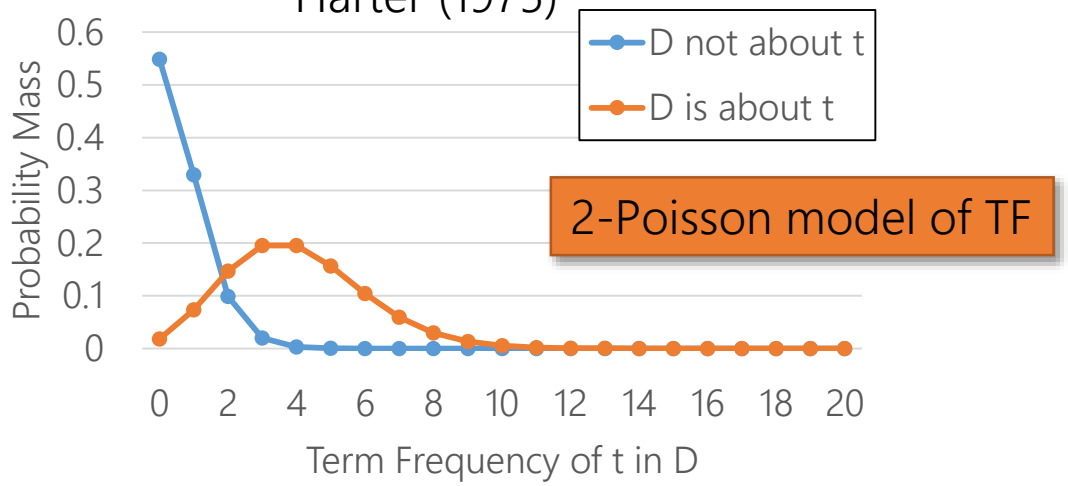
RSJ naïve Bayes model of IDF

$$w_i^{\text{RSJ}} = \log \frac{(r_i + 0.5)(N - R - n_i + r_i + 0.5)}{(n_i - r_i + 0.5)(R - r_i + 0.5)}$$

$$w_i^{\text{IDF}} = \log \frac{N - n_i + 0.5}{n_i + 0.5}$$

- **Term frequency**

Harter (1975)



2-Poisson model of TF

- Adjust TF model for **doc length**

Rank D by: $\sum_{t \in Q} TF(t, D) * IDF(t)$

Microsoft

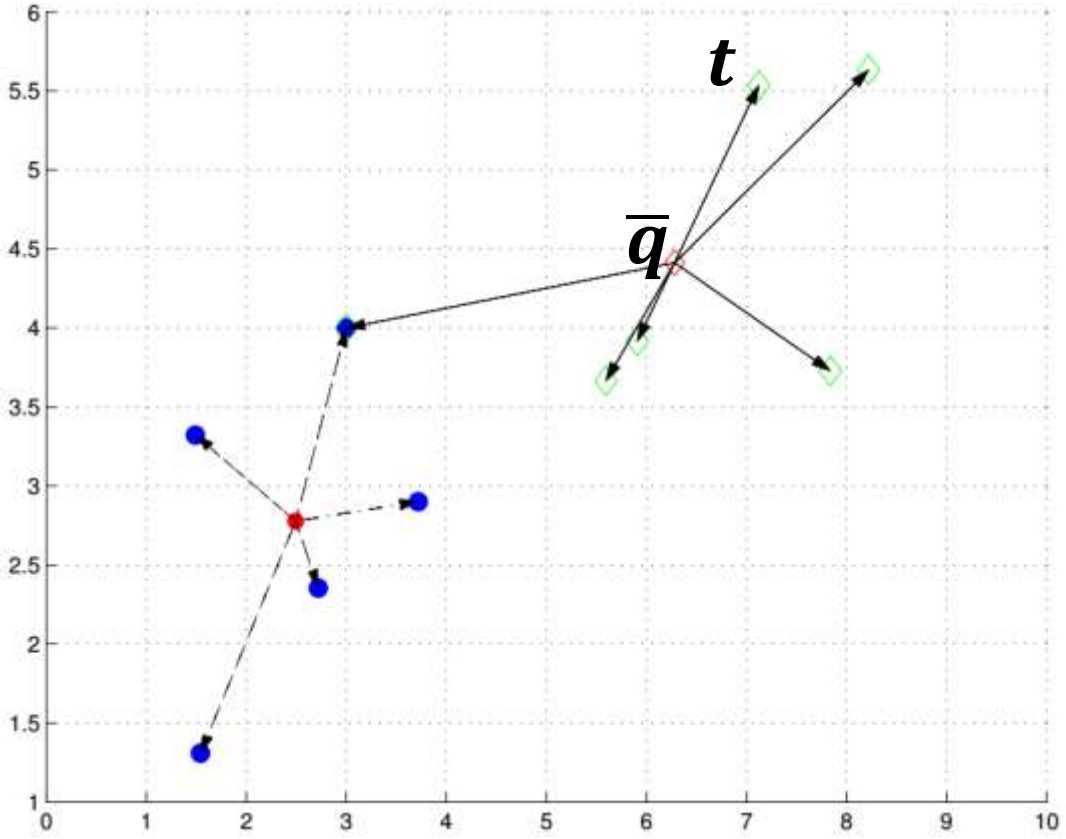
How to incorporate embeddings

A. Extend traditional IR models
   - Term weighting, language model smoothing, translation of vocab

B. Expand query using embeddings (followed by non-neural IR)
   - Add words similar to the query

C. IR models that work in the embedding space
   - Centroid distance, word mover's distance

Term weighting using word embeddings



$y = \frac{r}{R}$ (term recall)

- Fraction of positive docs with t
- The r and R were missing in RSJ

$\boldsymbol{x} = \boldsymbol{t} - \bar{\boldsymbol{q}}$

- $\boldsymbol{t}$ is embedding of t
- $\bar{\boldsymbol{q}}$ is centroid of all query terms

- Weight TF-IDF using $\hat{y}$

Zheng and Callan. Learning to reweight terms with distributed representations. SIGIR 2015

Traditional IR model: Query likelihood

- Language modeling approach to IR is quite extensible

$$\mathrm{p}(d|q) \approx p(q|d)p(d)$$

- $\mathrm{p}(d)$ can be assumed uniform across docs
- $\mathrm{p}(q|d) = \prod_{w \in q} p(w|d)$ depends on modeling the document
  - Frequent words in d are more likely (term frequency)
  - Smoothing according to the corpus (plays the role of IDF)
  - Various ways of dealing with document length normalization

Ponte and Croft. A language modeling approach to information retrieval. SIGIR 1998
Zhai and Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. SIGIR 2001

Generalized Language Model

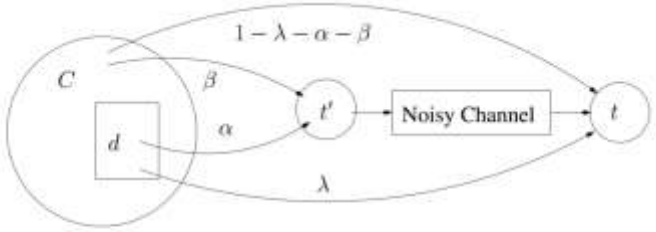


Figure 1: Schematics of generating a query term t in our proposed Generalized Language Model (GLM). GLM degenerates to LM when $\alpha = \beta = 0$.

Compares query term with every document term

$$P(t|d) = \lambda P(t|d) + \alpha \sum_{t' \in d} P(t,t'|d)P(t') +$$

$$\beta \sum_{t' \in N_t} P(t,t'|C)P(t') + (1-\lambda-\alpha-\beta)P(t|C)$$

$$P(t,t'|d) = \frac{sim(t',t)}{\Sigma(d)} \frac{tf(t',d)}{|d|}$$

Microsoft

Ganguly, Roy, Mitra, and Jones. Word embedding based generalized language model for information retrieval. SIGIR 2015

Neural Language Translation Model

NTLM - cbow

w = insider		w = trading	
u	$p(w\|u)$	u	$p(w\|u)$
insider	0.285	trading	0.216
fraud	0.104	traders	0.103
drexel	0.095	market	0.094
criminal	0.084	stock	0.090
securities	0.084	markets	0.085
racketeering	0.084	futures	0.084

NTLM - skipgram

w = insider		w = trading	
u	$p(w\|u)$	u	$p(w\|u)$
insider	0.169	trading	0.164
fraud	0.102	traders	0.103
drexel	0.099	futures	0.099
securities	0.096	stock	0.097
racketeering	0.093	exchange	0.094
bribery	0.091	market	0.093

Translation probability from document term u to query term w

$$p_t(w|d) = \sum_{u \in d} p_t(w|u)p(u|d)$$

$$p_{cos}(u|w) = \frac{cos(u,w)}{\sum_{u' \in V} cos(u',w)}$$

Considers all query-document term pairs

Based on: Berger and Lafferty. "Information retrieval as statistical translation." SIGIR 1999

Zuccon, Koopman, Bruza, and Azzopardi. "Integrating and evaluating neural word embeddings in information retrieval." Australasian Document Computing Symposium 2015

B. Query expansion

Both passages have same number of gold query matches.

Yet non-query green matches can be a good evidence of *aboutness*.

We need methods to consider non-query terms. Traditionally: automatic query expansion.

Query: Albuquerque

Albuquerque is the most populous city in the U.S. state of New Mexico. The high-altitude city serves as the county seat of Bernalillo County, and it is situated in the central part of the state, straddling the Rio Grande. The city population is 557,169 as of the July 1, 2014, population estimate from the United States Census Bureau, and ranks as the 32nd-largest city in the U.S. The Metropolitan Statistical Area (or MSA) has a population of 902,797 according to the United States Census Bureau's most recently available estimate for July 1, 2013.

(a)

Allen suggested that they could program a BASIC interpreter for the device; after a call from Gates claiming to have a working interpreter, MITS requested a demonstration. Since they didn't actually have one, Allen worked on a simulator for the Altair while Gates developed the interpreter. Although they developed the interpreter on a simulator and not the actual device, the interpreter worked flawlessly when they demonstrated the interpreter to MITS in Albuquerque, New Mexico in March 1975; MITS agreed to distribute it, marketing it as Altair BASIC.

(b)

Example from Mitra et al., 2016

Query expansion using w2v

Identify expansion terms using w2v cosine similarity

3 different strategies: pre-retrieval, post-retrieval, and pre-retrieval incremental

$$P(w|Q_{exp}) = \alpha P(w|Q) \; + \; (1-\alpha)\frac{Sim(w,Q)}{\sum_{w \in Q_{exp}} Sim(w,Q)}$$

Beats no expansion, but does not beat non-neural expansion

Roy, Paul, Mitra and Garain. Using Word embeddings for automatic query expansion. Neu-IR Workshop 2016

Query expansion using w2v

- Related paper. Distance δ has a sigmoid transform. First model:

$$p(w|\theta_Q) \propto \sum_{w' \in Q} \frac{\delta(w, w')}{\sum_{w'' \in V} \delta(w'', w')} \times \frac{c(w', Q)}{|Q|}$$

- Their second model uses the first round of retrieval

$$p_{sem}(Q|w, D) = \prod_{i=1}^{k} p_{sem}(q_i|w, D) \stackrel{\Delta}{=} \prod_{i=1}^{k} \frac{\delta(q_i, w)c(q_i, D)}{Z}$$

  - Can beat non-neural expansion

Microsoft

Zamani and Croft. Embedding-based query language models. *International Conference on the Theory of Information Retrieval 2016*
Lavrenko and Croft. Relevance based language models. SIGIR 2001

Optimizing the query vector

$$\vec{q}^* = \arg\max_{\vec{q}} \sum_{w \in V} p(w|\theta_q) \log p(w|\vec{q})$$

$$\arg\max_{\vec{q}} \sum_{w \in V} p(w|\theta_q) \log \delta(\vec{w}, \vec{q})$$

$$p(w|\theta_q^*) = \alpha \; p_{mle}(w|\theta_q) + (1 - \alpha) \; p(\vec{w}|\vec{q})$$

Microsoft

Zamani and Croft. Estimating embedding vectors for queries. *International Conference on the Theory of Information Retrieval 2016*

Global vs. local embedding spaces

global	local
cutting	tax
squeeze	deficit
reduce	vote
slash	budget
reduction	reduction
spend	house
lower	bill
halve	plan
soften	spend
freeze	billion

Figure 3: Terms similar to 'cut' for a word2vec model trained on a general news corpus and another trained only on documents related to 'gasoline tax'.

Train w2v on documents from first round of retrieval

Fine-grained word sense disambiguation

A large number of embedding spaces can be cached in practice

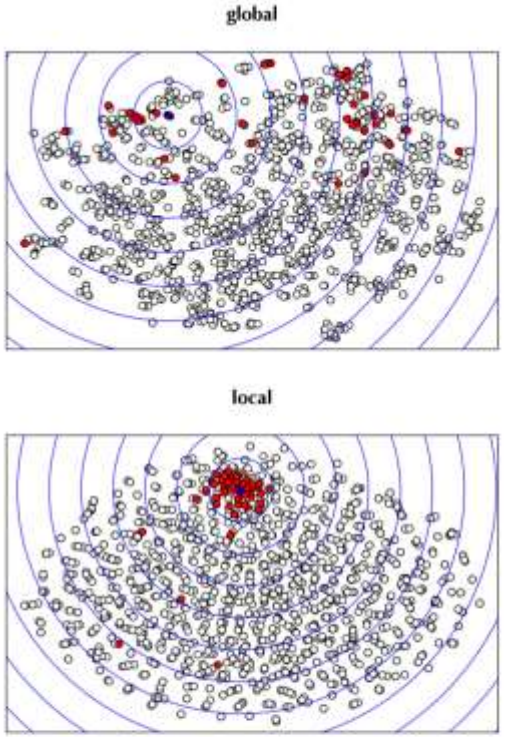


Figure 5: Global versus local embedding of highly relevant terms. Each point represents a candidate expansion term. Red points have high frequency in the relevant set of documents. White points have low or no frequency in the relevant set of documents. The blue point represents the query. Contours indicate distance from the query.

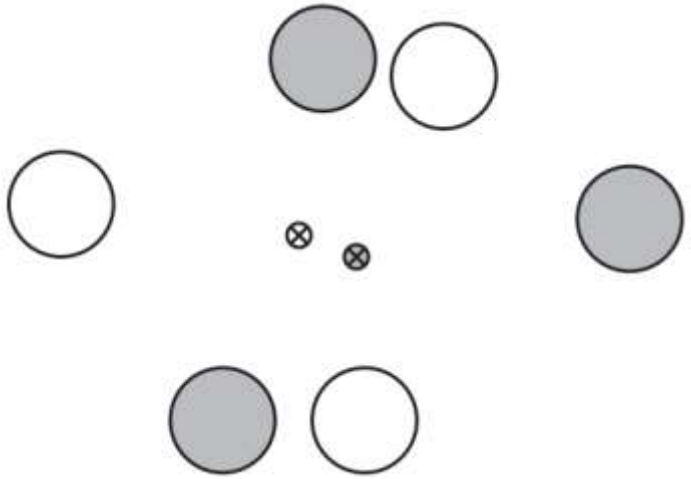
(Diaz et al., 2016)

C. IR models in the embedding space

- Q: Bag of word vectors
- D: Bag of word vectors
- Considerations:
  - Deal with variable length of Q and D
  - Match all pairs? Do some alignment of Q and D?

Comparing short texts

- Weighted semantic network
  - Related to word alignment
  - Each word in longer text is connected to its most similar
  - BM25-like edge weighting
- Generates features for supervised learning of short text similarity

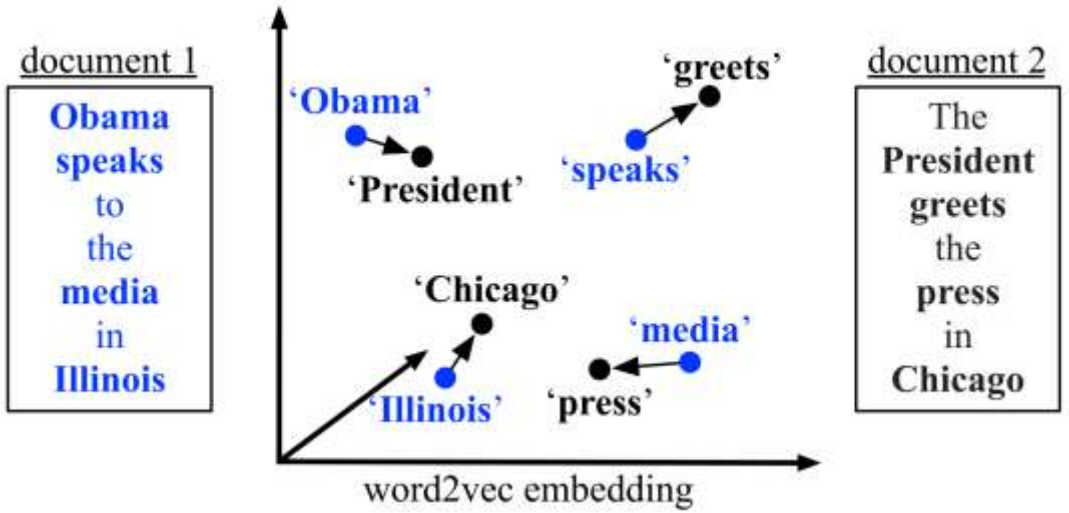


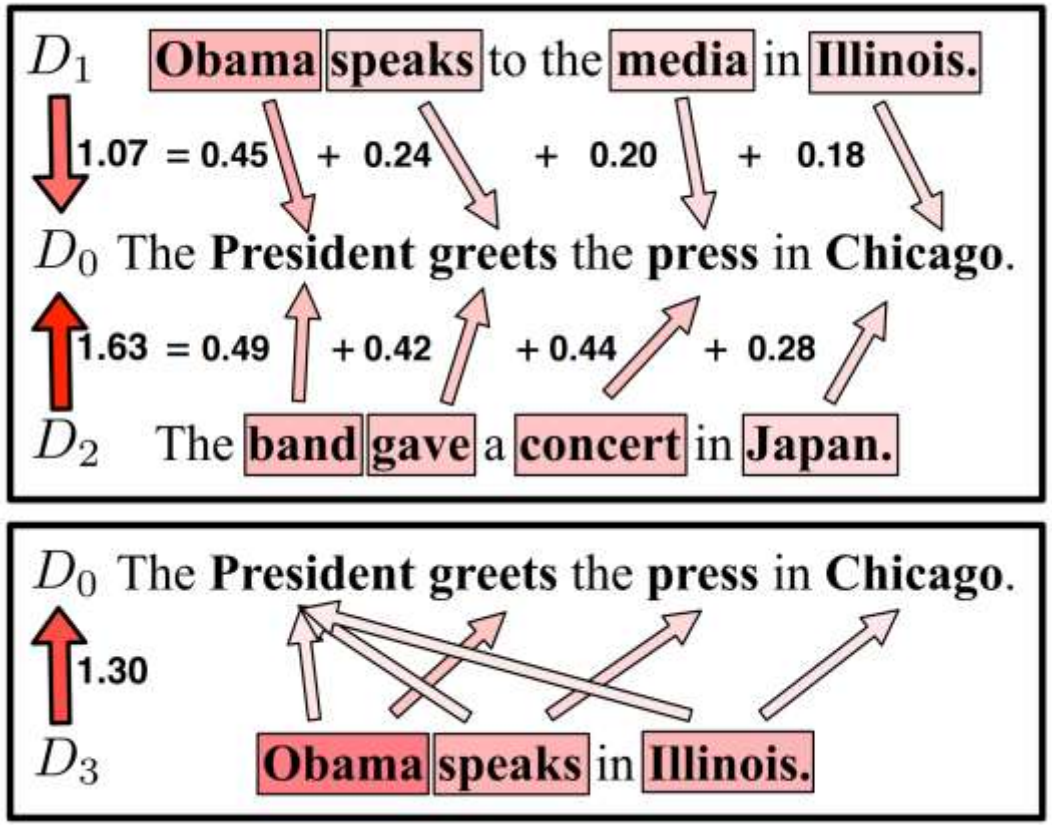
$$f_{sts}(s_l, s_s) = \sum_{w \in s_l} \text{IDF}(w) \cdot \frac{sem(w, s_s) \cdot (k_1 + 1)}{sem(w, s_s) + k_1 \cdot (1 - b + b \cdot \frac{|s_s|}{\text{avgsl}})}$$

$$sem(w, s) = \max_{w' \in s} f_{sem}(w, w').$$

Kenter and de Rijke. Short text similarity with word embeddings. CIKM 2015

Word Mover's Distance





Sparse normalized TF vectors: $|\boldsymbol{d}| = |\boldsymbol{d}'| = 1$
Sparse flow matrix: $\sum_j T_{ij} = d_i \quad \sum_i T_{ij} = d'_j$
Minimize cost: $\sum_{i,j} T_{ij} c(i,j)$ ← word2vec

Microsoft

Kusner, Sun, Kolkin and Weinberger. From Word Embeddings To Document Distances. ICML 2015
See also: Huang, Guo, Kusner, Sun, Weinberger and Sha. Supervised Word Mover's Distance. NIPS 2016

Bounds on word mover's distance

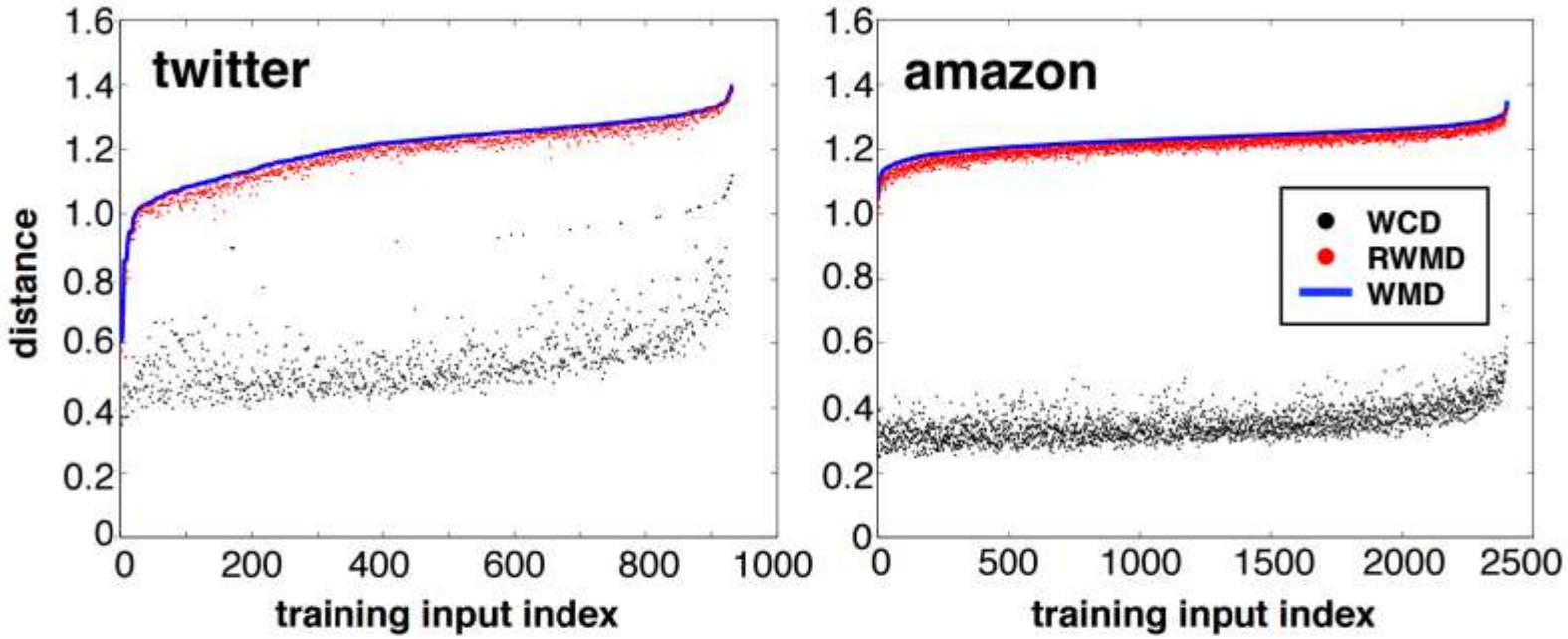


Figure 6. The WCD, RWMD, and WMD distances (sorted by WMD) for a random test query document.

