Lecture 8. Neural Information Retrieval

Foundations and Trends® in
Information Retrieval
13:1

An Introduction to Neural Information Retrieval

Bhaskar Mitra and Nick Craswell

now
the essence of knowledge

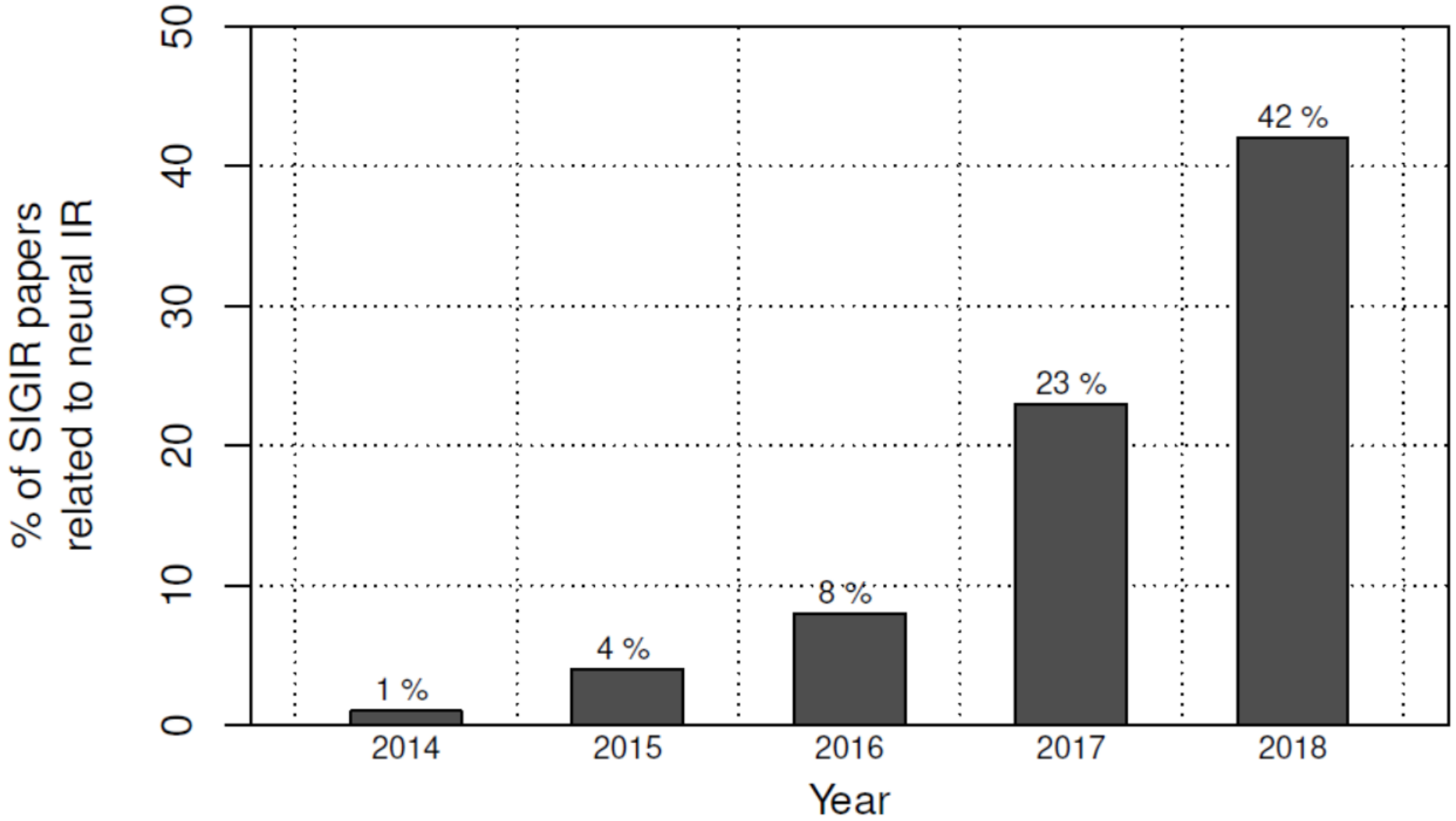
Neural IR

- Application of shallow or deep neural networks to retrieval tasks
- IR scenarios
  - Ad-hoc retrieval
  - recommender systems
  - multi media search
  - conversational systems: generate answer in response to natural language questions
- Ranking documents in response to a query (in this lecture)