WCD <= RWMD <= WMD
WCD: Word centroid distance
RWMD: Relaxed WMD

Prefetch and prune algorithm:

- Sort by WCD
- Compute WMD on top-k
- Continue, using RWMD to prune 95% of WMD

Microsoft

Kusner, Sun, Kolkin and Weinberger. From Word Embeddings To Document Distances. ICML 2015

Centroid and related techniques

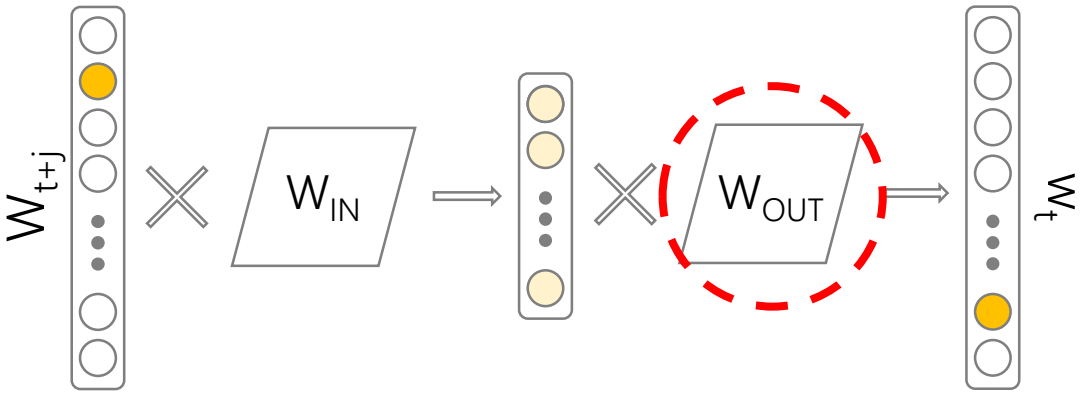
- Goal: Bag of vectors → Single fixed size vector
- Centroid, weighted centroid (or sum [Vulic and Moens, 2015])
- Aggregate using a Fisher Kernel
  - Clinchant and Perronnin. Aggregating continuous word embeddings for information retrieval. (ACL) Workshop on Continuous Vector Space Models and their Compositionality, 2013
  - Using Fisher Kernel from Jaakkola and Haussler (1999)

What if I told you everyone using w2v is throwing half the model away?

Two sets of embeddings are trained (W_{IN} and W_{OUT})

But W_{OUT} is generally discarded

IN-OUT dot product captures log prob. of co-occurrence



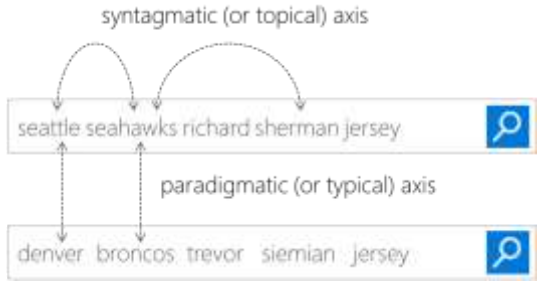
(Mitra et al., 2016)

Notions of Similarity

IN-OUT captures more Topical notion of similarity than IN-IN and OUT-OUT

Effect is exaggerated when embeddings are trained on short text (e.g., queries)

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



(Mitra et al., 2016)

Dual Embedding Space Model

Query and document representation:
Centroid of all words, including repetition

Compare query terms with every document term

$$DESM_{IN-OUT}(Q, D) = \frac{1}{|Q|} \sum_{q_i \in Q} \frac{q_{IN,i}^T \overline{D_{OUT}}}{\|q_{IN,i}\| \|\overline{D_{OUT}}\|}$$

Equivalent to taking centroid of all term vectors

$$\overline{\mathbf{D}} = \frac{1}{|D|} \sum_{\mathbf{d}_j \in D} \frac{\mathbf{d}_j}{\|\mathbf{d}_j\|}$$

Combine with traditional retrieval model

$$MM(Q, D) = \alpha DESM(Q, D) + (1 - \alpha) BM25(Q, D)$$
$$\alpha \in \mathbb{R}, 0 \leq \alpha \leq 1$$

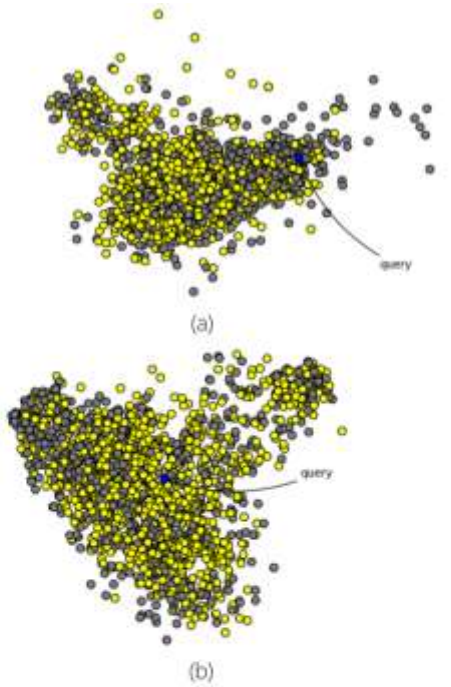


Figure 1: A two dimensional PCA projection of the 200-dimensional embeddings. Relevant documents are yellow, irrelevant documents are grey, and the query is blue. To visualize the results of multiple queries at once, before dimensionality reduction we centre query vectors at the origin and represent documents as the difference between the document vector and its query vector. (a) uses IN word vector centroids to represent both the query and the documents. (b) uses IN for the queries and OUT for the documents, and seems to have a higher density of relevant documents near the query.

(Mitra et al., 2016)

Telescoping

- DESM also works well when "telescoping" i.e. reranking a small set
  - For example, DESM can confuse Oxford and Cambridge
  - Bing rarely makes the Oxford-Cambridge mistake
  - The upstream ranker (Bing) saves DESM from making that mistake
  - If DESM sees more documents, it may make the mistake

Matveeva, Burges, Burkard, Laucius, and Wong. High accuracy retrieval with multiple nested ranker. SIGIR 2006

Because Cambridge is not "an African even-toed ungulate mammal"

Passage about the city of Cambridge

the city of cambridge is a university city and the county town of cambridgeshire, england. it lies in east anglia, on the river cam, about 50 miles (80 km) north of london. according to the united kingdom census 2011, its population was (including students. this makes cambridge the second largest city in cambridgeshire after peterborough, and the 54th largest in the united kingdom. there is archaeological evidence of settlement in the area during the bronze age and roman times; under viking rule cambridge became an important trading centre. the first town charters were granted in the 12th century, although city status was not conferred until 1951.

Passage about the city of Oxford

oxford is a city in the south east region of england and the county town of oxfordshire. with a population of it is the 52nd largest city in the united kingdom, and one of the fastest growing and most ethnically diverse. oxford has a broad economic base. its industries include motor manufacturing, education, publishing and a large number of information technology and businesses, some being academic offshoots. the city is known worldwide as the home of the university of oxford, the oldest university in the world. buildings in oxford demonstrate examples of every english architectural period since the arrival of the saxons, including the radcliffe camera. oxford is known as the city of dreaming spires, a term coined by poet matthew arnold.

Passage about giraffes

the giraffe (giraffa camelopardalis) is an african ungulate mammal, the tallest living terrestrial animal and the largest ruminant. its species name refers to its shape and its colouring. its chief distinguishing characteristics are its extremely long neck and legs, its , and its distinctive coat patterns. it is classified under the family , along with its closest extant relative, the okapi. the nine subspecies are distinguished by their coat patterns. the scattered range of giraffes extends from chad in the north to south africa in the south, and from niger in the west to somalia in the east. giraffes usually inhabit savannas, grasslands, and open woodlands.

Passage about giraffes, but 'giraffe' is replayed by 'Cambridge'

the cambridge (giraffa camelopardalis) is an african ungulate mammal, the tallest living terrestrial animal and the largest ruminant. its species name refers to its shape and its colouring. its chief distinguishing characteristics are its extremely long neck and legs, its , and its distinctive coat patterns. it is classified under the family , along with its closest extant relative, the okapi. the nine subspecies are distinguished by their coat patterns. the scattered range of giraffes extends from chad in the north to south africa in the south, and from niger in the west to somalia in the east. giraffes usually inhabit savannas, grasslands, and open woodlands.

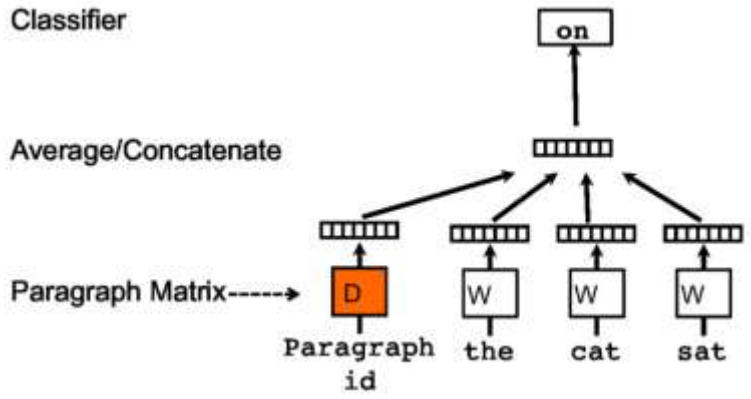
Microsoft

https://github.com/bmitra-msft/Demos/blob/master/notebooks/DESM.ipynb

Paragraph2vec

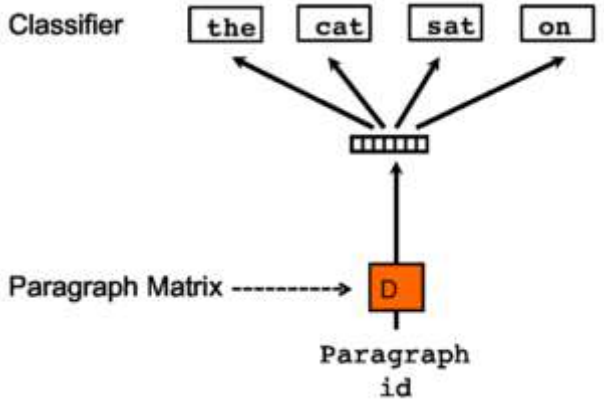
PV-DM

A mix of LSA and and w2v style training data, less sparse than training just on paragraph ID



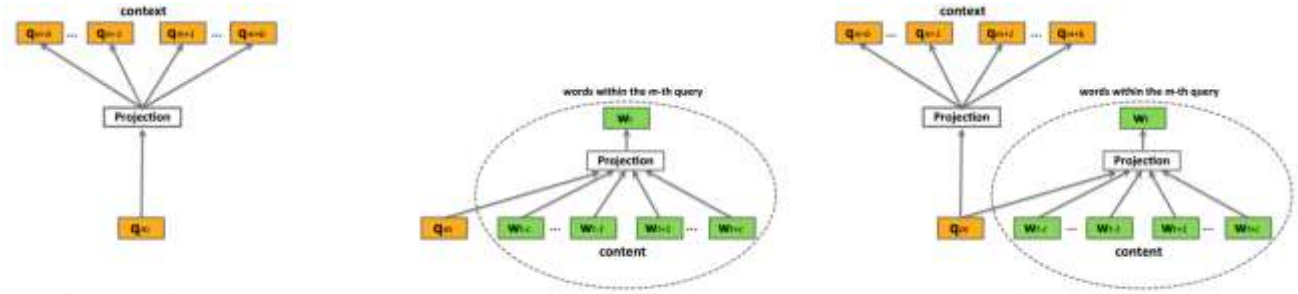
PV-DBOW

Effectively modelling the same relationship as LSA (factorizing word-document matrix)



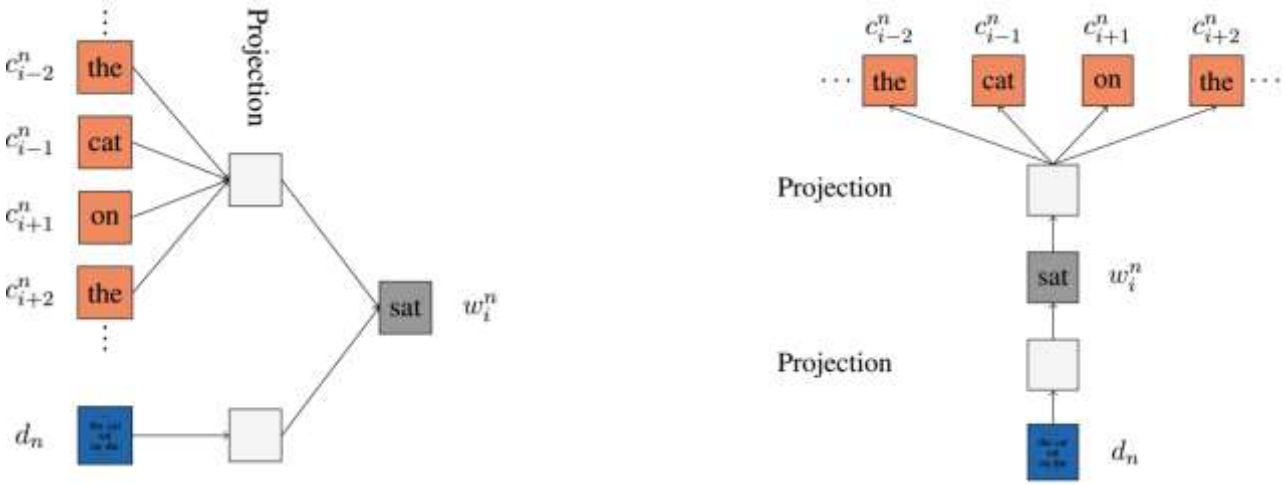
Le and Mikolov. Distributed Representations of Sentences and Documents. ICML 2014

Models of similar flavour



(a) context2vec (b) content2vec (c) context-content2vec

Grbovic, Djuric, Radosavljevic, Silvestri and Bhamidipati.
Context-and content-aware embeddings for query rewriting in sponsored search.
SIGIR 2015.



Sun, Guo, Lan, Xu and Cheng
Learning Word Representations by Jointly Modeling Syntagmatic and Paradigmatic Relations
ACL 2015

Adapting PV-DBOW for IR

Negative sampling using IDF instead of corpus frequency

L2 regularization constraint on the norm to avoid over-fitting on short documents

Reduce sparseness by predicting context words (similar to PV-DM?)

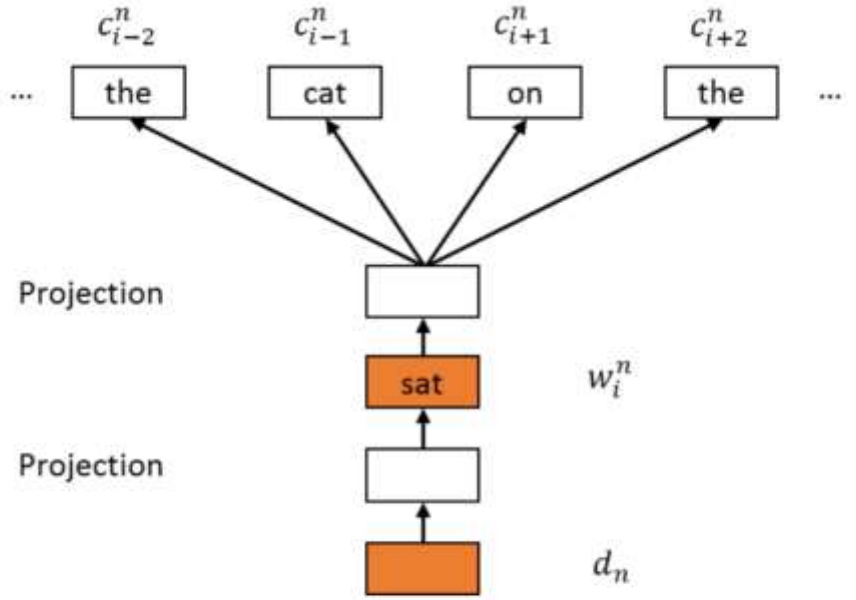


Figure 5: The structure of two-layer PV-DBOW. The document is trained to predict the observed word and then the observed word is trained to predict its context.

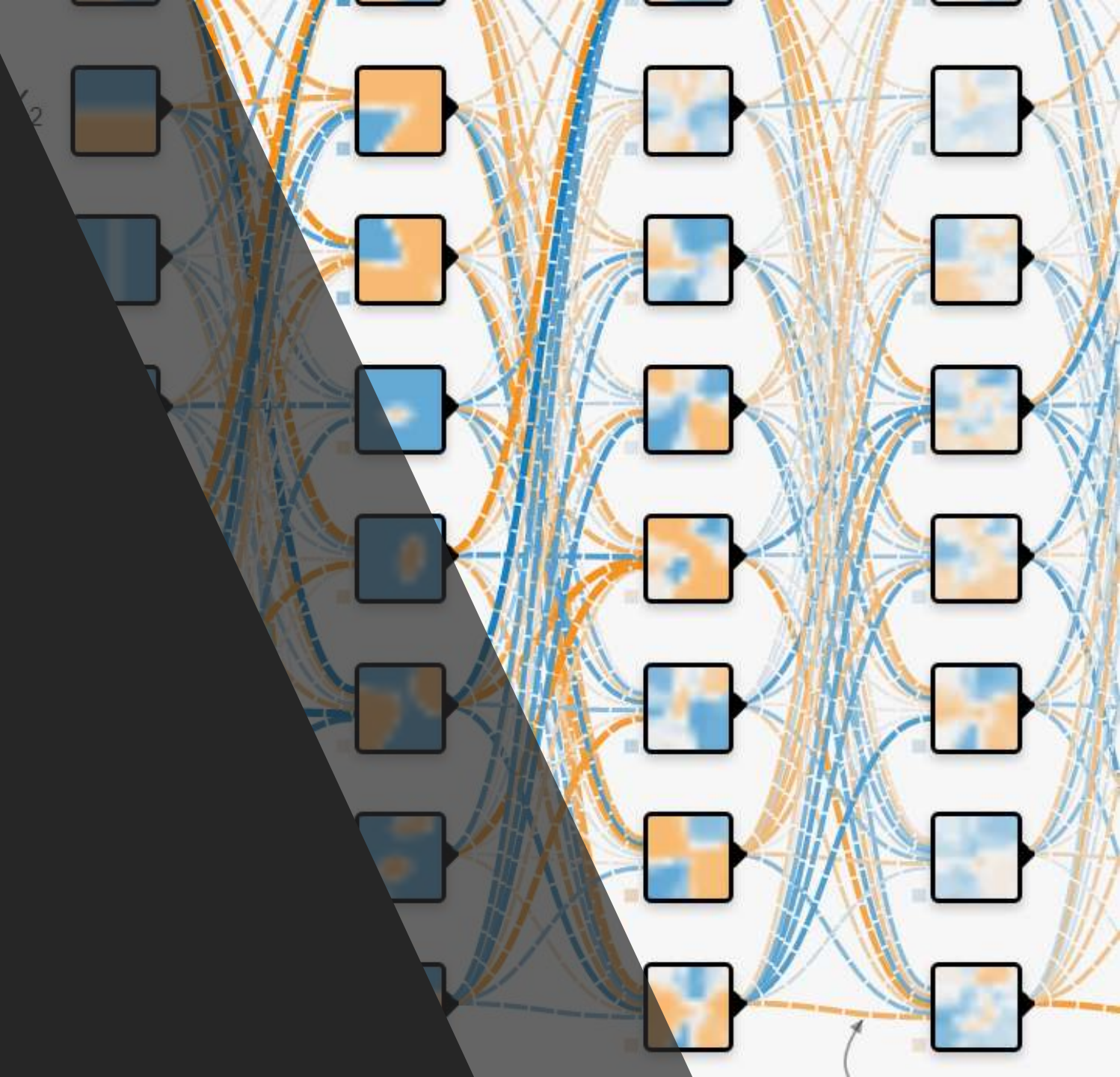
Ai, Yang, Guo and Croft. Analysis of the Paragraph Vector Model for Information Retrieval. ICTIR 2016

Discussion of embeddings in IR

- The right mix of lexical and semantic matching is important
  - In non-telescoping settings, need to combine with lexical IR
- Many IR models do some sort of vector comparison of Q-D words
- Open question: Which notion of similarity is best for each IR model
  - Typical vs Topical vs some mix
- These methods seem promising if:
  - High-quality embeddings domain-appropriate embeddings available
  - No large-scale supervised IR data available
- If large-scale supervised IR data is available… (after the break)

Chapter 4

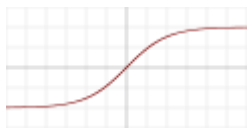
Deep Neural Networks



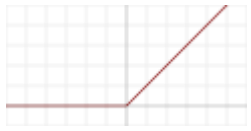
A primer on neural networks

Chains of parameterized linear transforms (e.g., multiply weight, add bias) followed by non-linear functions (σ)

Popular choices for σ:



Tanh

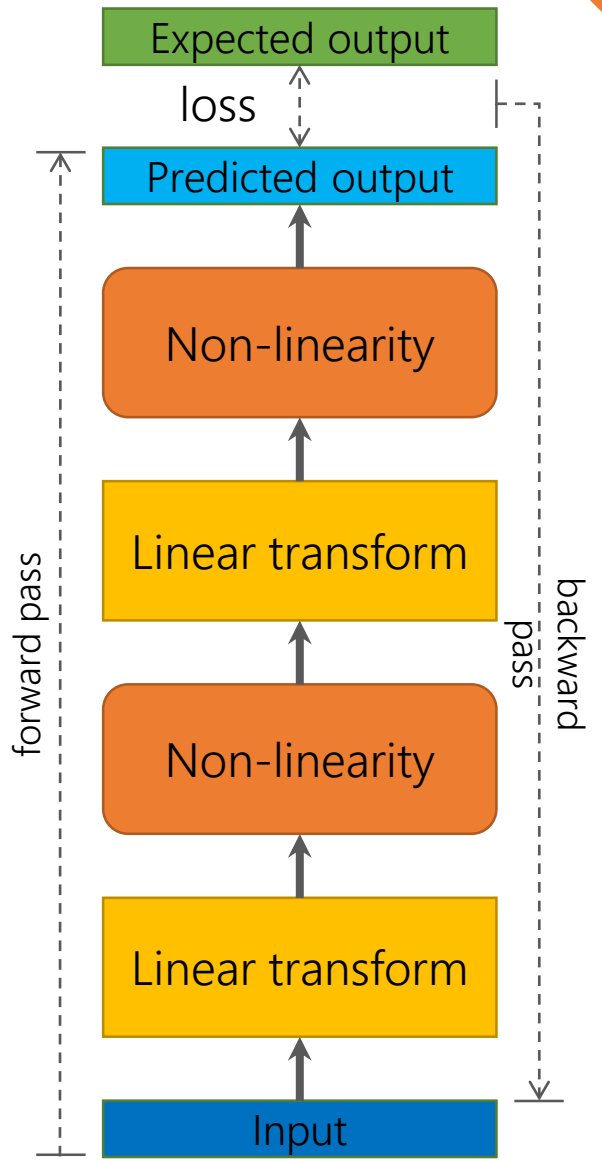


ReLU

Parameters trained using backpropagation

E2E training over millions of samples in batched mode

Many choices of architecture and hyper-parameters



Visual motivation for hidden units

Consider the following "toy" challenge for classifying tech queries:

Vocab: {surface, kerberos, book, library}

Labels:

"surface book", "kerberos library" ✓

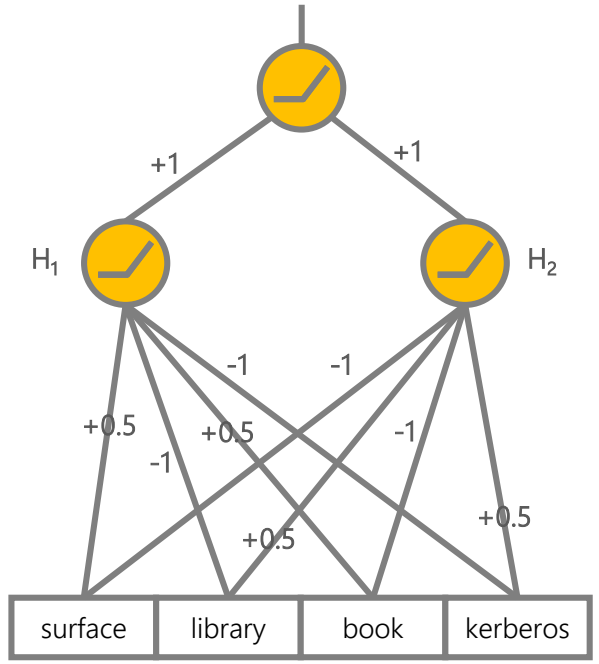
"kerberos surface", "library book" ✗

Or more succinctly…

Input features				Label
surface	kerberos	book	library	
1	0	1	0	✓
1	1	0	0	✗
0	1	0	1	✓
0	0	1	1	✗

can't separate using a linear model!

But let's consider a tiny neural network with one hidden layer…



Microsoft

Visual motivation for hidden units

Consider the following "toy" challenge for classifying tech queries:

Vocab: {surface, kerberos, book, library}

Labels:

"surface book", "kerberos library" ✓

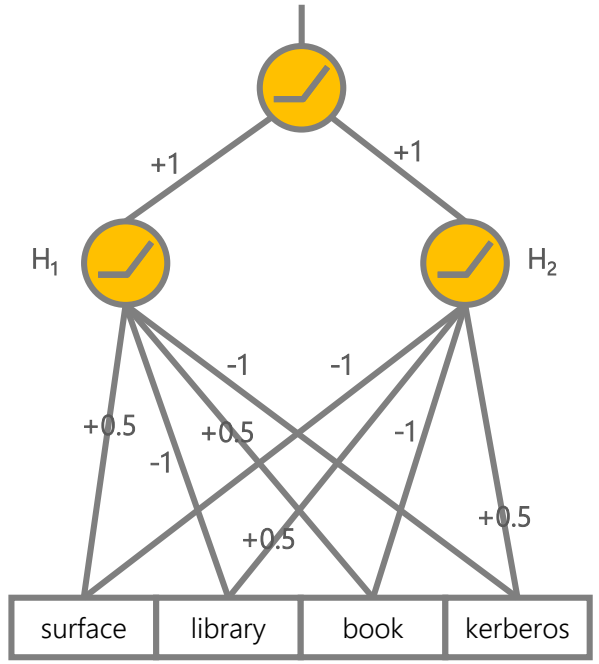
"kerberos surface", "library book" ✗

Or more succinctly…

Input features				Hidden layer		Label
surface	kerberos	book	library	H_1	H_2	
1	0	1	0	1	0	✓
1	1	0	0	0	0	✗
0	1	0	1	0	1	✓
0	0	1	1	0	0	✗

can separate using a linear model!

But let's consider a tiny neural network with one hidden layer…



Microsoft

Why adding depth helps

Deeper networks can split the input space in many (non-independent) linear regions than shallow networks

Corollary 5. *A rectifier neural network with n_0 input units and L hidden layers of width $n \geq n_0$ can compute functions that have $\Omega\left(\left(n/n_0\right)^{(L-1)n_0} n^{n_0}\right)$ linear regions.*

Thus we see that the number of linear regions of deep models grows exponentially in L and polynomially in n, which is much faster than that of shallow models with nL hidden units. Our result

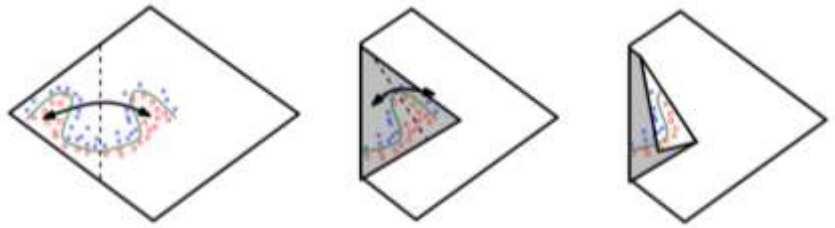
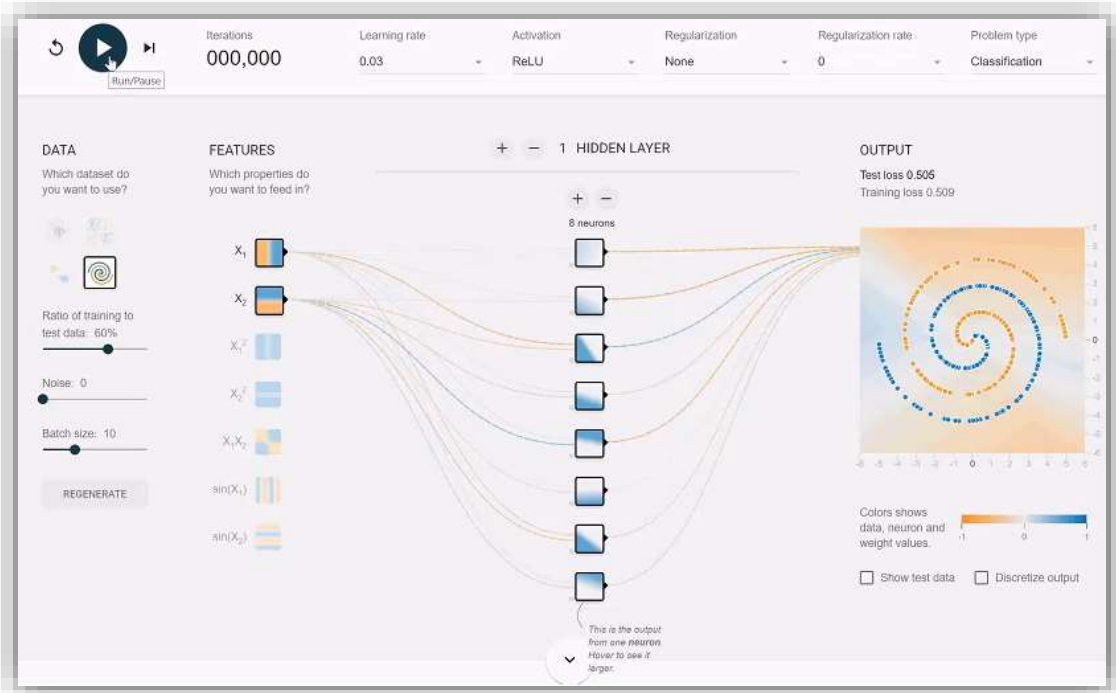
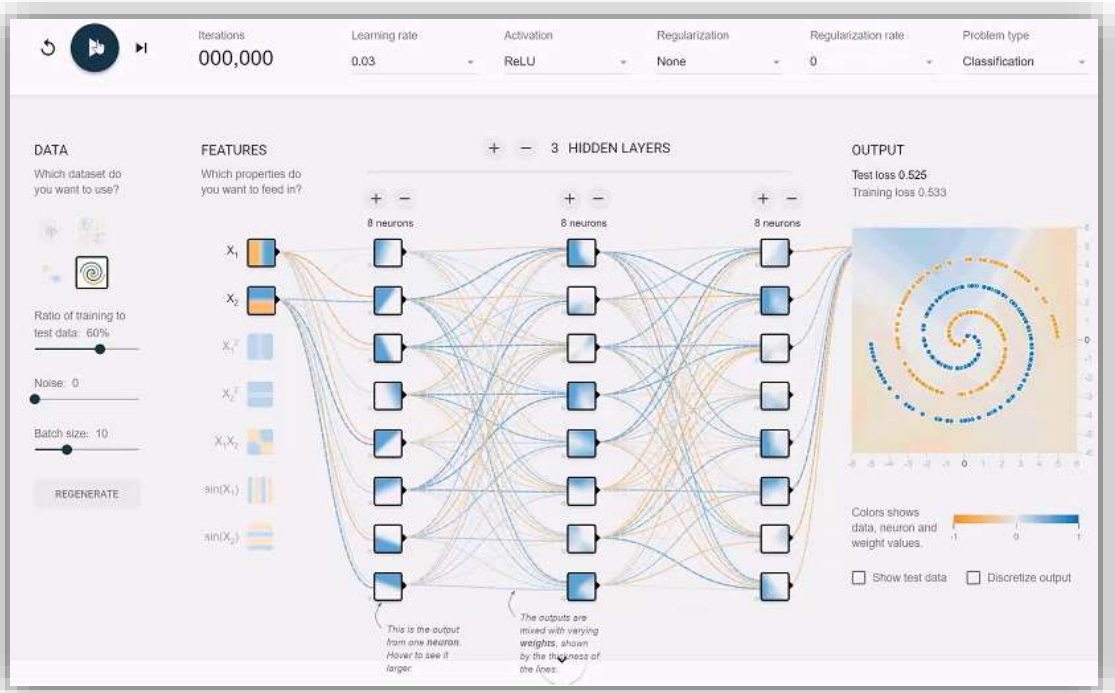


Figure 3: Space folding of 2-D space in a non-trivial way. Note how the folding can potentially identify symmetries in the boundary that it needs to learn.

Montúfar, Pascanu, Cho and Bengio. On the number of linear regions of deep neural networks NIPS 2014

Why adding depth helps

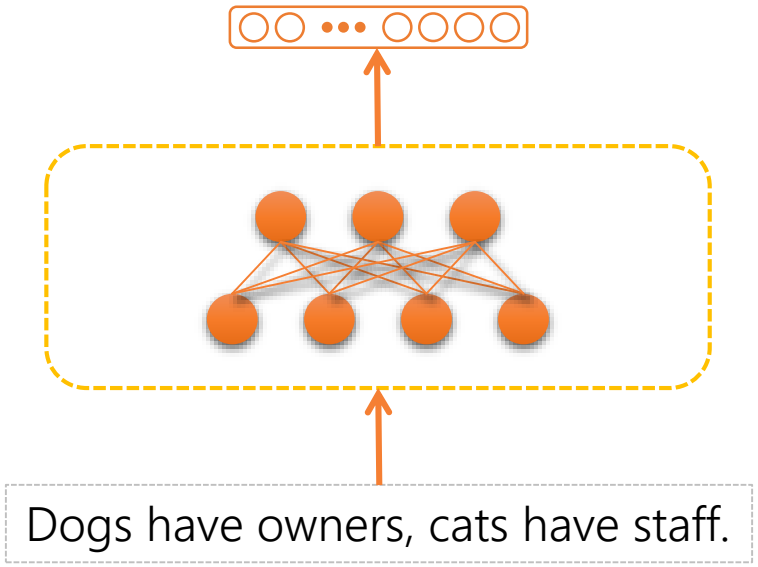




Microsoft

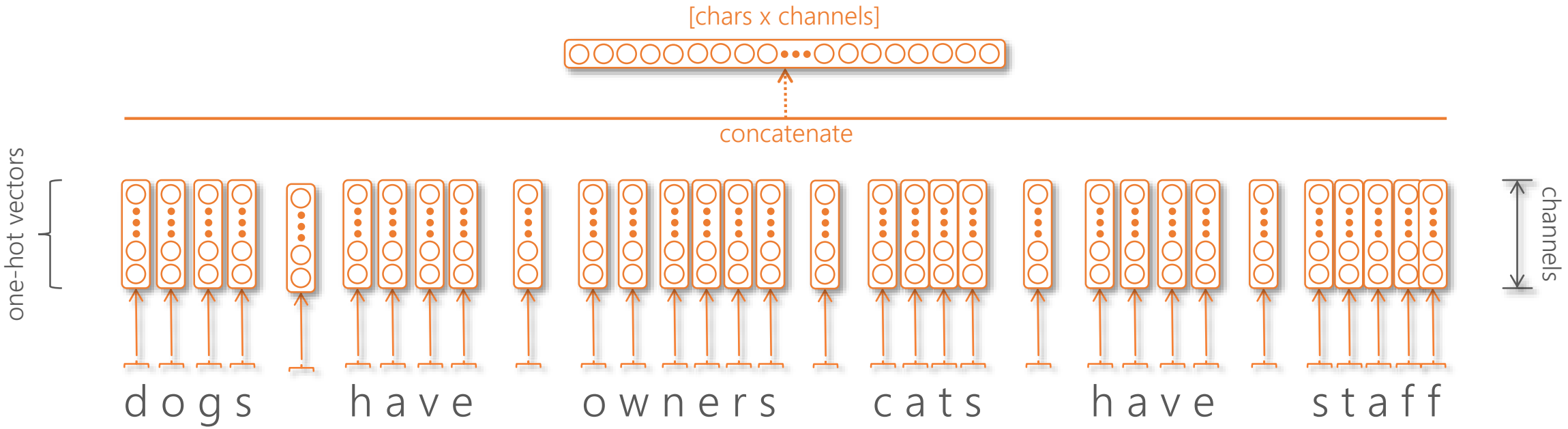
http://playground.tensorflow.org

Neural models for text



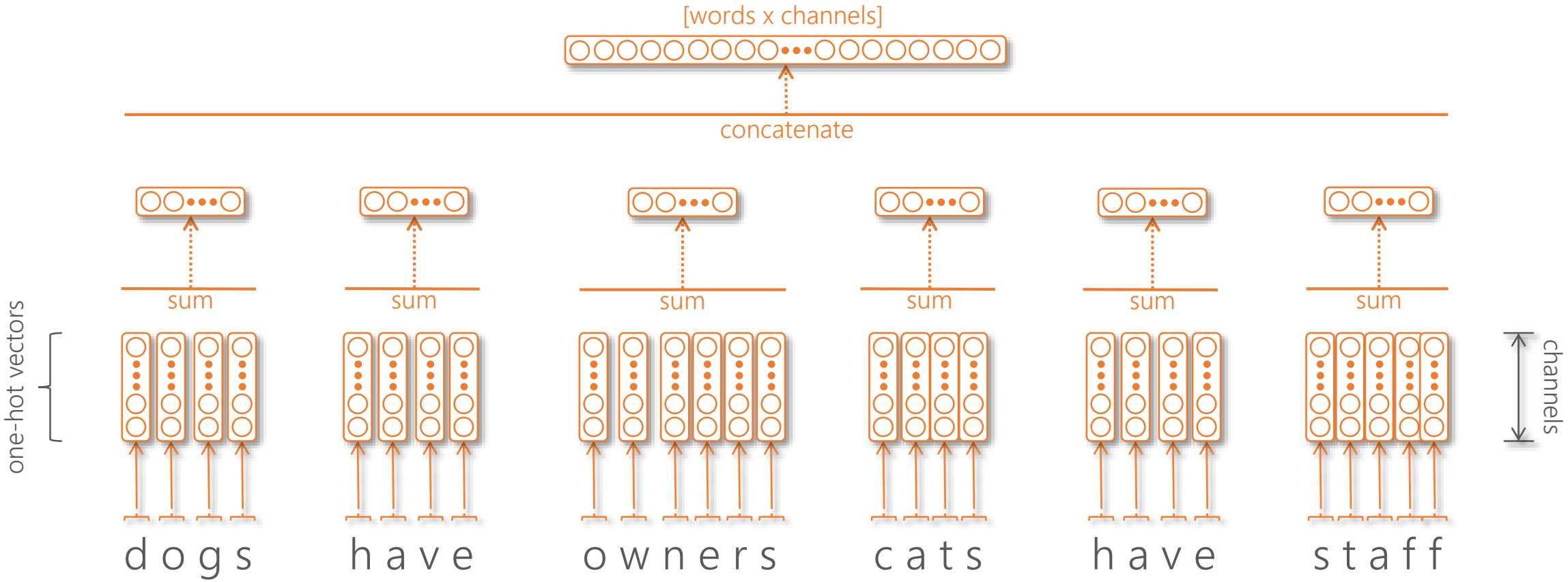
But how do you feed text to a neural network?

Local representations of input text



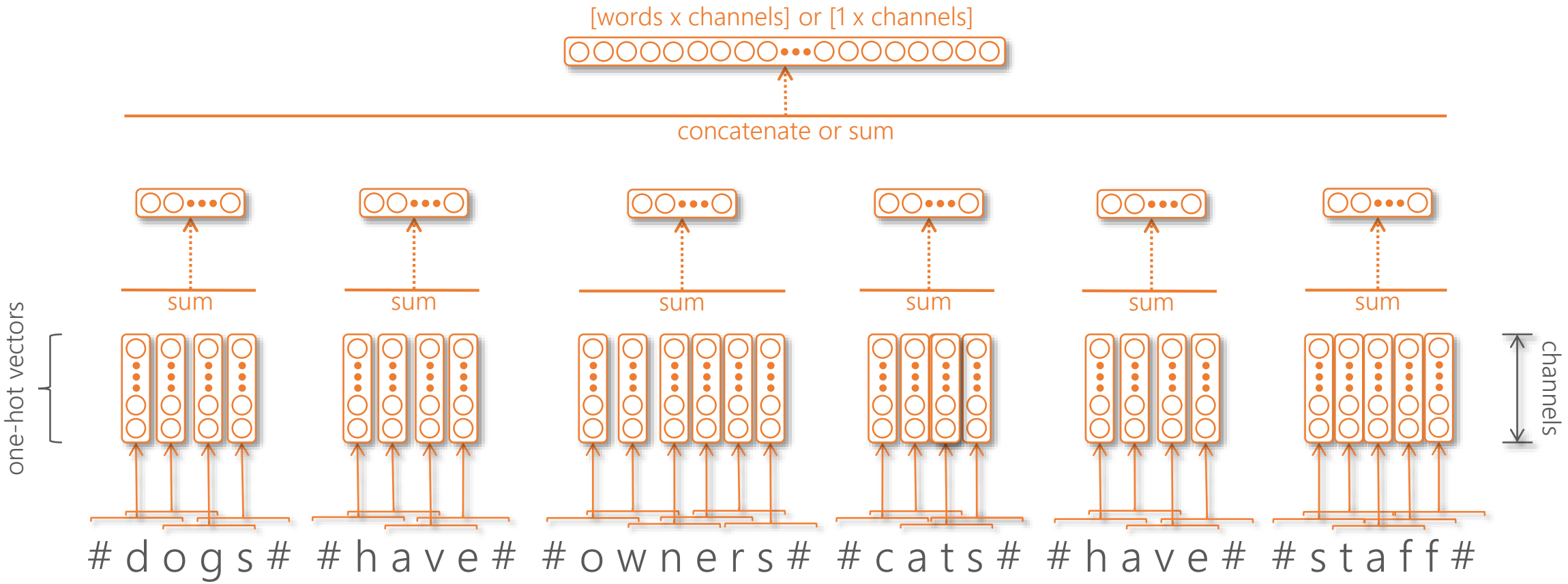
Char-level models

Local representations of input text



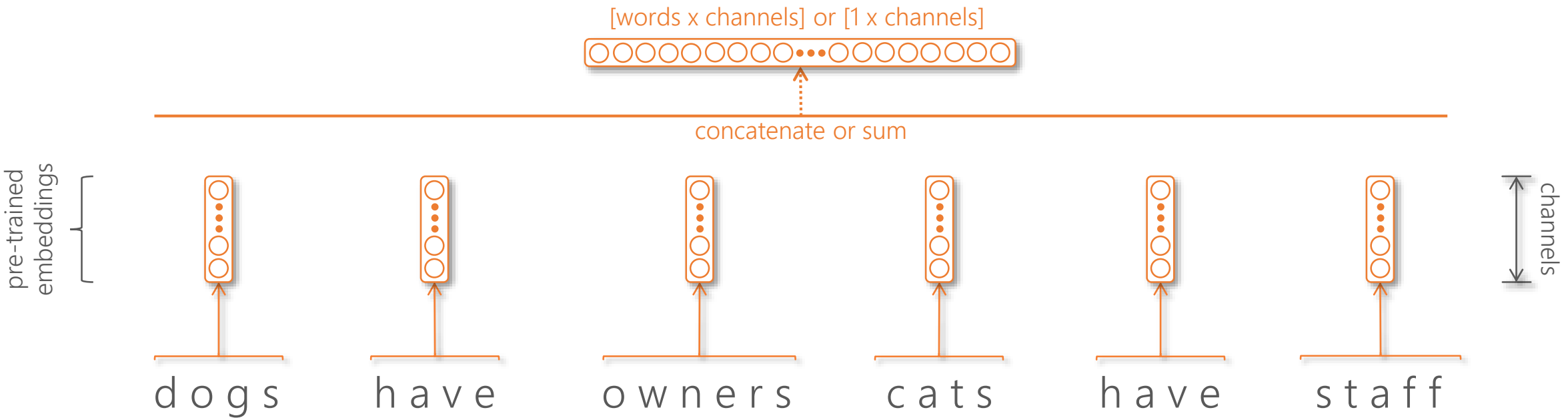
Word-level models w/ bag-of-chars per word

Local representations of input text



Word-level models w/ bag-of-trigraphs per word

Local representations of input text



Word-level models w/ pre-trained embeddings

Shift-invariant neural operations

Detecting a pattern in one part of input space is same as detecting it in another

(also applies to sequential inputs, and inputs with dims >2)

Leverage redundancy by operating on a moving window over whole input space and then aggregate

The repeatable operation is called a kernel, filter, or cell

Aggregation strategies leads to different architectures

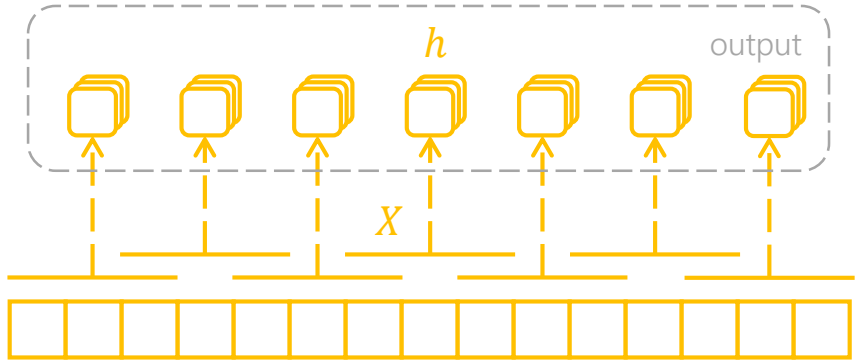
Convolution

Move the window over the input space each time applying the same cell over the window

A typical cell operation can be,

$$h = \sigma(WX + b)$$

Full Input	[words x in_channels]
Cell Input	[window x in_channels]
Cell Output	[1 x out_channels]
Full Output	[1 + (words – window) / stride x out_channels]



Pooling

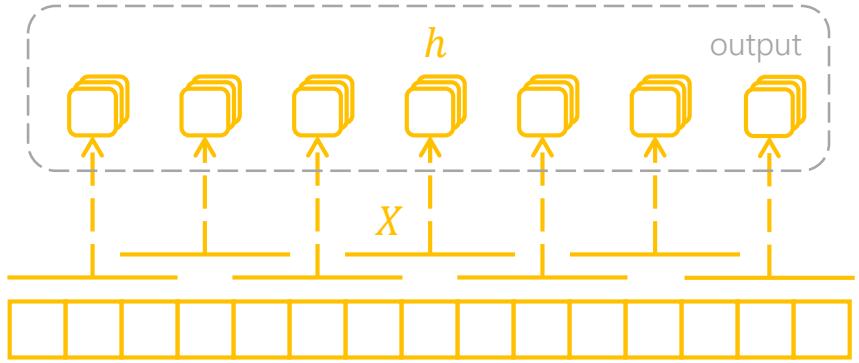
Move the window over the input space each time applying an aggregate function over each dimension in within the window

$$h_j = max_{i \in win}(X_{i,j}) \quad or \quad h_j = avg_{i \in win}(X_{i,j})$$

max -pooling

average -pooling

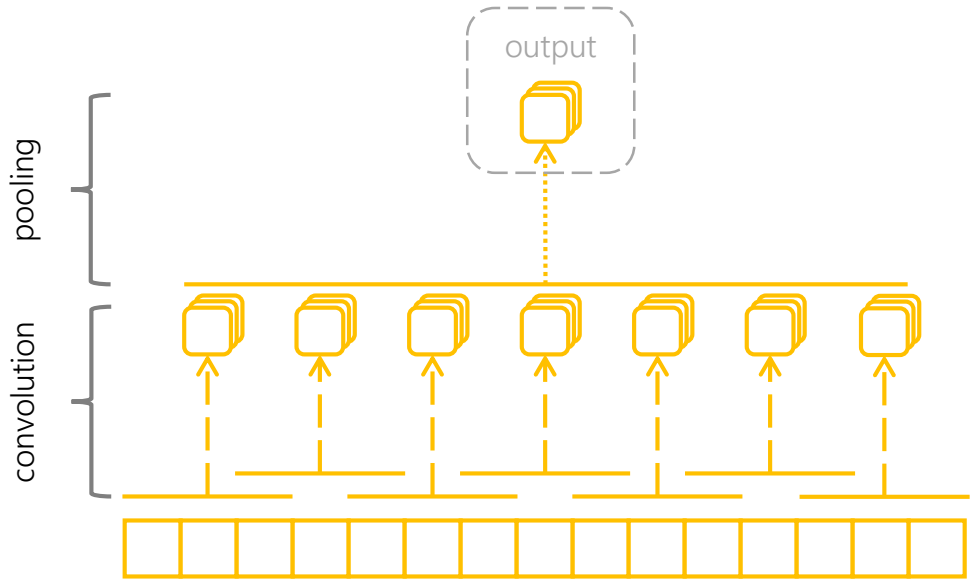
Full Input	[words x channels]
Cell Input	[window x channels]
Cell Output	[1 x channels]
Full Output	[1 + (words – window) / stride x channels]



Convolution w/ Global Pooling

Stacking a global pooling layer on top of a convolutional layer is a common strategy for generating a fixed length embedding for a variable length text

Full Input	[words x in_channels]
Full Output	[1 x out_channels]