Neural Models for IR

- Use vector representations of text
- Contain a large number of parameters that need to tuned
- Compared with traditional learning to rank (LTR)
  - LTR: train ML models over a set of hand-crafted features
  - NM: accept the raw text of a query and document as input

% of Neural IR Papers at SIGIR



Meaning	Notation
Single query	q
Single document	d
Set of queries	Q
Collection of documents	D
Term in query q	t_q
Term in document d	t_d
Full vocabulary of all terms	T
Set of ranked results retrieved for query q	R_q
Result tuple (document d at rank i)	$\langle i, d \rangle$, where $\langle i, d \rangle \in R_q$
Relevance label of document d for query q	$rel_q(d)$
d_i is more relevant than d_j for query q	$rel_q(d_i) > rel_q(d_j)$, or $d_i \succ_q d_j$
Frequency of term t in document d	$tf(t, d)$
Number of documents that contain term t	$df(t)$
Vector representation of text z	$\vec{v}_z$
Probability function for an event $\mathcal{E}$	$p(\mathcal{E})$

Traditional IR Models

b=0: BM15, *b*=1:BM11

$$\mathcal{B}_{i,j} = \frac{(K_1+1)f_{i,j}}{K_1\left[(1-b)+b\frac{len(d_j)}{avg\_doclen}\right]+f_{i,j}}$$

$$sim_{BM25}(d_j,q) \sim \sum_{k_i[q,d_j]} \mathcal{B}_{i,j} \times \log\left(\frac{N-n_i+0.5}{n_i+0.5}\right)$$

BM25

$$BM25(q,d) = \sum_{t_q \in q} idf(t_q) \cdot \frac{tf(t_q,d)\cdot(k_1+1)}{tf(t_q,d)+k_1\cdot\left(1-b+b\cdot\frac{|d|}{avgdl}\right)}$$

k1 is in the range [1.2,2.0] and b=0.75

$$idf(t) = \log\frac{|D|-df(t)+0.5}{df(t)+0.5}$$

Language Modeling (LM)

$$p(d|q) = \frac{p(q|d).p(d)}{\sum_{\bar{d} \in D} p(q|\bar{d}).p(\bar{d})}$$

$$\propto p(q|d).p(d)$$

$$= p(q|d), \text{ assuming p(d) is uniform}$$

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

$$P(q|M_j) = \prod_{k_i \in q} P(k_i|M_j)$$

$$p(t_q|d) = \frac{tf(t_q, d)}{|d|}$$

$$P^s_{\in}(k_i|M_j,\lambda) = (1-\lambda)\frac{f_{i,j}}{\sum_i f_{i,j}} + \lambda\frac{F_i}{\sum_i F_i}$$

Smoothing in LM

1. Jelinek-Mercer smoothing (Jelinek and Mercer, 1980)

$$p(t_q|d) = \left(\lambda\frac{tf(t_q,d)}{|d|} + (1-\lambda)\frac{\sum_{\bar{d}\in D} tf(t_q,\bar{d})}{\sum_{\bar{d}\in D}|\bar{d}|}\right)$$

2. Dirichlet Prior Smoothing (MacKay and Peto, 1995)

$$p(t_q|d) = \left(tf(t_q,d) + \mu\frac{\sum_{\bar{d}\in D} tf(t_q,\bar{d})}{\sum_{\bar{d}\in D}|\bar{d}|}\right)/\left(|d|+\mu\right)$$

$$P^s_{\in}(k_i|M_j,\lambda) = \frac{f_{i,j} + \lambda\frac{F_i}{\sum_i F_i}}{\sum_i f_{i,j} + \lambda}$$

BM25 and LM

- Both BM25 and LM estimate document relevance based on the count of only the query terms in the document.
- The position of these occurrences and the relationships with other terms in the document are ignored.