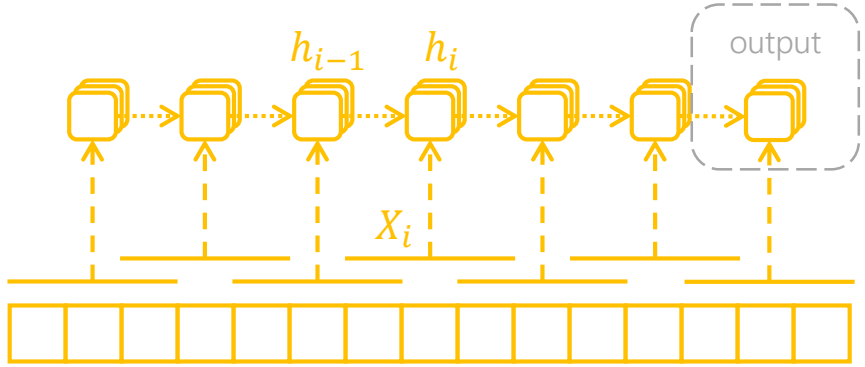
Recurrent Neural Network

Similar to a convolution layer but additional dependency on previous hidden state

A simple cell operation shown below but others like LSTM and GRUs are more popular in practice,

$$h_i = \sigma(WX_i + Uh_{i-1} + b)$$

Full Input	[words x in_channels]
Cell Input	[window x in_channels] + [1 x out_channels]
Cell Output	[1 x out_channels]
Full Output	[1 x out_channels]

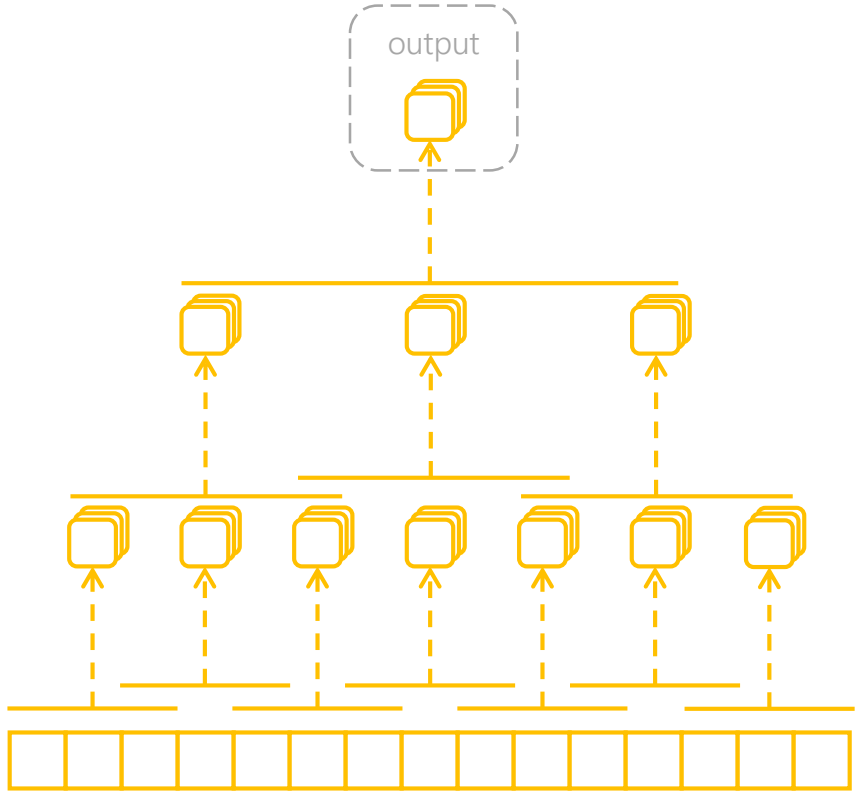


Recursive NN or Tree-RNN

Shared weights among all a levels of the tree

Cell can be an LSTM or as simple as

$$h = \sigma(WX + b)$$



Full Input	[words x channels]
Cell Input	[window x channels]
Cell Output	[1 x channels]
Full Output	[1 x channels]

Autoencoders

Unsupervised models trained to minimize reconstruction errors

$$\mathcal{L}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|^2$$

Information Bottleneck method (Tishby et al., 1999)

The bottleneck layer captures "minimal sufficient statistics" of X and is a compressed representation of the input

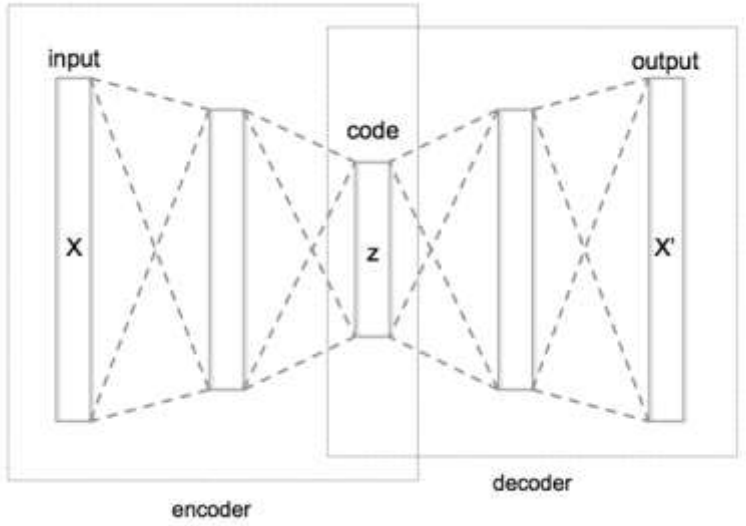


Image source: https://en.wikipedia.org/wiki/Autoencoder

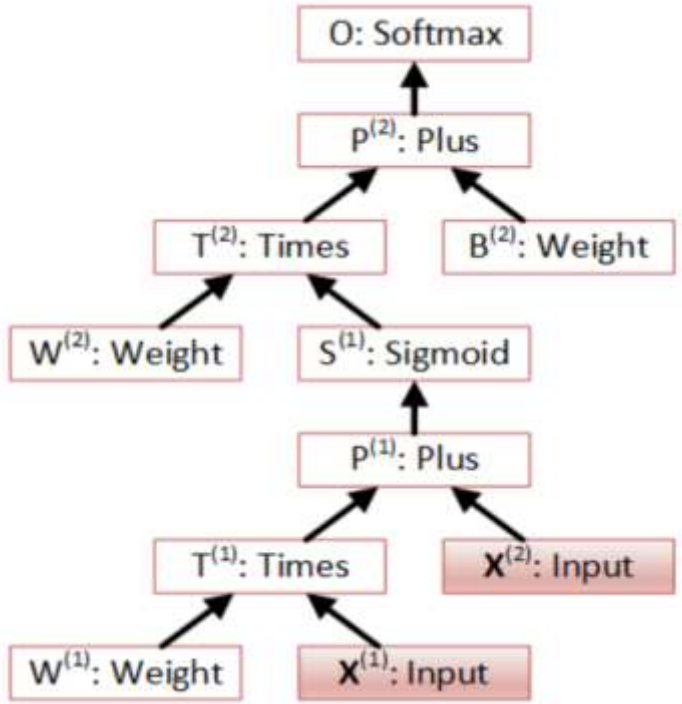
Computation Networks

The “Lego” approach to specifying DNN architectures

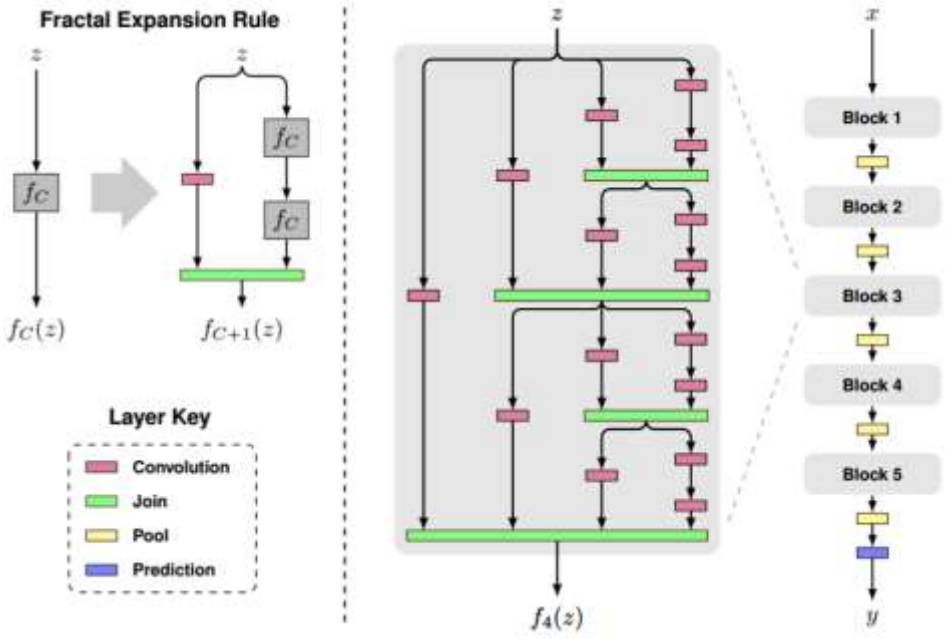
Library of computation nodes, each node defines logic for:

1. **Forward pass**: compute output given input
2. **Backward pass**: compute gradient of loss w.r.t. inputs, given gradient of loss w.r.t. outputs
3. **Parameter gradient**: compute gradient of loss w.r.t. parameters, given gradient of loss w.r.t. outputs

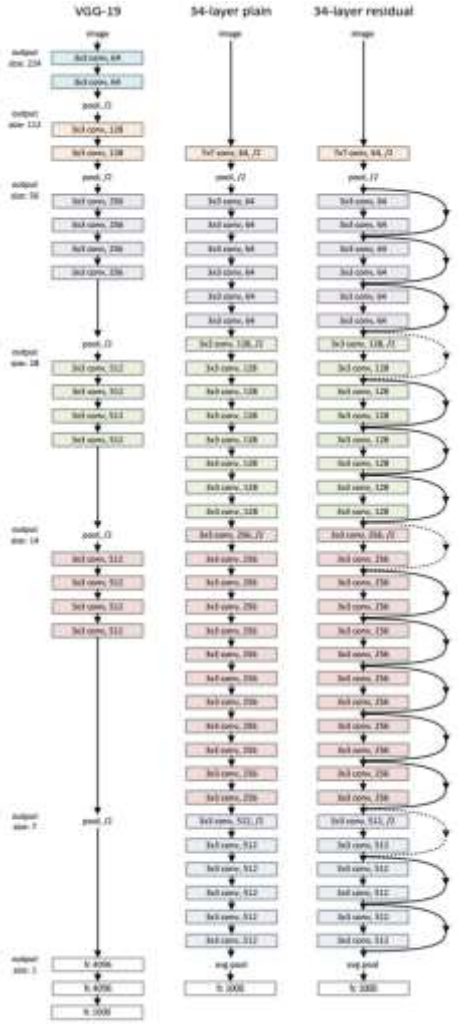
Chain nodes to create bigger and more complex networks



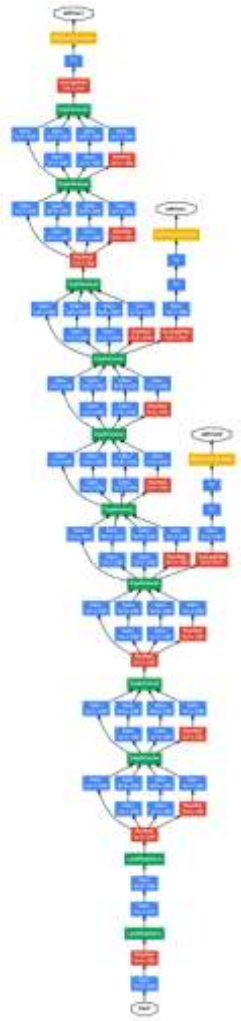
Really Deep Neural Networks



(Larsson et al., 2016)



(He et al., 2015)

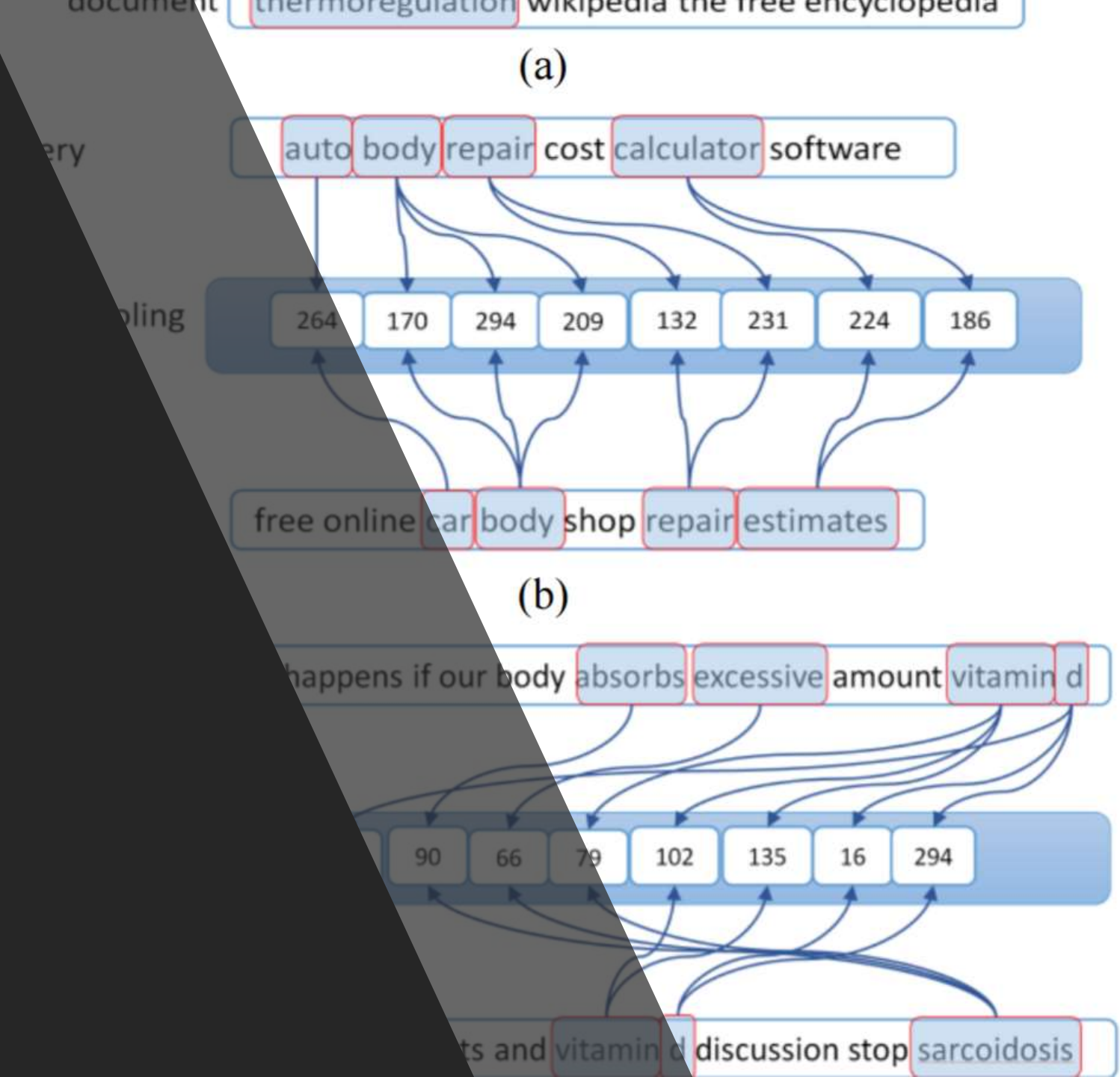


(Szegedy et al., 2014)

Microsoft

Chapter 6

DNNs for IR

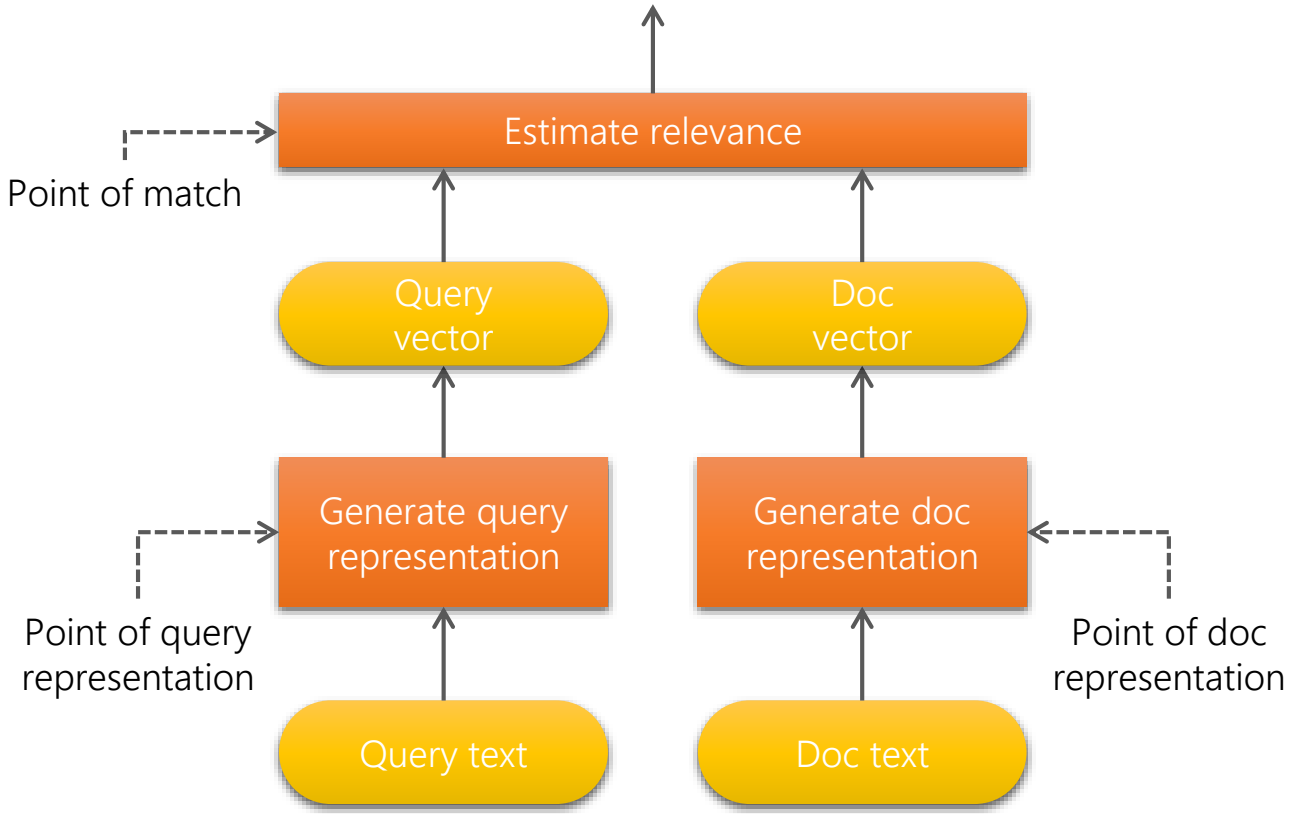


Taxonomy of Models

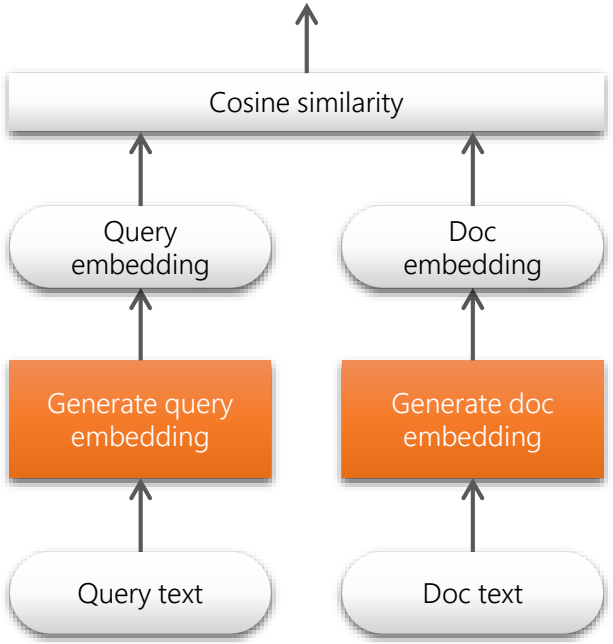
Neural networks can be used to:

1. Generate query representation,
2. Generate document representation, and/or
3. Estimate relevance.

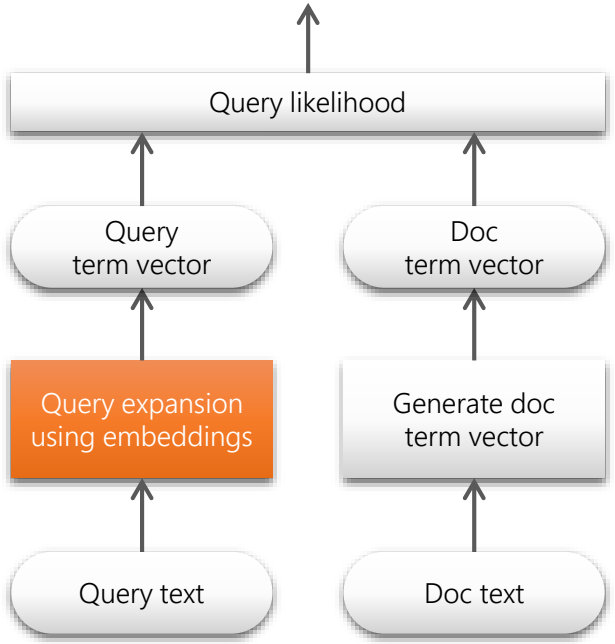
This can involve pre-trained representations or end-to-end training or some combination of approaches.



You've already seen…



e.g., DESM (Mitra et al., 2016)



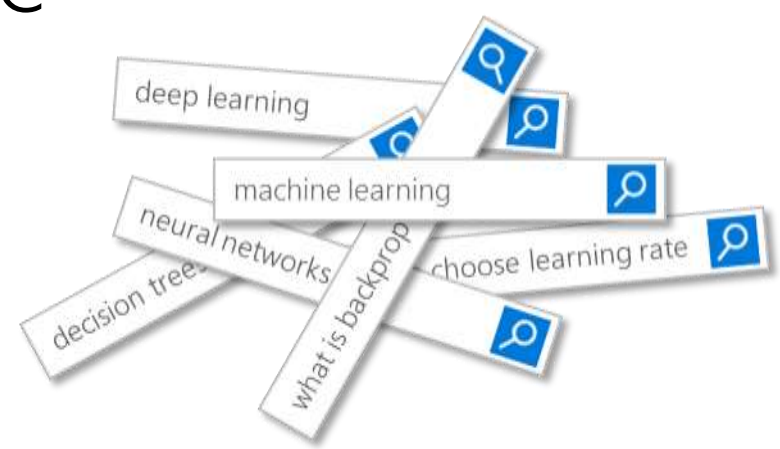
e.g., Roy et al., 2016 and Diaz et al, 2016

What does ad-hoc IR data look like?

Traditional IR uses human labels as ground truth for evaluation

So ideally we want to train our ranking models on human labels

User interaction data is rich but may contain different biases compared to human annotated labels



search queries

document corpus

user interaction / click data

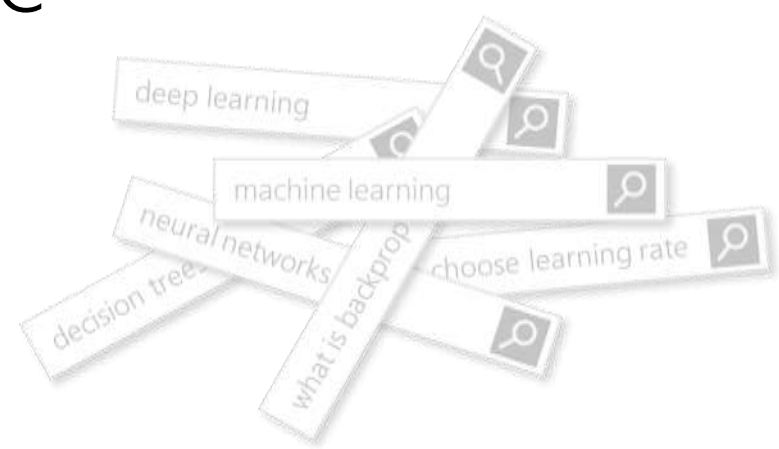
human annotated labels

What does ad-hoc IR data look like?

Traditional IR uses human labels as ground truth for evaluation

So ideally we want to train our ranking models on human labels

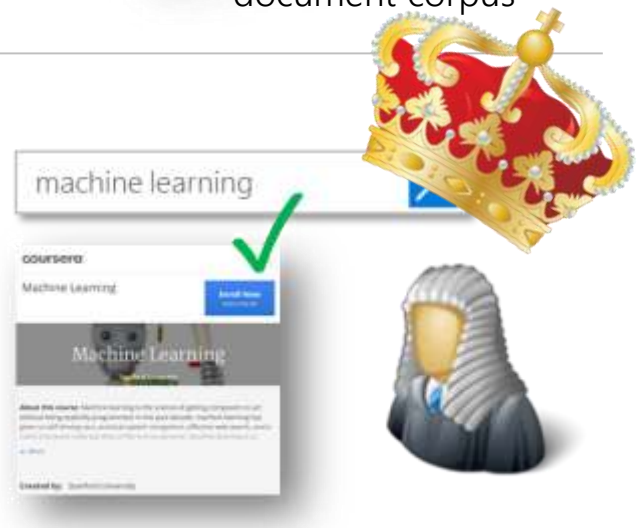
User interaction data is rich but may contain different biases compared to human annotated labels



search queries

document corpus

user interaction / click data



human annotated labels

What does ad-hoc IR data look like?

In industry:

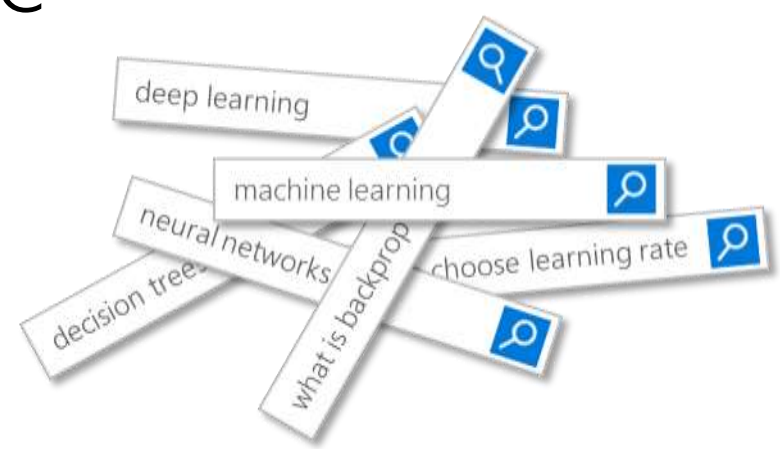
Document corpus: billions?

Query corpus: many billions?

Labelled data w/ raw text: hundreds of thousands of queries

Labelled data w/ learning-to-rank style features: same as above

User interaction data: billions?



search queries

document corpus

user interaction / click data

human annotated labels

What does ad-hoc IR data look like?

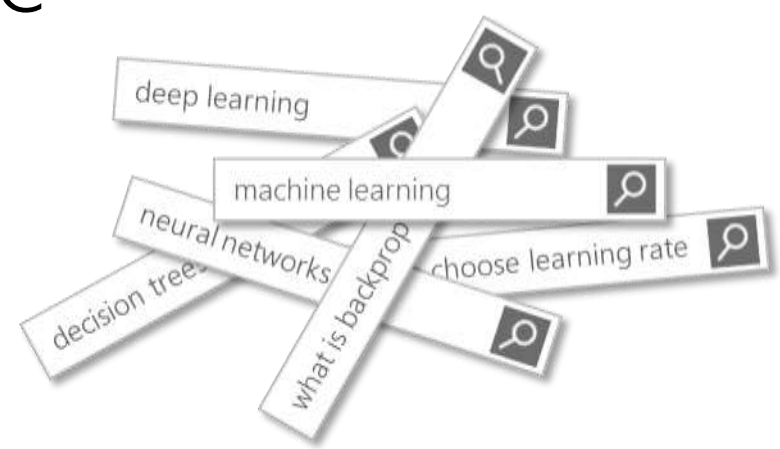
In academia:

Document corpus: few billion

Query corpus: few million but other sources (e.g., wiki titles) can be used

Labelled data w/ raw text: few hundred to few thousand queries

Labelled data w/ learning-to-rank style features: tens of thousands of queries



search queries

document corpus

human annotated labels

Microsoft

A pessimistic summary…

"~~Water, water~~, everywhere,
Nor any ~~drop to drink~~."

Data, data,
sample to train a supervised deep neural document ranking model

Levels of Supervision (or the optimistic view)

Unsupervised	Semi-supervised	Fully supervised
• Train embeddings on unlabeled corpus and use in traditional IR models • E.g., GLM, NTLM, DESM, Semantic hashing	• DNN models using pre-trained embeddings for input text representation • E.g., DRMM, MatchPyramid	• DNNs w/ raw text input (one-hot word vectors or n-graph vectors) trained on labels or click • E.g., Duet

We covered most of these

*Of course, ad-hoc retrieval isn't everything. A lot of recent DNN for IR literature focuses on other tasks such as short-text similarity or QnA, where enough training data is available.

Semantic hashing

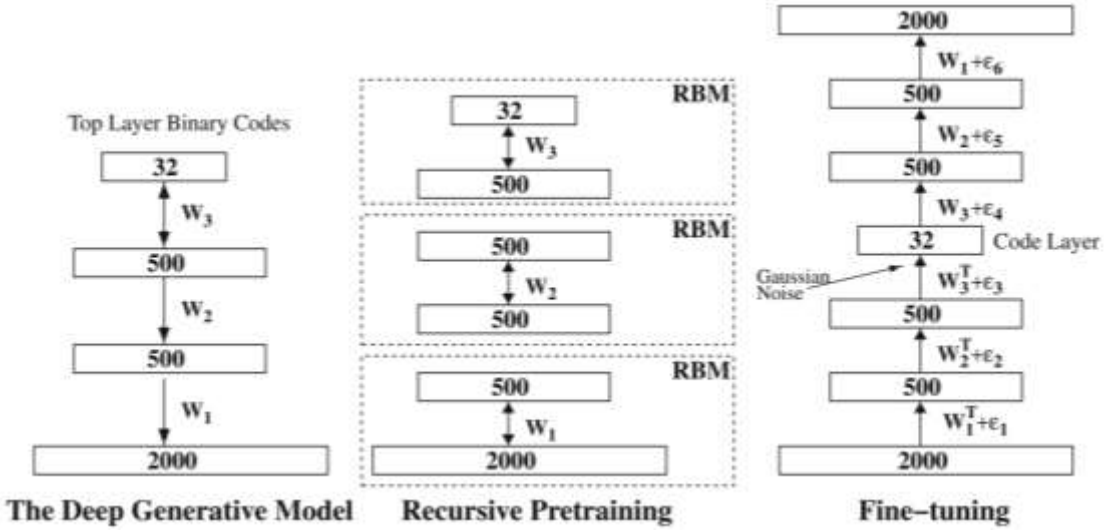
Autoencoder minimizing document reconstruction error

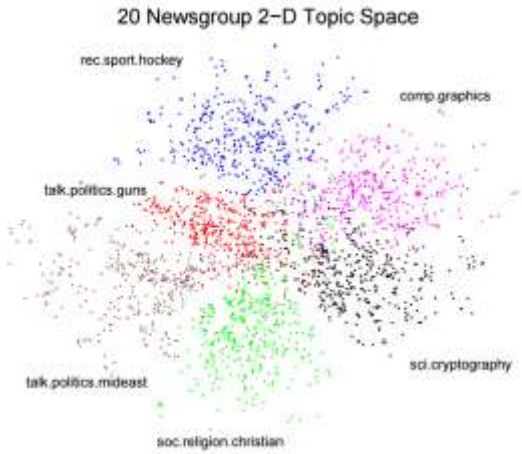
Vocab: 2K popular words (stopwords removed)

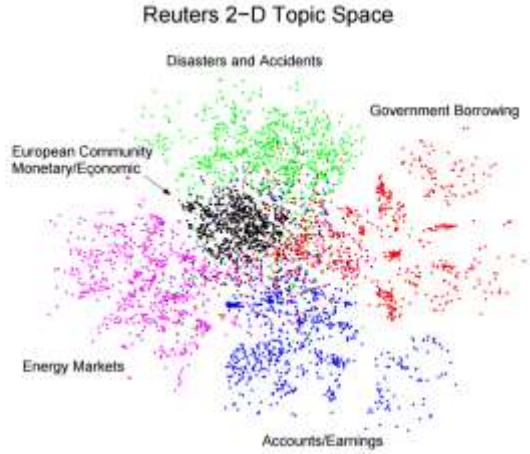
In/out = word counts and binary hidden units

Stacked RBMs w/ layer-by-layer pre-training followed by E2E tuning

Class labels based eval, no relevance judgments







(Salakhutdinov and Hinton, 2007)

Deep Semantic Similarity Model

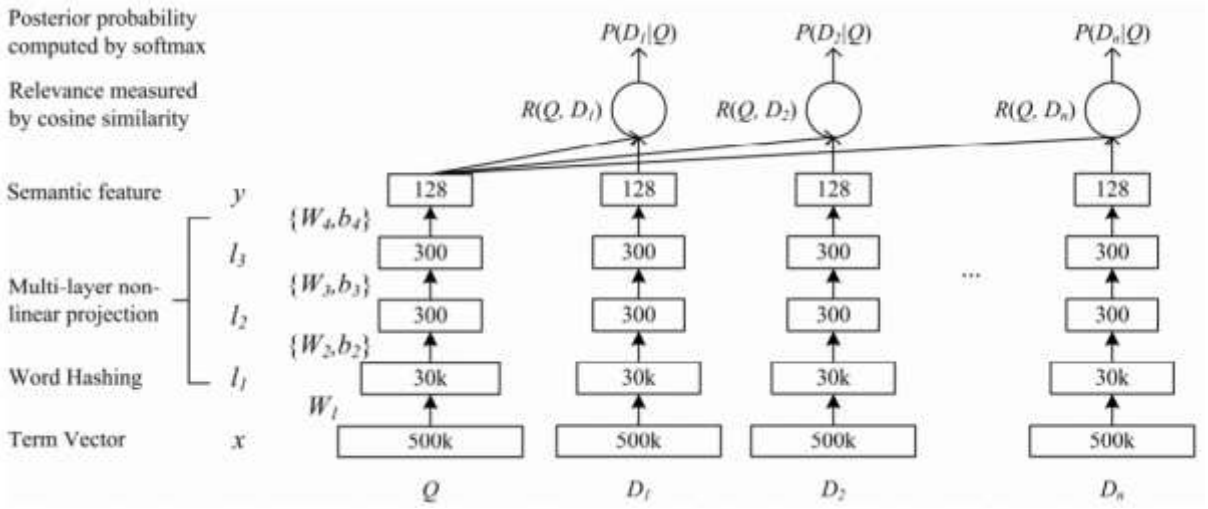
Siamese network trained E2E on query and document title pairs

Relevance is estimated by cosine similarity between query and document embeddings

Input: character trigraph counts (bag of words assumption)

Minimizes cross-entropy loss against randomly sampled negative documents

$$P(D|Q) = \frac{\exp(\gamma R(Q,D))}{\sum_{D' \in \mathbf{D}} \exp(\gamma R(Q,D'))} \qquad L(\Lambda) = -\log \prod_{(Q,D^+)} P(D^+|Q)$$



(Huang et al., 2013)

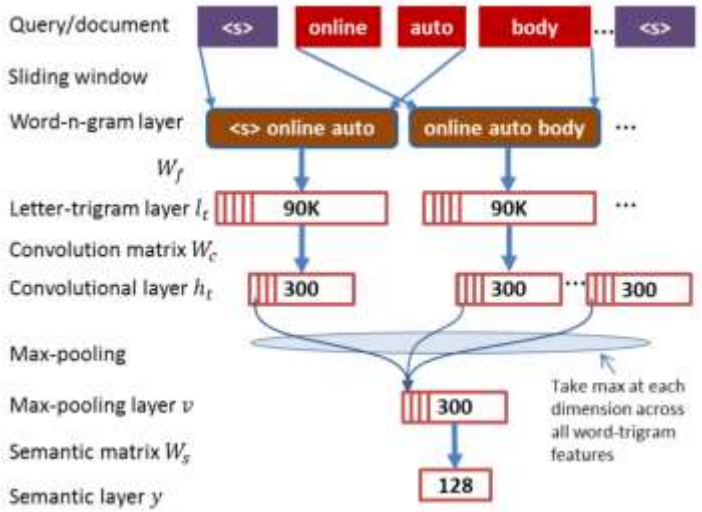
Microsoft

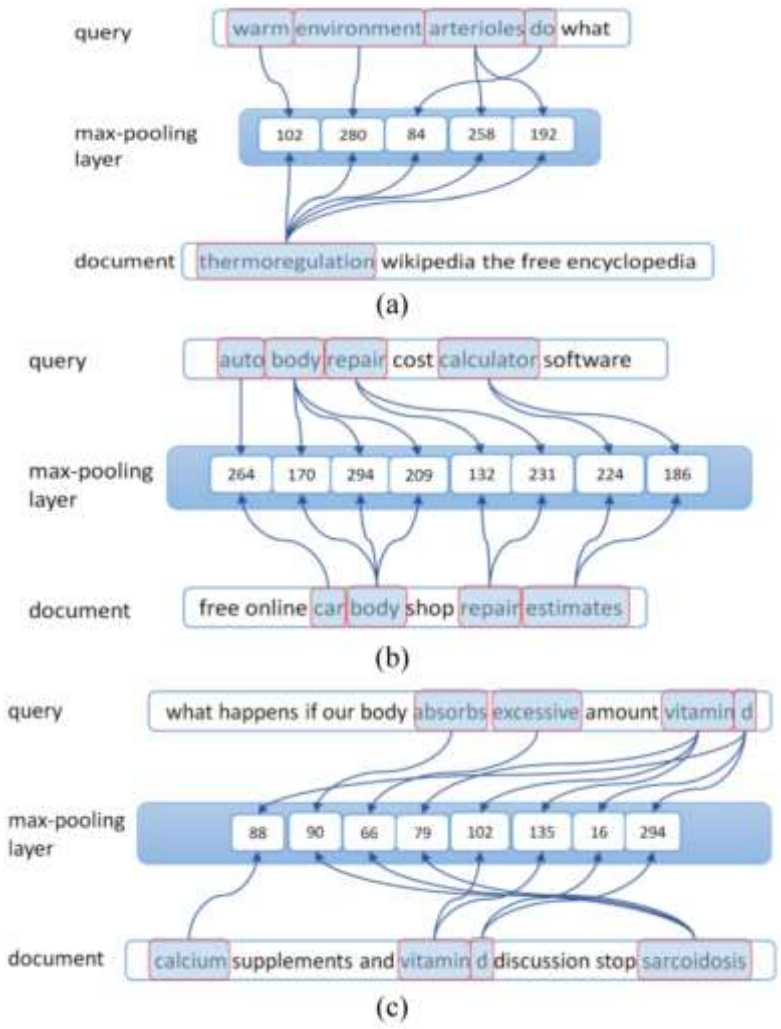
Convolutional DSSM

Replace bag-of-words assumption by concatenating term vectors in a sequence on the input

Convolution followed by global max-pooling

Performance improves further by including more negative samples during training (50 vs. 4)

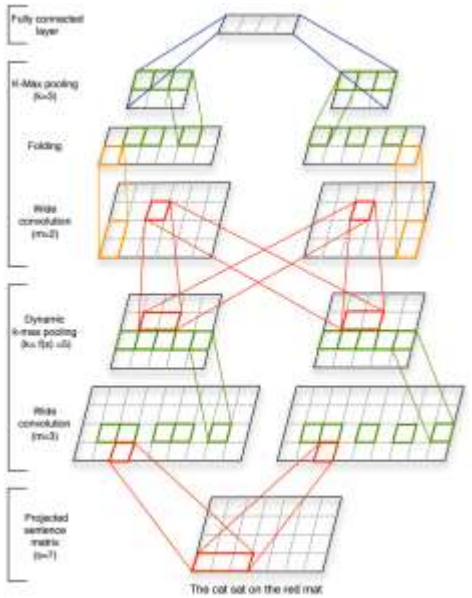




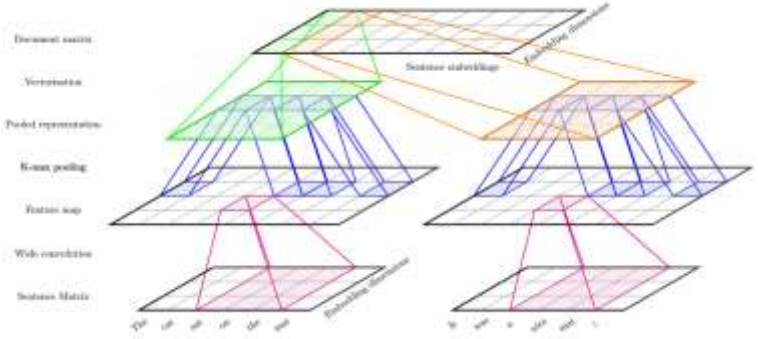
(Shen et al., 2014)

Microsoft

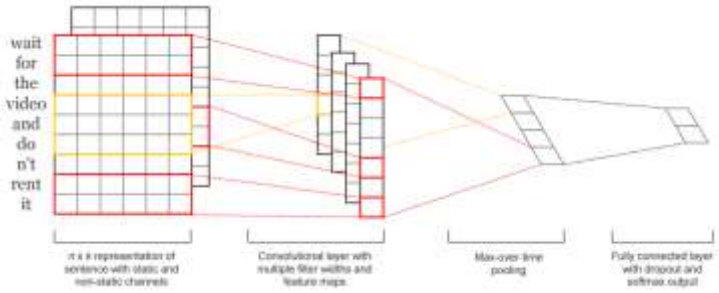
Rich space of neural architectures



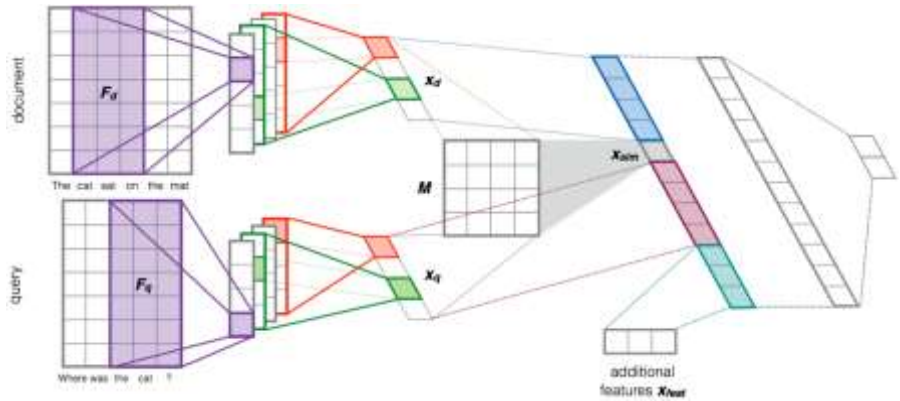
(Kalchbrenner et al., 2014)



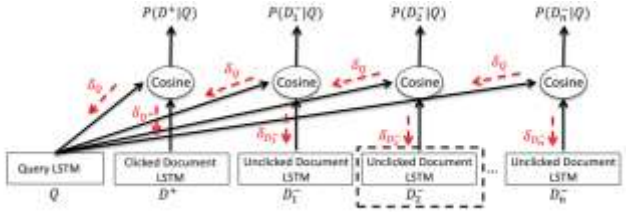
(Denil et al., 2014)

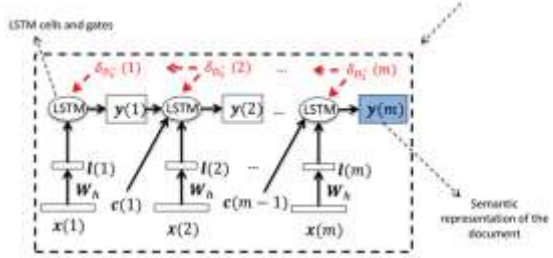


(Kim, 2014)

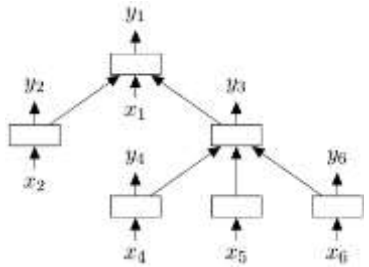


(Severyn and Moschitti, 2015)

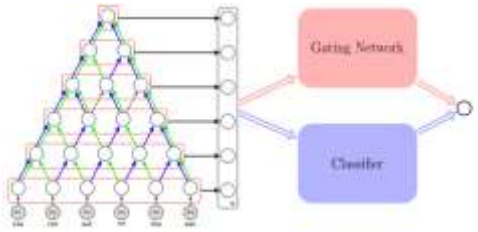




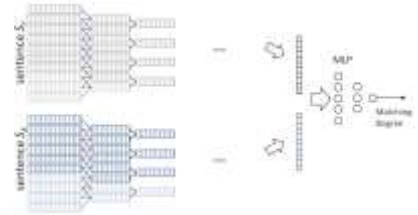
(Palangi et al., 2015)



(Tai et al., 2015)

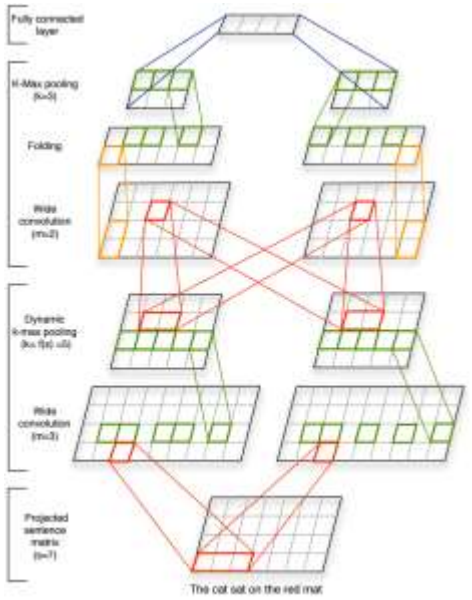


(Zhao et al., 2015)



(Hu et al., 2014)

Rich space of neural architectures

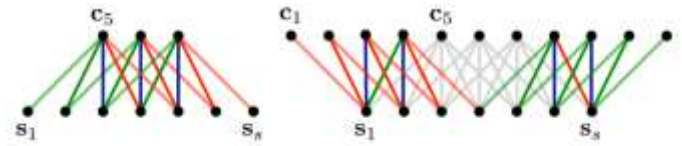


(Kalchbrenner et al., 2014)

Dynamic Convolutional Neural Network

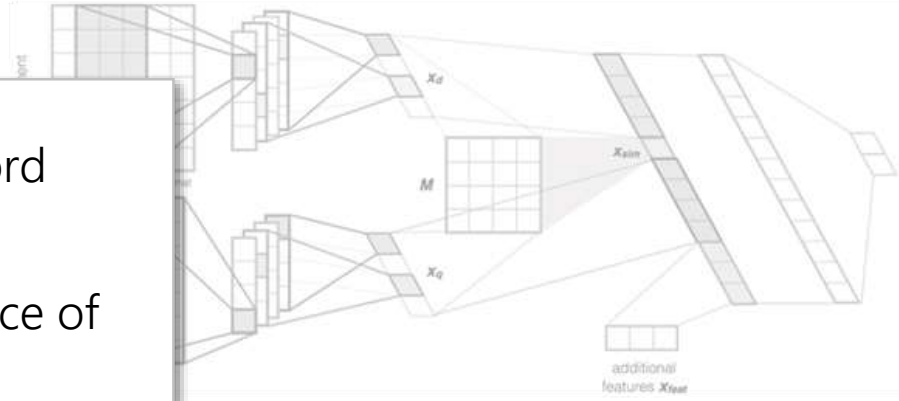
Input representation based on pre-trained word embeddings

Wide vs. narrow convolution (presence/absence of zero padding)

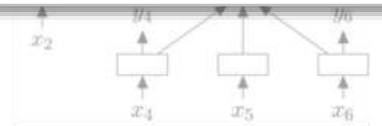


Dynamic k max-pooling, where k is a function of the number of convolutional layers and sentence length

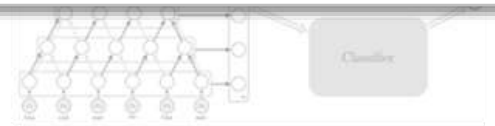
Evaluated on non-IR tasks (sentiment prediction and question-type classification)