Translation Model

- Assume the query q is being generated via a translation process from the document d.
- Relevance from non-query terms in the document

$$p(t_q|d) = \sum_{t_d \in d} p(t_q|t_d) \cdot p(t_d|d)$$

- Estimate $p(t_q|t_d)$ from query-document paired data similar to techniques in statistical machine translation
- BM25, LM, and Translation Model
  - Do not consider proximity between query terms.

Jelinek-Mercer smoothing (Jelinek and Mercer, 1980)

$$p(t_q|d) = \left(\lambda \frac{tf(t_q,d)}{|d|} + (1-\lambda)\frac{\sum_{\bar{d}\in D} tf(t_q,\bar{d})}{\sum_{\bar{d}\in D}|\bar{d}|}\right)$$

Dependency Model

$$DM(q,d) = (1-\lambda_{ow}-\lambda_{uw}) \sum_{t_q \in q} \log\left((1-\alpha_d)\frac{tf(t_q,d)}{|d|} + \alpha_d \frac{\sum_{\bar{d}\in D} tf(t_q,\bar{d})}{\sum_{\bar{d}\in D}|\bar{d}|}\right)$$
$$+\lambda_{ow} \sum_{c_q \in ow(q)} \log\left((1-\alpha_d)\frac{tf_{\#1}(c_q,d)}{|d|} + \alpha_d \frac{\sum_{\bar{d}\in D} tf_{\#1}(c_q,\bar{d})}{\sum_{\bar{d}\in D}|\bar{d}|}\right)$$
$$+\lambda_{uw} \sum_{c_q \in uw(q)} \log\left((1-\alpha_d)\frac{tf_{\#uwN}(c_q,d)}{|d|} + \alpha_d \frac{\sum_{\bar{d}\in D} tf_{\#uwN}(c_q,\bar{d})}{\sum_{\bar{d}\in D}|\bar{d}|}\right)$$

where, $ow(q)$ and $uw(q)$ are the set of all contiguous n-grams (or phrases) and the set of all bags of terms that can be generated from query q. $tf_{\#1}$ and $tf_{\#uwN}$ are the ordered-window and unordered-window operators from Indri (Strohman *et al.*, 2005). Finally, λ_{ow} and λ_{uw} are the tuneable parameters of the model.

Relevance Model (RM)

$$R(d;Q) = KL(M_Q \,||\, M_d) = \sum_{t \in V} P(t \mid M_Q) \log \frac{P(t \mid M_Q)}{P(t \mid M_d)}$$

- Pseudo Relevance Feedback (PRF)
- Compute the KL divergence between the query language model θ_q and the document language model θ_d.

$$score(q,d) = -\sum_{t \in T} p(t|\theta_q) \log \frac{p(t|\theta_q)}{p(t|\theta_d)}$$

Without PRF,

$$p(t|\theta_q) = \frac{tf(t,q)}{|q|}$$

TF-IDF and language modelling based approaches only consider the count of query term occurrences in the document. Dependence model considers phrasal matches. Translation and PRF models can deal with vocabulary mismatch between query and document.