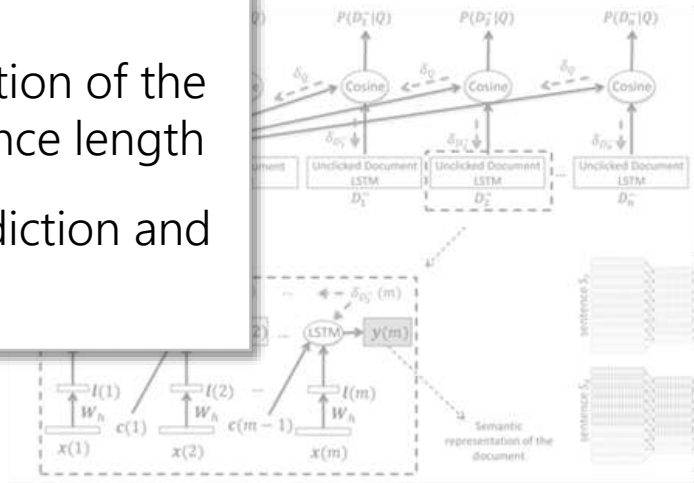
(Severyn and Moschitti, 2015)



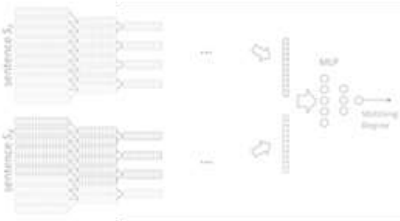
(Tai et al., 2015)



(Zhao et al., 2015)



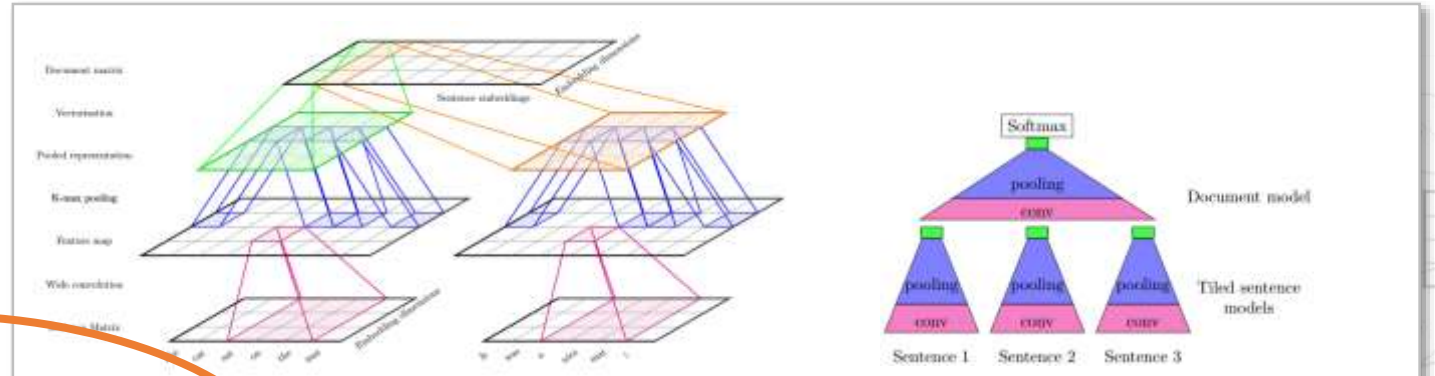
(Palangi et al., 2015)



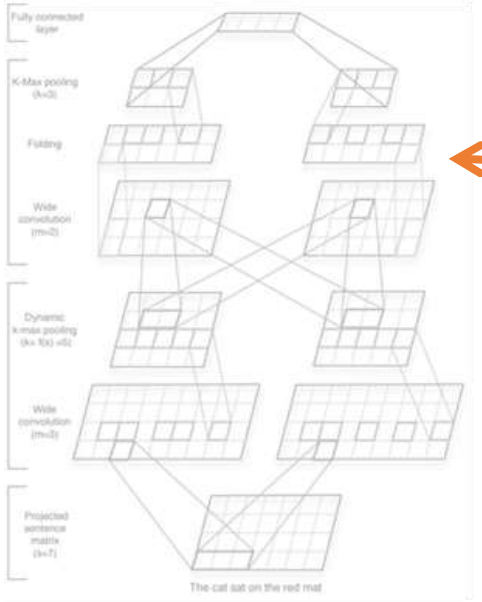
(Hu et al., 2014)

Microsoft

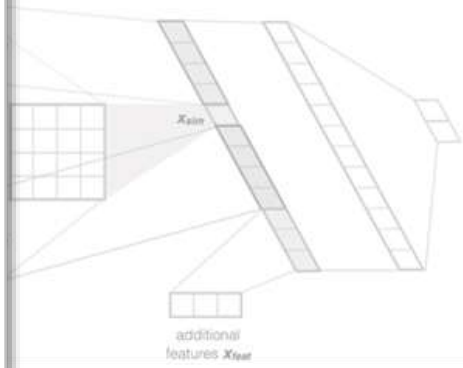
Rich space of neural architectures



(Denil et al., 2014)



(Kalchbrenner et al., 2014)



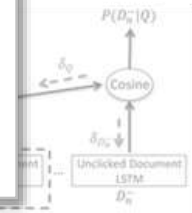
yn and Moschitti, 2015)

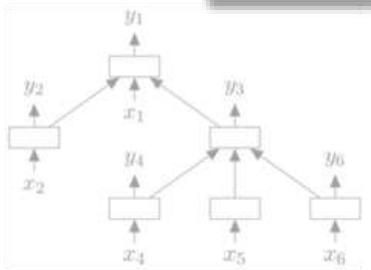
Uses the DCNN model to learn document embeddings

First DCNN on word embeddings to get sentence embedding

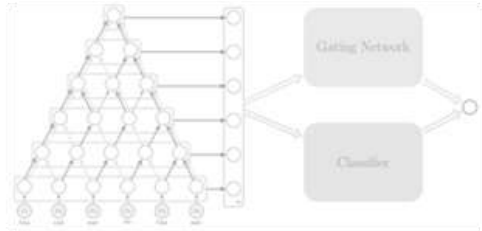
Then separate DCNN on sentences to get document embedding

Evaluated on classification tasks

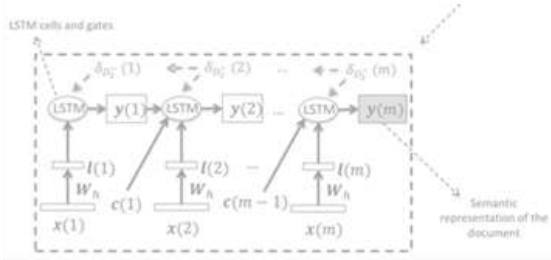




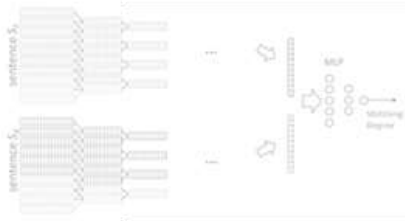
(Tai et al., 2015)



(Zhao et al., 2015)



(Palangi et al., 2015)



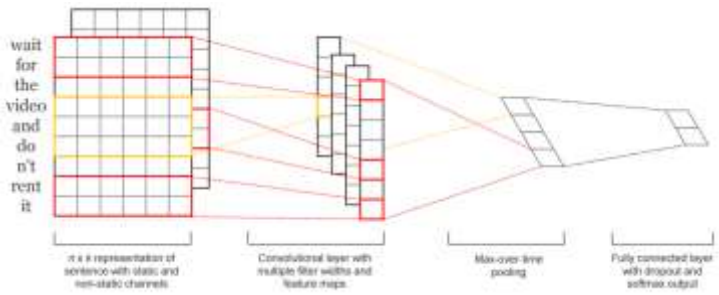
(Hu et al., 2014)

Microsoft

Rich space of neural architectures

Convolutional model for text classification and sentiment prediction

Compares strategies for input word representation using word embeddings



(Kim, 2014)

Diff. strategy wins on diff. dataset, size of E2E training data must be an important factor

1. Embeddings are randomly initialized and learnt during E2E training
2. Pre-trained embeddings not updated during E2E training
3. Pre-trained embeddings further tuned during E2E training
4. Concatenation of two different vectors, both initialized with pre-trained embeddings but only one update during E2E training

(Kalchbrenner et al., 2014)

(Tai et al., 2015)

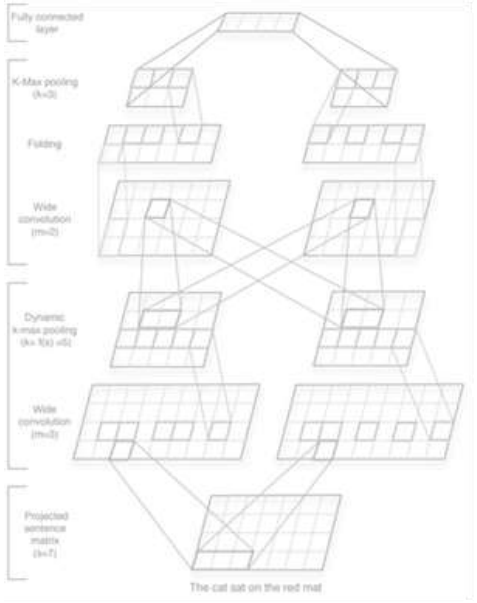
(Zhao et al., 2015)

(Palangi et al., 2015)

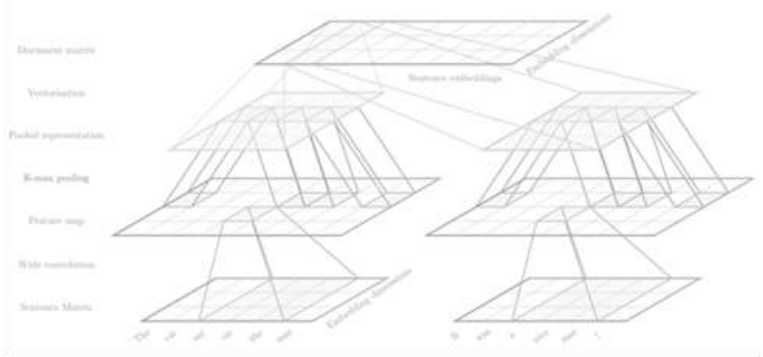
(Hu et al., 2014)

Microsoft

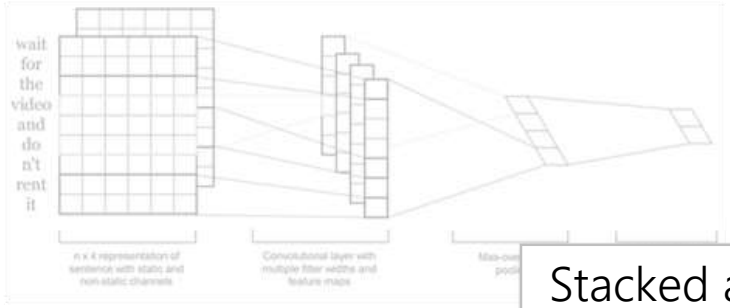
Rich space of neural architectures



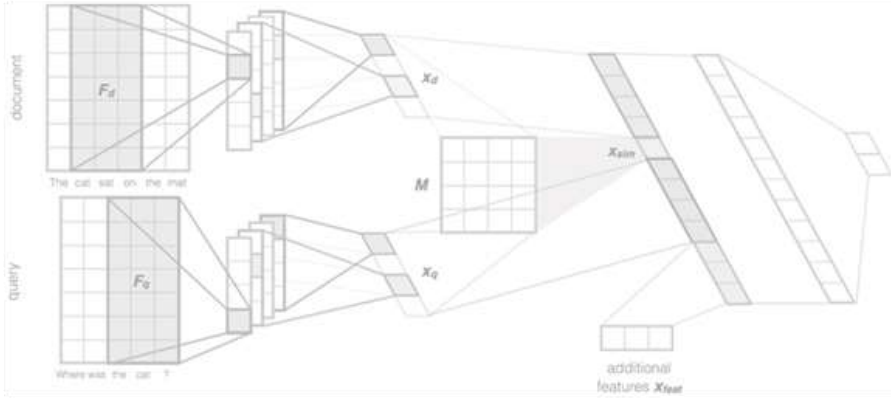
(Kalchbrenner et al., 2014)



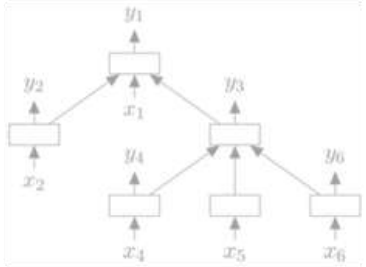
(Denil et al., 2014)



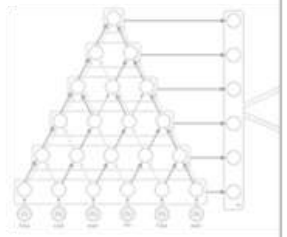
(Kim, 2014)



(Severyn and Moschitti, 2015)



(Tai et al., 2015)



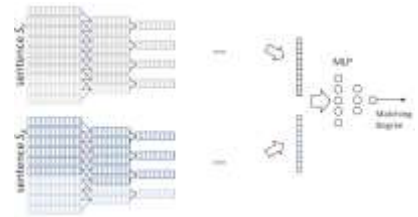
(Zhao et al.,

Stacked alternate layers of convolution and max-pooling

Depth of network is fixed

Unlike recursive models, the weights are not shared between convolutions at different depth

ARC-I



(Hu et al., 2014)

Microsoft

Rich space of neural architectures

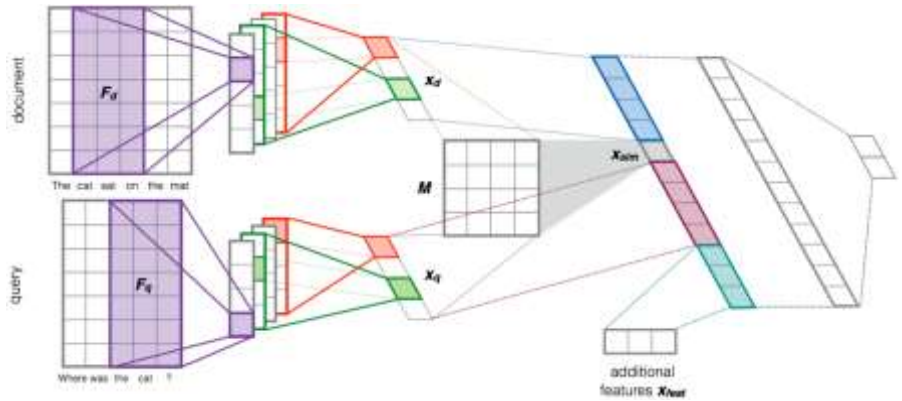
Convolutional model for short text ranking

Pre-trained word embeddings for input, not tuned during E2E training

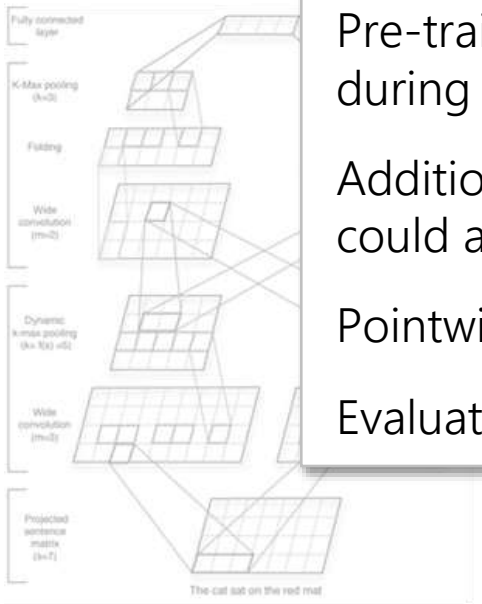
Additional layers for Q-D embedding comparison, could also include non embedding features

Pointwise loss function, model predicts relevance class

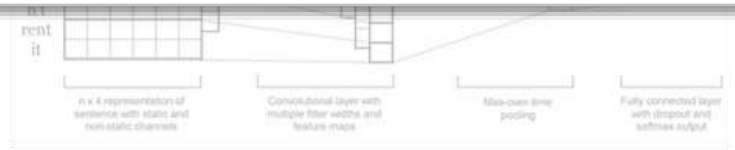
Evaluated on microblog retrieval, unfortunately no comparisons to DSSM / CDSSM



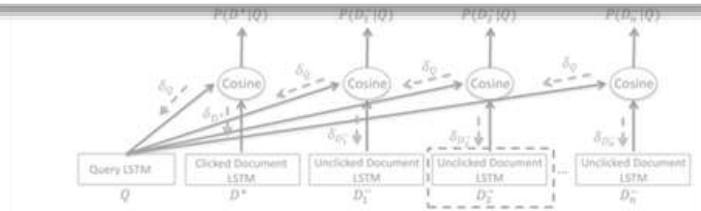
(Severyn and Moschitti, 2015)

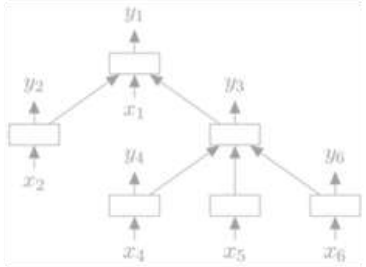


(Kalchbrenner et al., 2014)

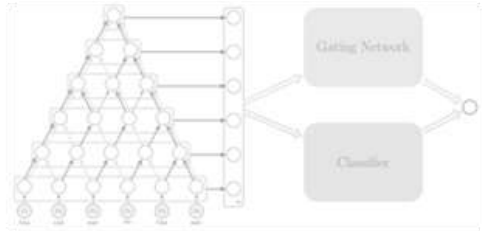


(Kim, 2014)

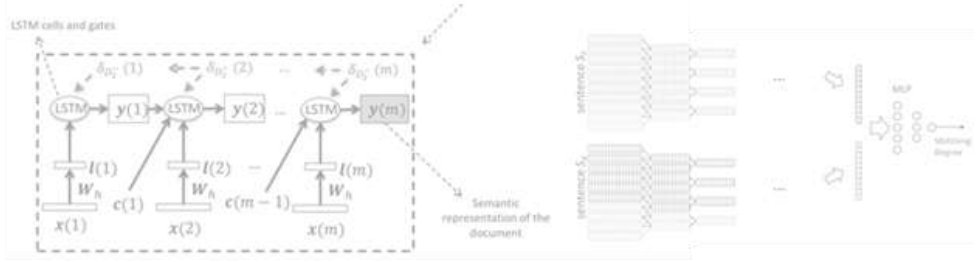




(Tai et al., 2015)



(Zhao et al., 2015)

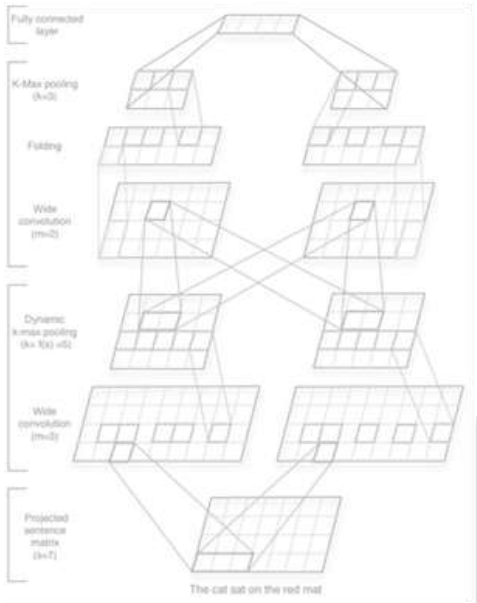


(Palangi et al., 2015)

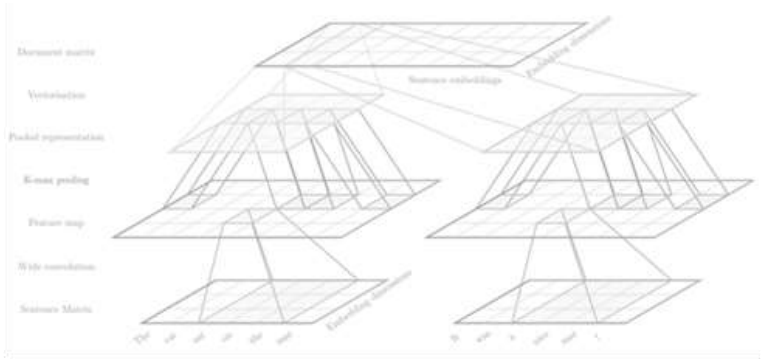
(Hu et al., 2014)

Microsoft

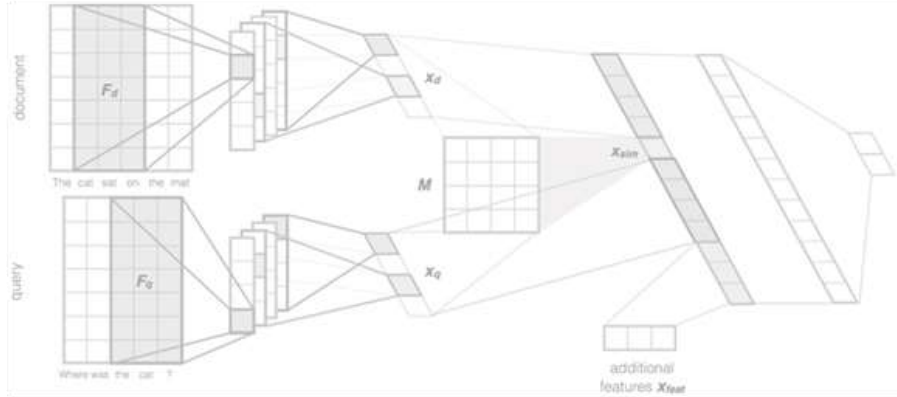
Rich space of neural architectures



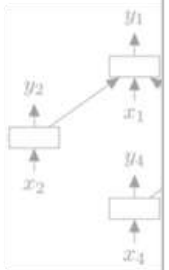
(Kalchbrenner et al., 2014)



(Denil et al., 2014)



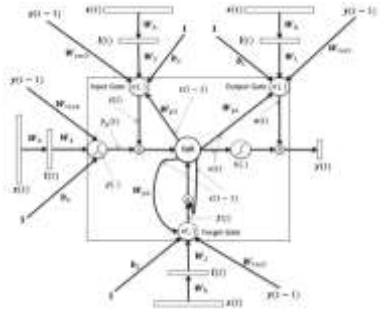
(Severyn and Moschitti, 2015)

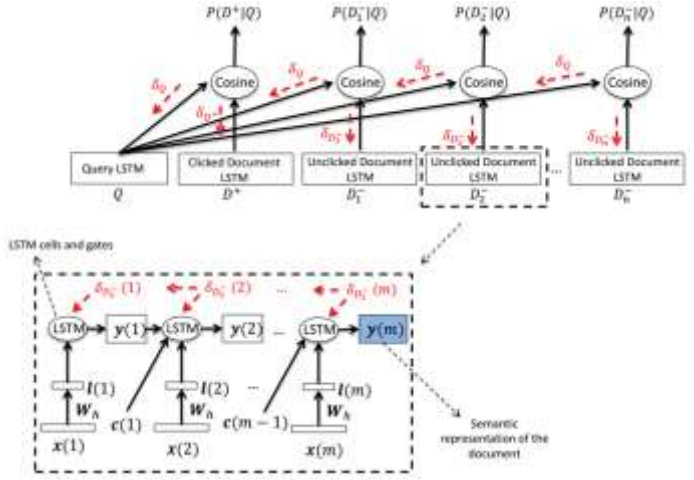


(Tai e

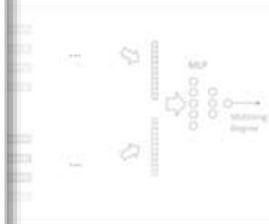
LSTM-DSSM

Replace convolution-pooling w/ LSTM





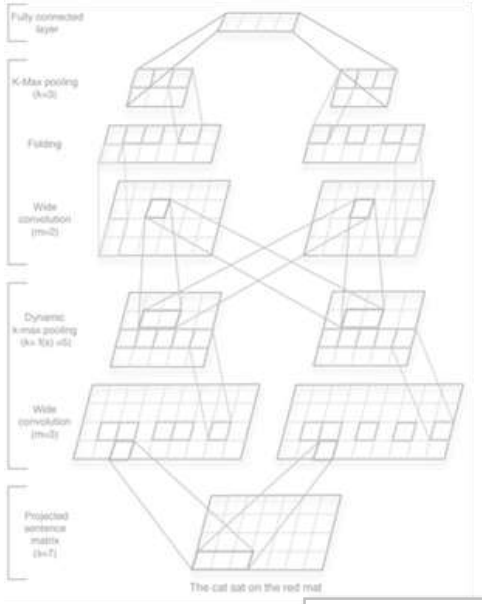
(Palangi et al., 2015)



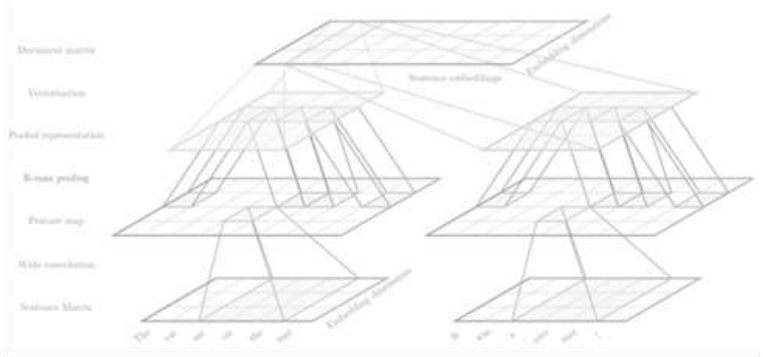
u et al., 2014)

Microsoft

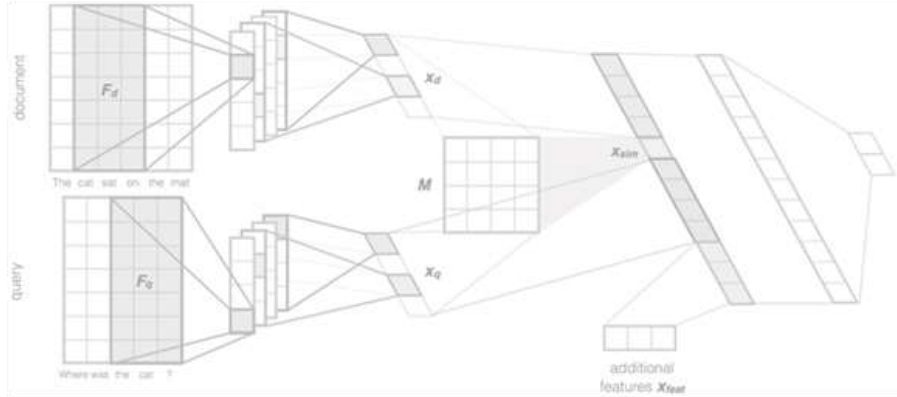
Rich space of neural architectures



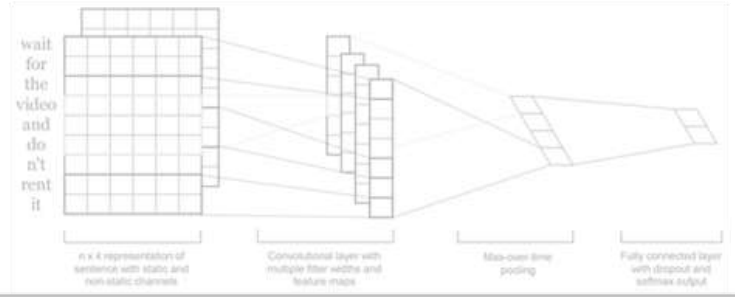
(Kalchbrenner et al.