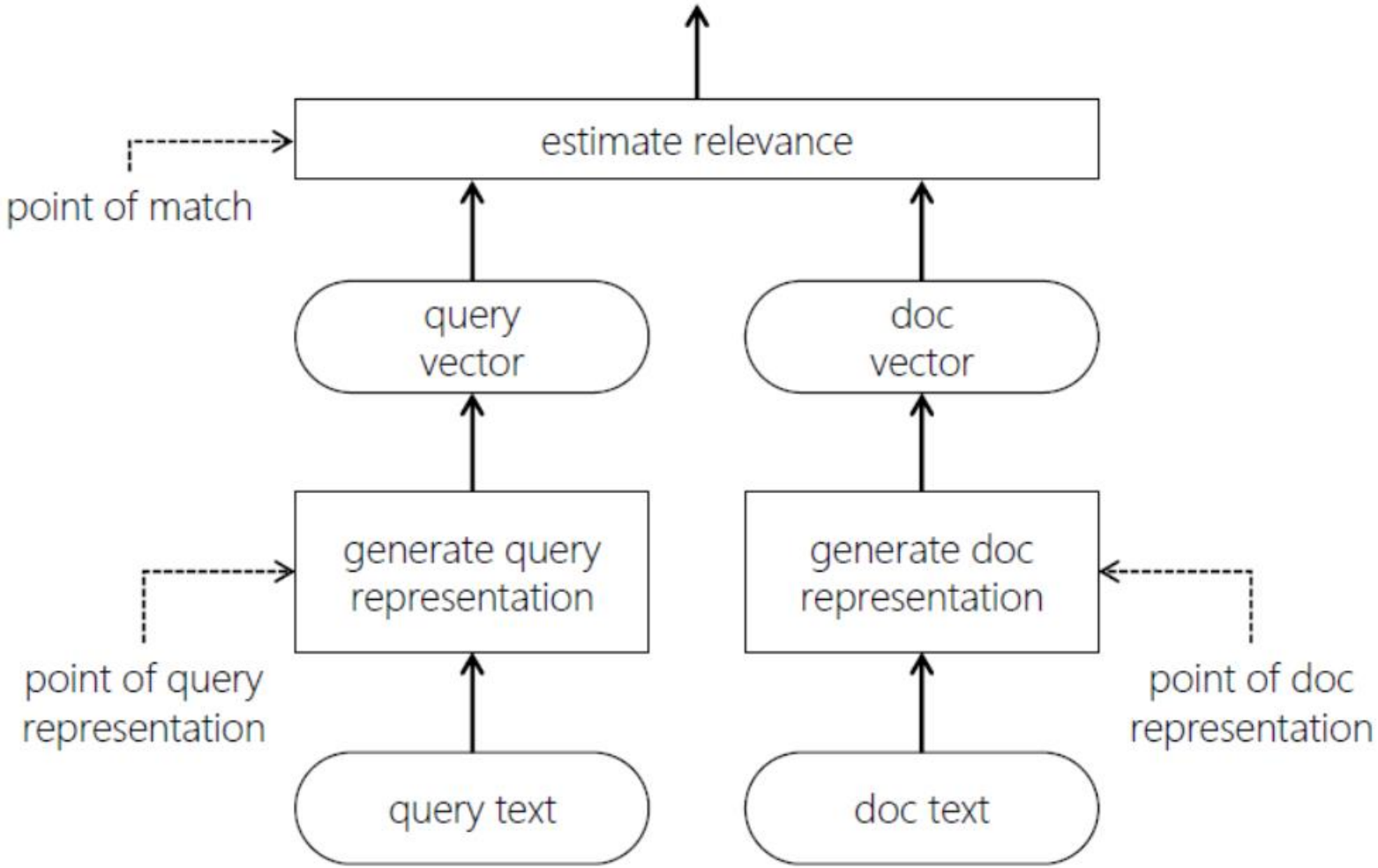
Neural Approach to IR

Document Ranking

- Generate a representation of the query that specifies the information need
- Generate a representation of the document that captures the distribution over information contained
- Match the query and the document representations to estimate their mutual relevance