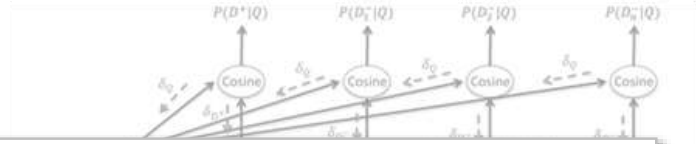


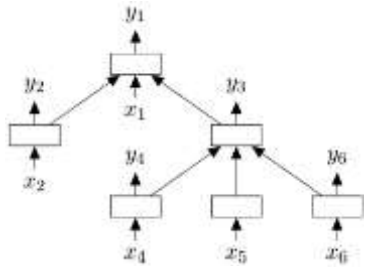
(Denil et al., 2014)



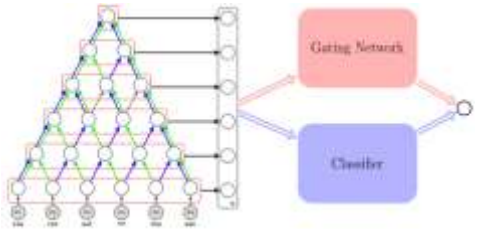
(Severyn and Moschitti, 2015)







(Tai et al., 2015)



(Zhao et al., 2015)

Recently various tree-structured RNNs / LSTMs have also been explored in the context of text similarity and classification tasks

(Hu et al., 2014)

Microsoft

Interaction matrix

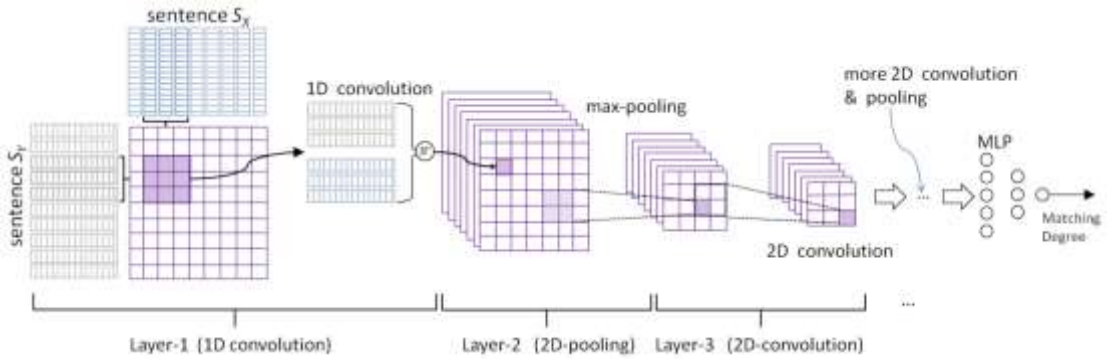
Alternative to Siamese networks

Interaction matrix X, where $x_{i,j}$ is obtained by comparing the the i^{th} word in source sentence with j^{th} word in target sentence

Comparison is generally lexical or based on pre-trained embeddings

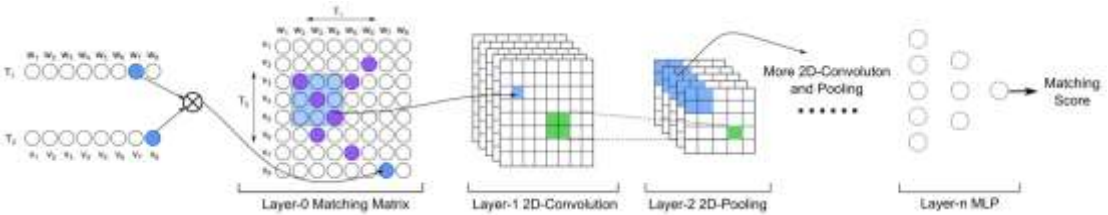
Focus on recognizing good matching patterns instead of learning good representations

ARC-II



(Hu et al., 2014)

MatchPyramid



(Pang et al., 2014)

Ad-hoc retrieval using local representation

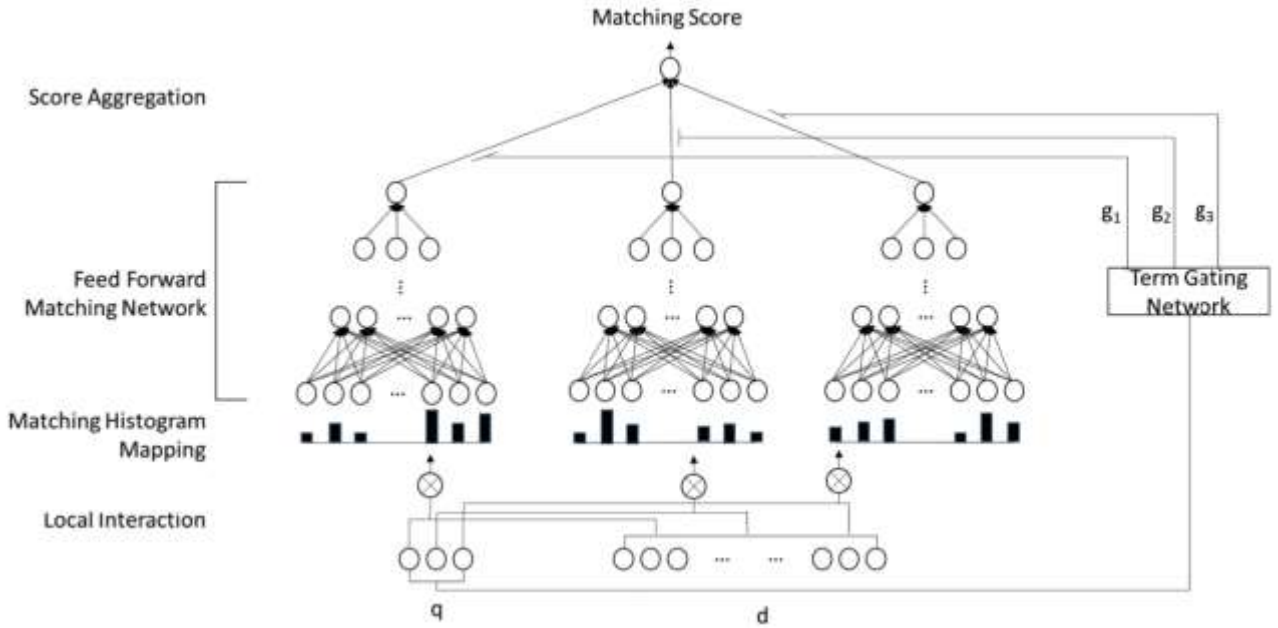
Deep Relevance Matching Model (DRMM) for ad-hoc retrieval

Argues exact matching more important than representation learning for ad-hoc retrieval

DNN on top of histogram based features

Gating network based on term IDF score

Word embeddings are also used but their impact not clear



(Guo et al., 2016)

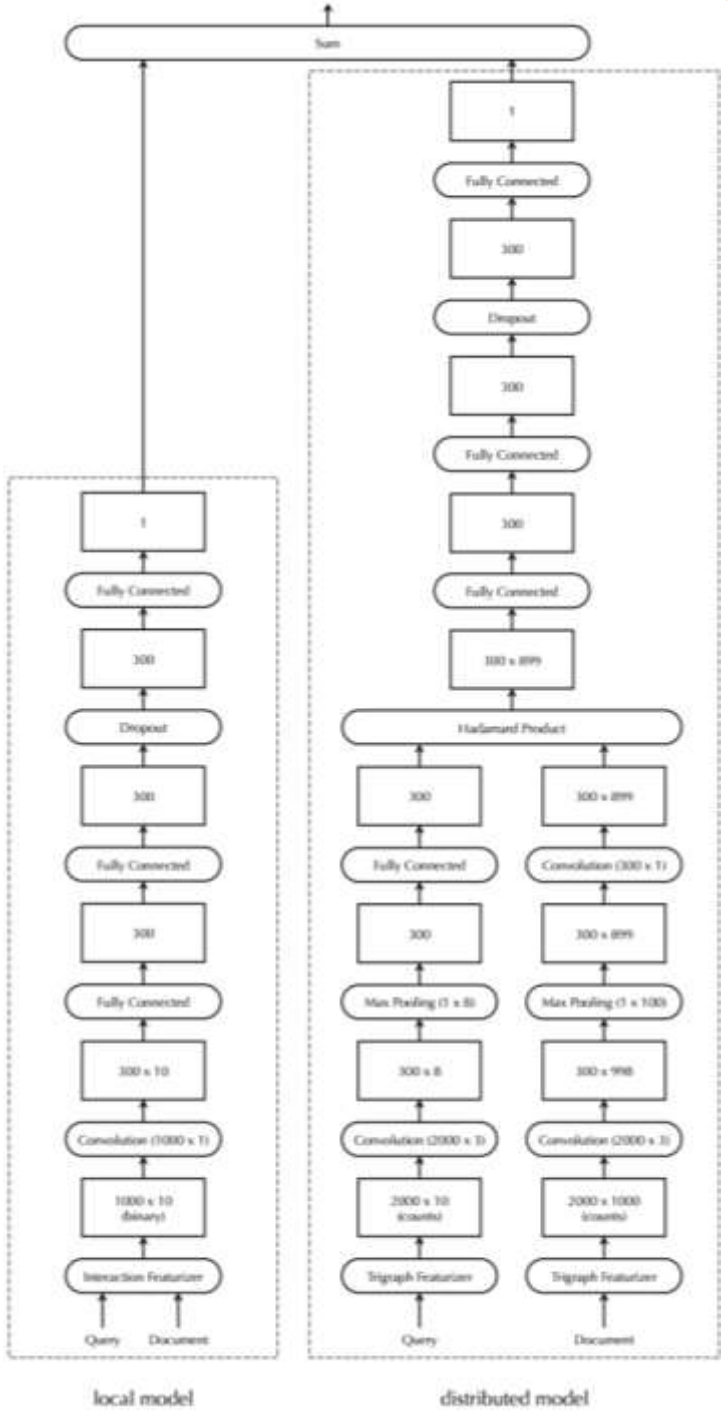
Ad-hoc retrieval using local and distributed representation

Argues both “lexical” and “semantic” matching is important for document ranking

Duet model is a linear combination of two DNNs using local and distributed representations of query/document as inputs, and jointly trained on labelled data

Local model operates on lexical interaction matrix

Distributed model operates on *n*-graph representation of query and document text



(Mitra et al., 2017)

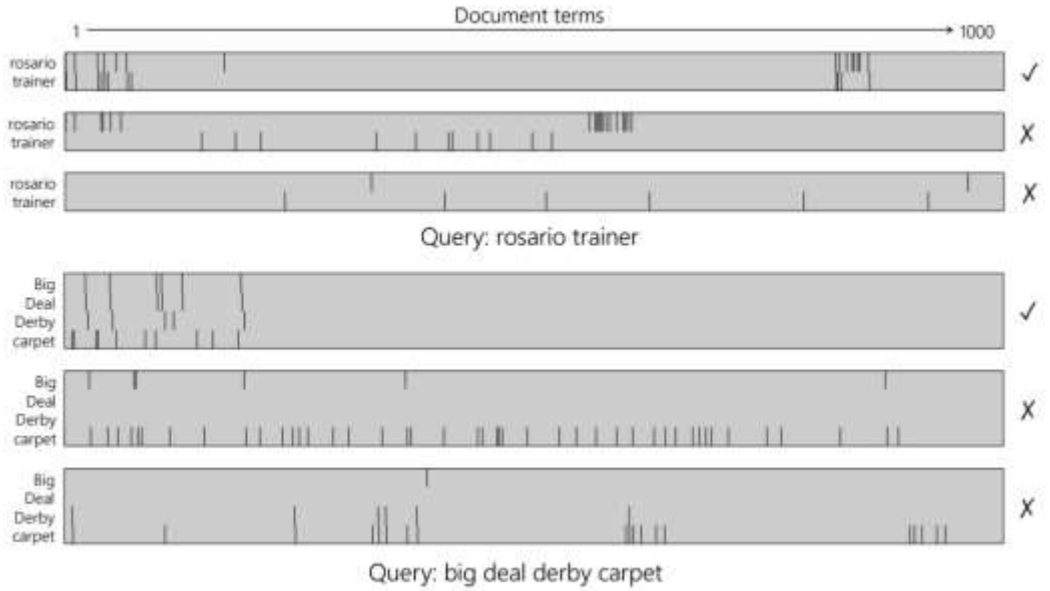
Ad-hoc retrieval using local and distributed representation

Argues both “lexical” and “semantic” matching is important for document ranking

Duet model is a linear combination of two DNNs using local and distributed representations of query/document as inputs, and jointly trained on labelled data

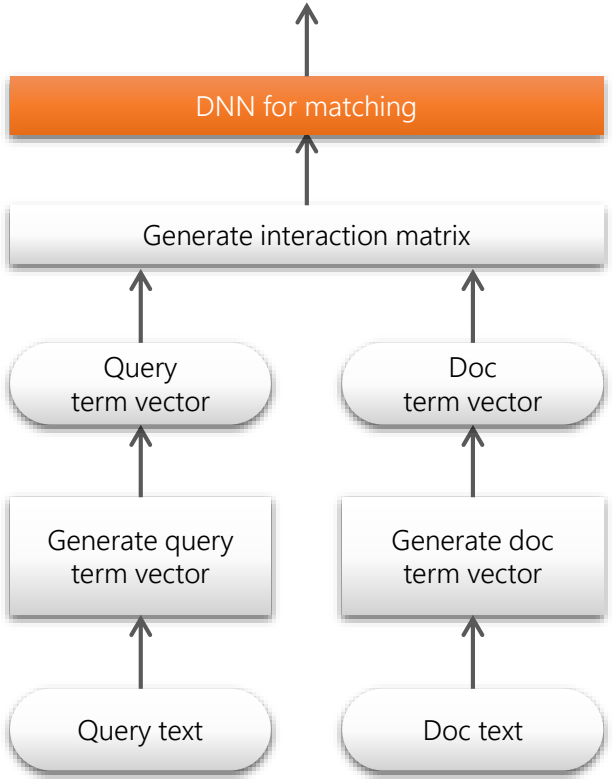
Local model operates on lexical interaction matrix

Distributed model operates on *n*-graph representation of query and document text

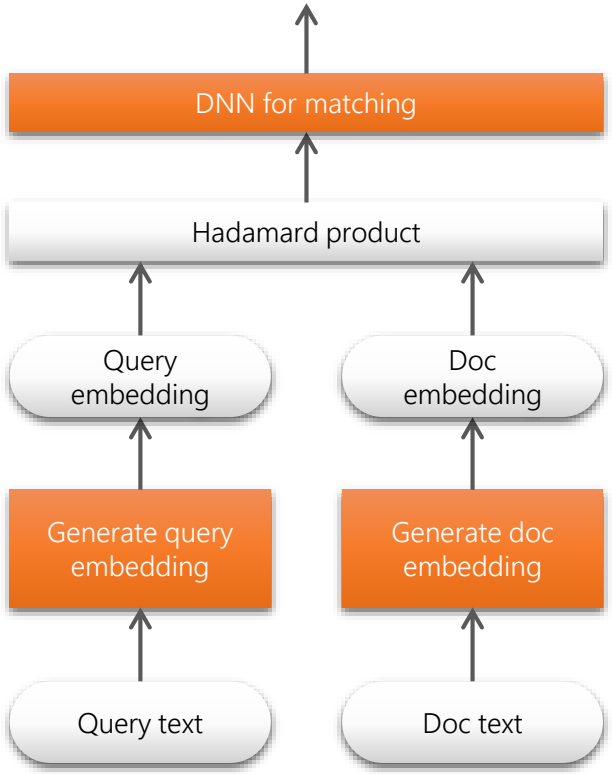


(Mitra et al., 2017)

Using our earlier taxonomy of models…



Local model



Distributed model

Ad-hoc retrieval using local and distributed representation

Distributed model not likely to have good representation for rare intents (e.g., a television model number 'SC32MN17')

For the query "what channel are the seahawks on today" documents may contain the term "ESPN" instead – distributed model likely to make the "channel" → "ESPN" connection

Distributed model more effective for popular intents, fall back on local model for rare ones

Local and distributed models likely to make different mistakes and therefore more effective when combined

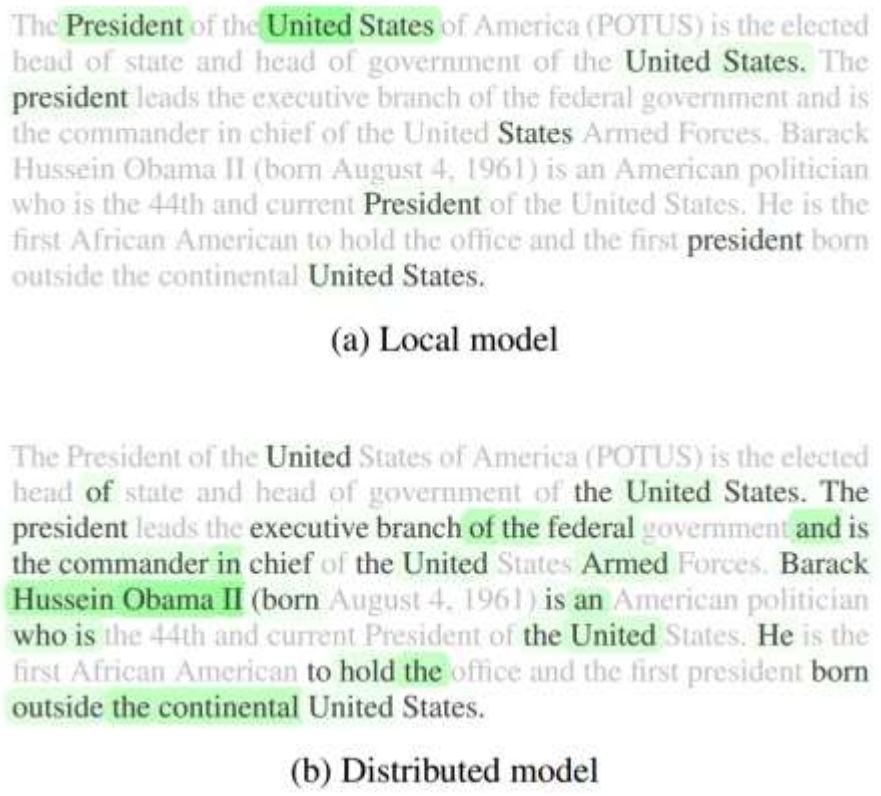


Figure 1: Visualizing the drop in each model's retrieval score by individually removing each of the passage terms for the query "united states president". Darker green signifies bigger drop. The local model uses only exact term matches. The distributed model uses matches based on a learned representation.

(Mitra et al., 2017)

The library vs. librarian dilemma

Is there a hidden assumption in distributed models that they know about everything in the universe?

The distributed model knows more about “Barack Obama” than “Bhaskar Mitra” and will perform better on the former

The local model doesn’t understand either but therefore might perform better for the latter query compared to the distributed model

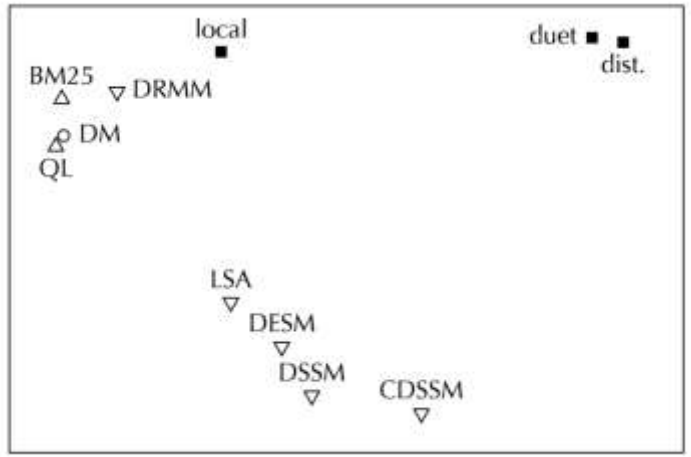
Is your model the library (**knows about the world**) or the librarian (**knows how to find** information in a without much prior domain knowledge)?

Big vs. small data regimes

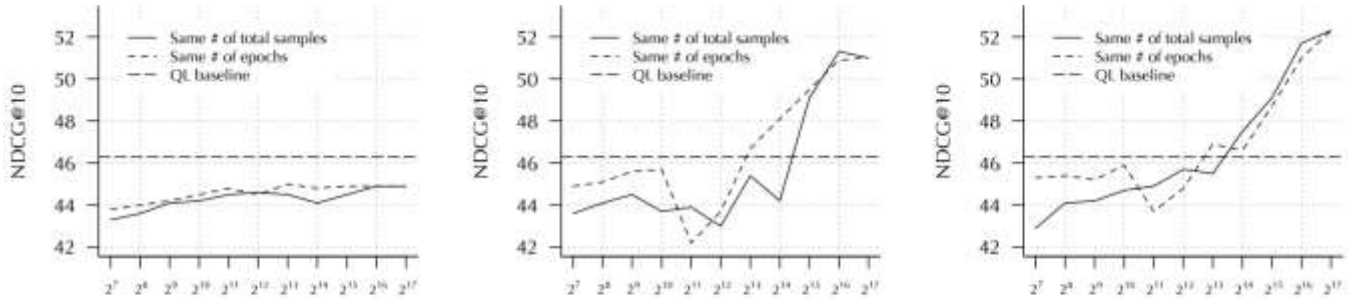
Big data seems to be more crucial for models that focus on good representation learning for text

Partial supervision strategies (e.g., unsupervised pre-training of word embeddings) can be effective but may be leaving the bigger gains on the table

Learning to train on unlabeled data may be key to making progress on neural ad-hoc retrieval



Which IR models are similar? Clustering based on query level retrieval performance.



(a) Local model (b) Distributed model (c) Duet model

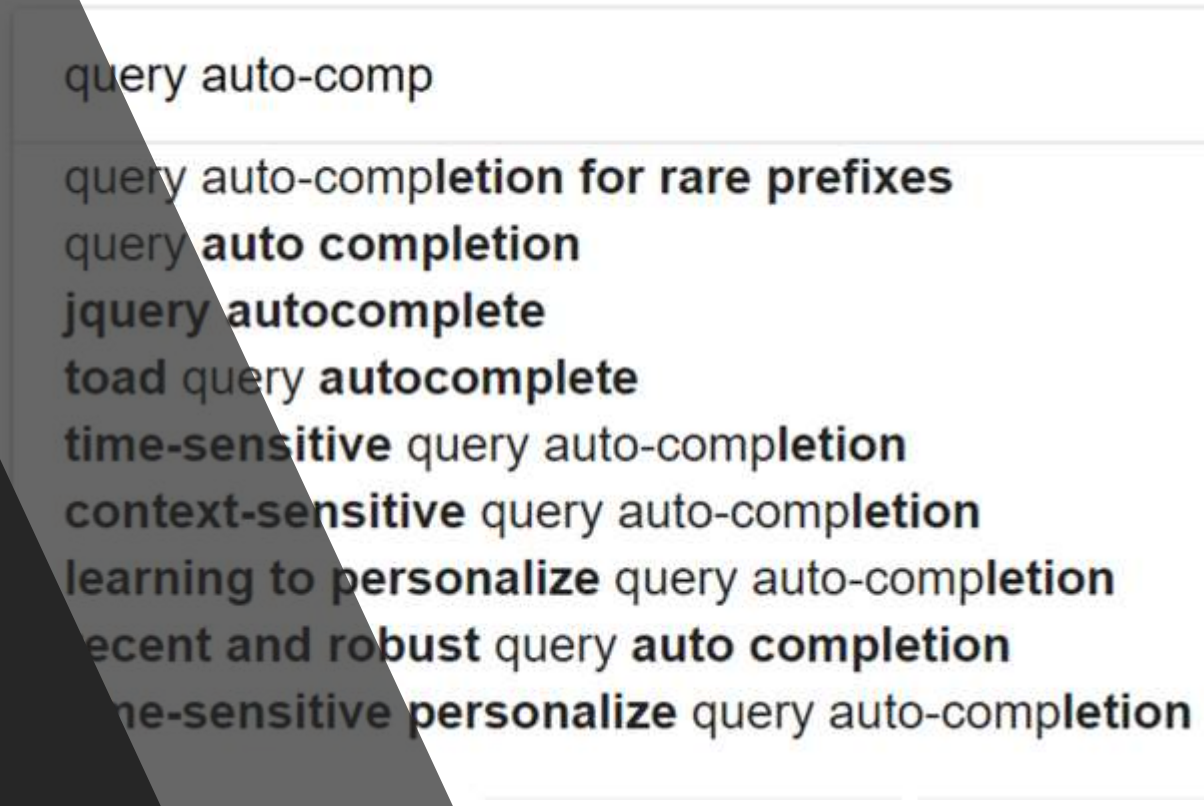
Microsoft

Implementing Duet using the Cognitive Toolkit (CNTK)

https://github.com/bmitra-msft/NDRM/blob/master/notebooks/Duet.ipynb

Chapter 7

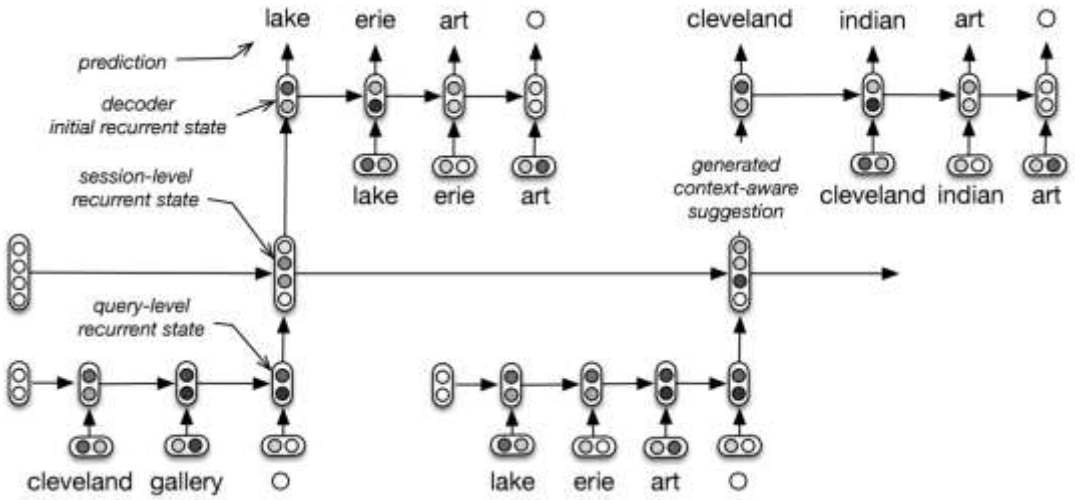
Other Applications



Query Recommendations

Hierarchical sequence-to-sequence model for term-by-term query generation

Similar to ad-hoc ranking the DNN feature alone performs poorly but shows significant improvements over a model with lexical contextual features



(Sordoni et al., 2015)

Query auto-completion

Given a (rare) query prefix retrieve relevant suffixes from a fixed set

CDSSM model trained on query prefix-suffix pairs can be used for suffix ranking

Training on prefix-suffix produces a leads to a more "Typical" embedding space

what to cook with chicken and broccoli and
what to cook with chicken and broccoli *and bacon*
what to cook with chicken and broccoli *and noodles*
what to cook with chicken and broccoli *and brown sugar*
what to cook with chicken and broccoli *and garlic*
what to cook with chicken and broccoli *and orange juice*
what to cook with chicken and broccoli *and beans*
what to cook with chicken and broccoli *and onions*
what to cook with chicken and broccoli *and ham soup*

cheapest flights from seattle to
cheapest flights from seattle *to dc*
cheapest flights from seattle *to washington dc*
cheapest flights from seattle *to bermuda*
cheapest flights from seattle *to bahamas*
cheapest flights from seattle *to aruba*
cheapest flights from seattle *to punta cana*
cheapest flights from seattle *to airport*
cheapest flights from seattle *to miami*

(Mitra and Craswell, 2015)

Analogies for session modelling

CDSSM vectors when trained on prefix-suffix pairs or session query pairs are amenable to analogical tasks

Query vector	Nearest neighbour
$vector(\text{“chicago”}) + vector(\text{“newspaper”})$	$vector(\text{“chicago suntimes”})$
$vector(\text{“new york”}) + vector(\text{“newspaper”})$	$vector(\text{“new york times”})$
$vector(\text{“san francisco”}) + vector(\text{“newspaper”})$	$vector(\text{“la times”})$
$vector(\text{“beyonce”}) + vector(\text{“pictures”})$	$vector(\text{“beyonce images”})$
$vector(\text{“beyonce”}) + vector(\text{“videos”})$	$vector(\text{“beyonce videos”})$
$vector(\text{“beyonce”}) + vector(\text{“net worth”})$	$vector(\text{“jaden smith net worth”})$
$vector(\text{“www.facebook.com”}) - vector(\text{“facebook”}) + vector(\text{“twitter”})$	$vector(\text{“www.twitter.com”})$
$vector(\text{“www.facebook.com”}) - vector(\text{“facebook”}) + vector(\text{“gmail”})$	$vector(\text{“www.googlemail.com”})$
$vector(\text{“www.facebook.com”}) - vector(\text{“facebook”}) + vector(\text{“hotmail”})$	$vector(\text{“www.hotmail.xom”})$
$vector(\text{“how tall is tom cruise”}) - vector(\text{“tom cruise”}) + vector(\text{“tom selleck”})$	$vector(\text{“how tall is tom selleck”})$
$vector(\text{“how old is gwen stefani”}) - vector(\text{“gwen stefani”}) + vector(\text{“meghan trainor”})$	$vector(\text{“how old is meghan trainor”})$
$vector(\text{“how old is gwen stefani”}) - vector(\text{“gwen stefani”}) + vector(\text{“ariana grande”})$	$vector(\text{“how old is ariana grande 2014”})$
$vector(\text{“university of washington”}) - vector(\text{“seattle”}) + vector(\text{“chicago”})$	$vector(\text{“chicago state university”})$
$vector(\text{“university of washington”}) - vector(\text{“seattle”}) + vector(\text{“denver”})$	$vector(\text{“university of colorado”})$
$vector(\text{“university of washington”}) - vector(\text{“seattle”}) + vector(\text{“detroit”})$	$vector(\text{“northern illinois university”})$

(Mitra, 2015)

Analogies for session modelling

soundcloud	→	www.soundcloud.com
coasthills coop	→	www.coasthills.coop
american express	→	www.barclaycardus.com login
duke energy bill pay	→	www.duke-energy.com pay my bill
cool math games	→	www.coolmath.com
majesty shih tzu	→	what is a majesty shih tzu
hard drive dock	→	what is a hard drive dock
lugia in leaf green	→	where is lugia in leaf green
red river log jam	→	what is th red river log jam
prowl	→	what does prowl mean
rottweiler	→	rottweiler facebook
sundry	→	sundry expense
elections	→	florida governor race 2014
pleurisy	→	pleurisy shoulder pain
elections	→	2014 rowan county election results
cna classes	→	cna classes in lexington tennessee
container services inc	→	container services ringgold ga
enclosed trailers for sale	→	enclosed trailers for sale north carolina
firewood for sale	→	firewood for sale in asheboro nc
us senate race in colorado	→	us senate race in georgia
siol	→	facebook
cowboy bebop	→	facebook
mr doob	→	google
great west 100 west 29th	→	facebook
avatar dragons	→	youtube

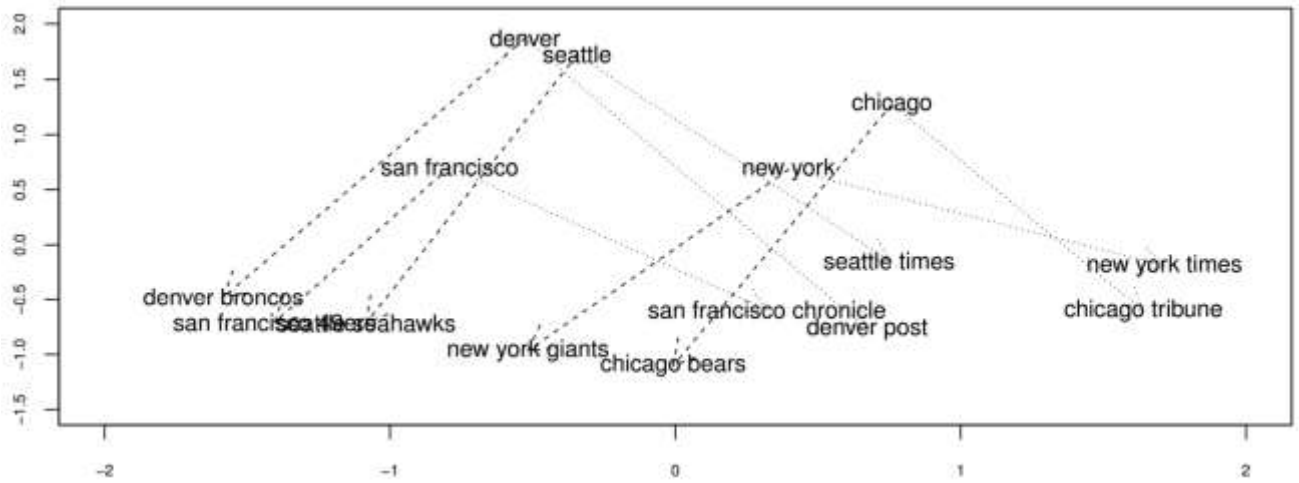
This property was shown to be useful for modelling session context

"london" → "things to do in london"

is similar to

"new york" → "new york tourist attractions"

Possible scope for modelling sessions as paths in the embedding space?



(Mitra, 2015)

Microsoft

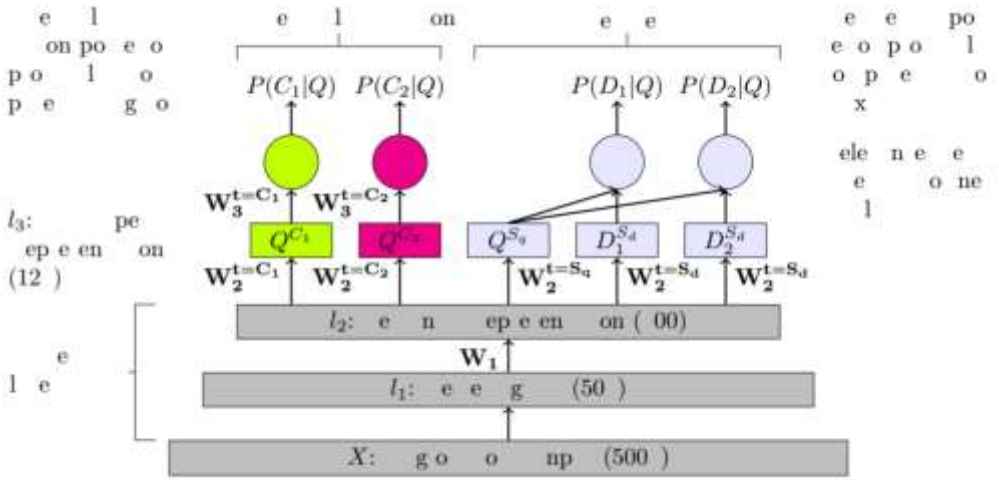
Multi-task learning

DSSM model trained under a multi-task setting

e.g., query classification and ranking

Lower layer weights are shared, top layer specific to the task

Leverages cross-task data as well as reduces over-fitting to particular task – learnt embedding may be useful for related tasks not directly trained for



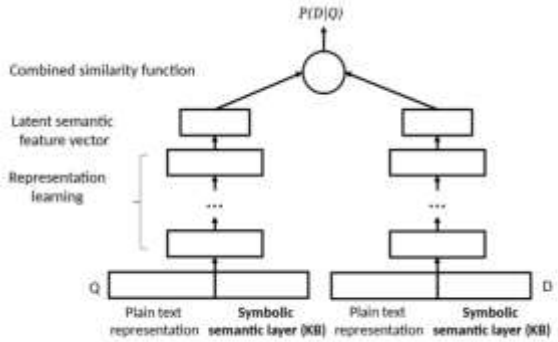
(Liu et al., 2015)

Microsoft

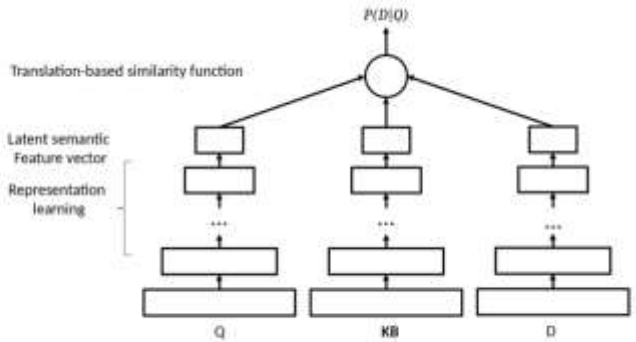
Neural approach for Knowledge-based IR

Incorporating distributed representations of KB entities in deep neural retrieval models

Advocates that KBs can either support learning of latent representations of text or serve as a semantic translator between their latent representations



(a) Enhanced representation using KB for IR



(b) The KB translation model

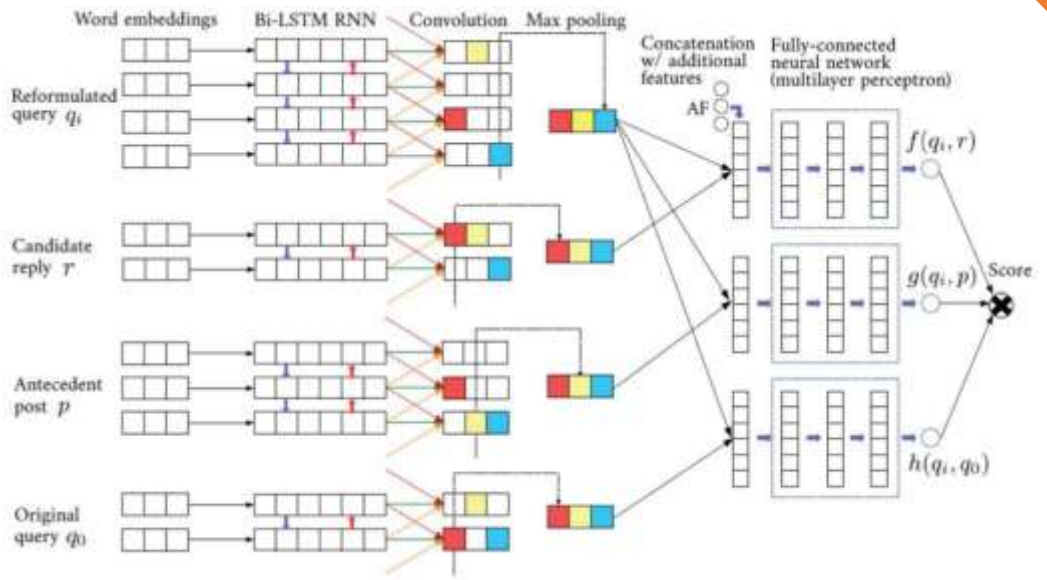
(Nguyen et al., 2016)

Conversational response retrieval

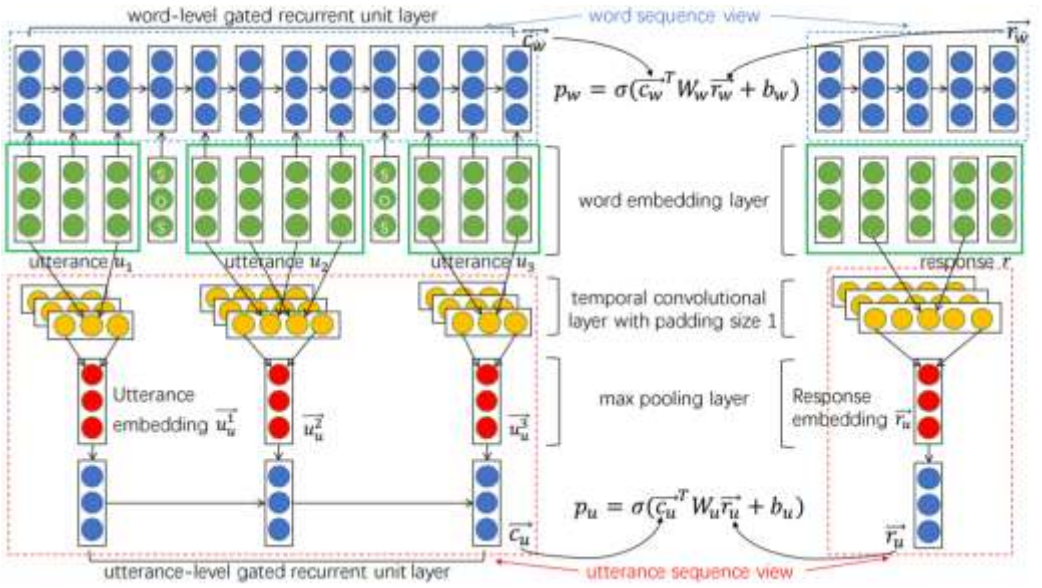
Ranking responses for conversational systems

Interesting challenges in modelling utterance level discourse structure and context

Can multi-turn conversational tasks become a key playground for neural retrieval models?



(Yan et al., 2016)



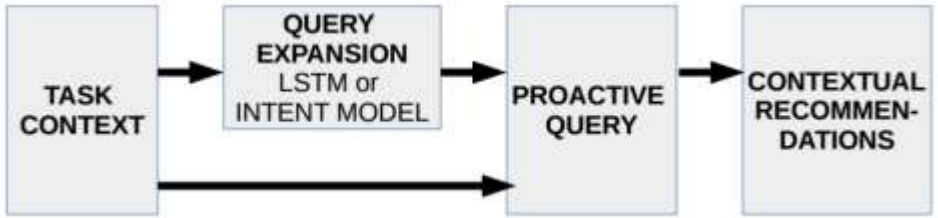
(Zhou et al., 2016)

Microsoft

Proactive retrieval

Given current task context generate a proactive query and retrieve recommendations

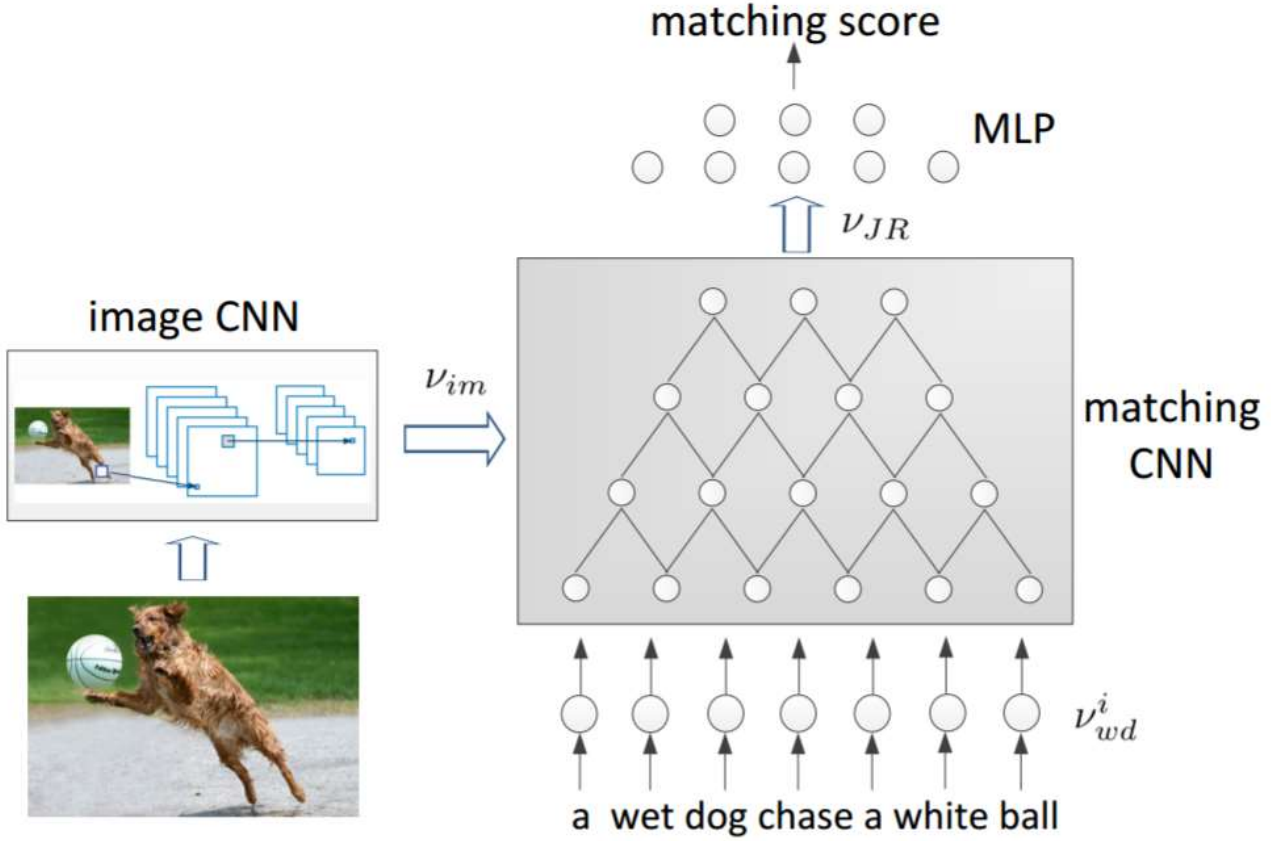
In this particular paper authors focused on recommendations during a document authoring task



(Luukkonen et al., 2016)

Multimodal retrieval

This tutorial is focused on text but neural representation learning is also leading towards breakthroughs in multimodal retrieval



(Ma et al., 2015)

Useful Resources

Next gen neural IR models (using reinforcement learning or adversarial learning) may need to perform retrieval as part of the model training / evaluation step

Pyndri for Python: https://github.com/cvangysel/pyndri

Luandri for LUA / Torch: https://github.com/bmitra-msft/Luandri

public word embedding datasets

Model	Description	
Word2vec	~3M words vocabulary trained on Google News dataset	Link
GloVe	Multiple datasets with ~400K-2M word vocabulary	Link
DESM	IN + OUT embeddings for ~3M words trained on Bing query logs	Link
NTLM	Multiple datasets with ~50K-3M word vocabulary	Link

bit.ly/
NeuIRTutorial-WSDM2017

send us your questions and feedback during or after the tutorial

Bhaskar Mitra

@UnderdogGeek

bmitra@microsoft.com

Nick Craswell

@nick_craswell

nickcr@microsoft.com

Microsoft