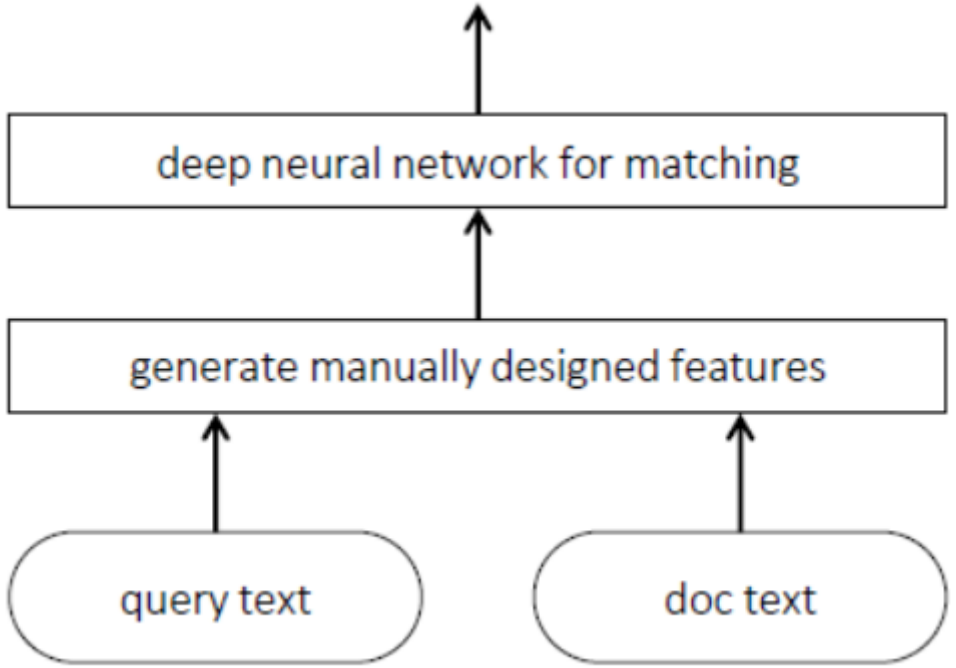
Neural IR models can be categorized based on whether they influence the query representation, the document representation, the relevance estimation, or a combination of these steps.

Neural Approaches



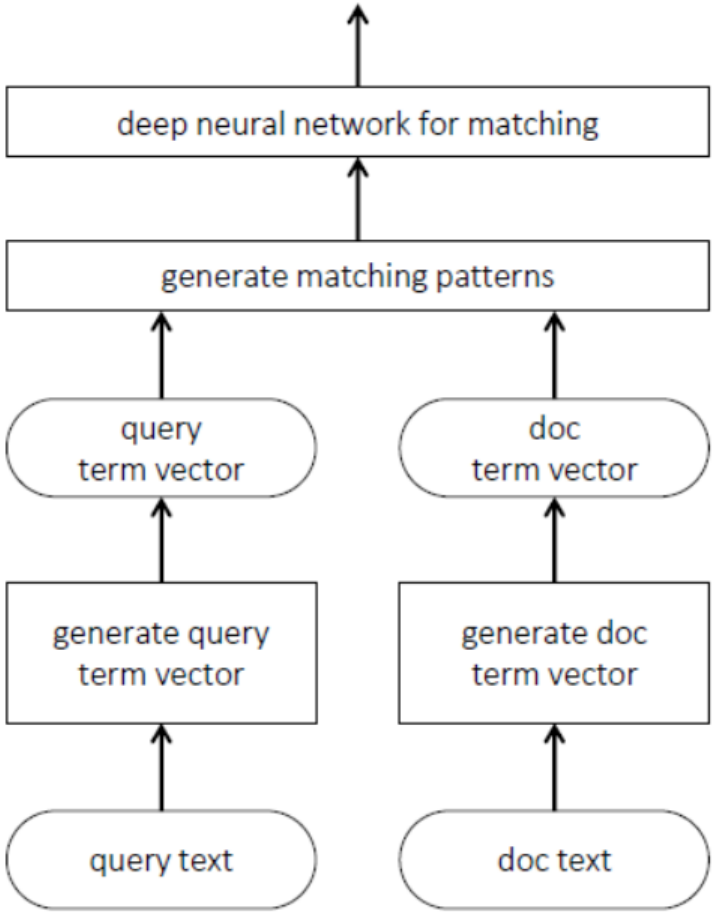
At the Point of Matching

- Learning to rank using manually designed features



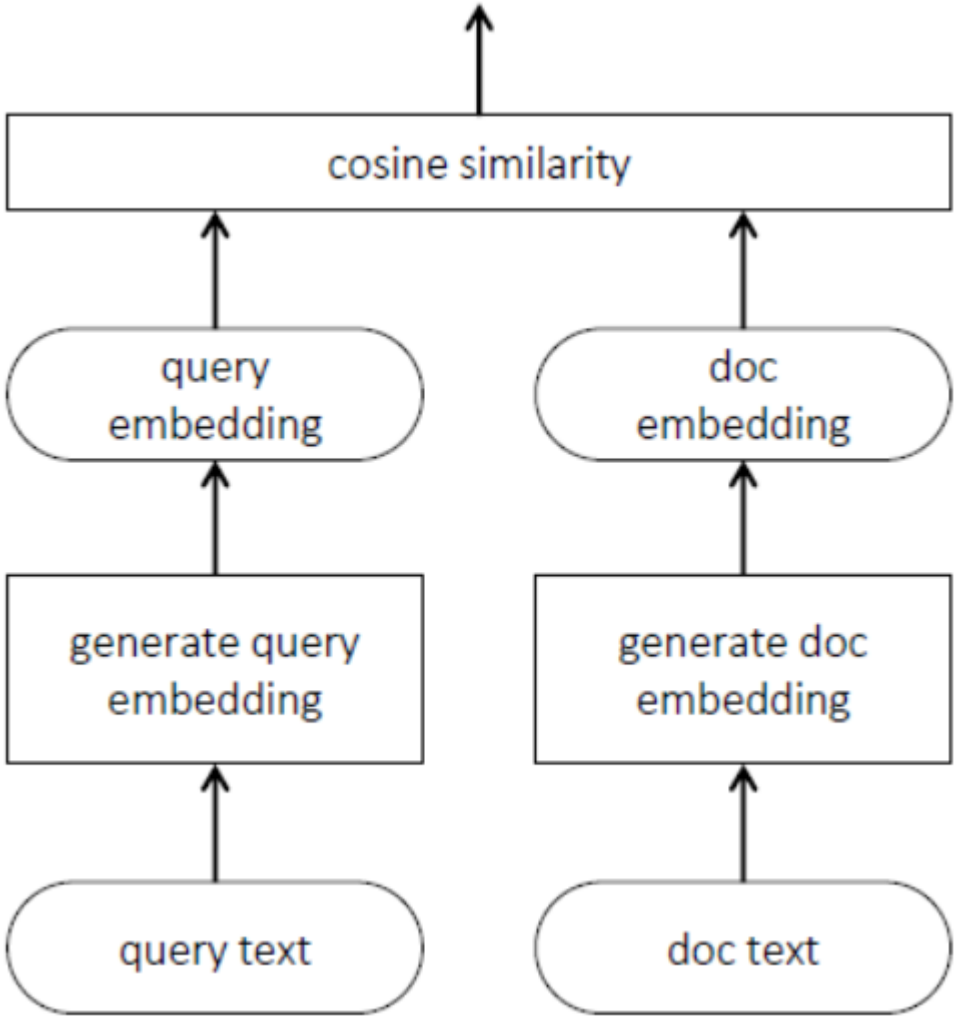
At the Point of Matching

- DNN models to estimate relevance based on patterns of exact query term matches in the documents

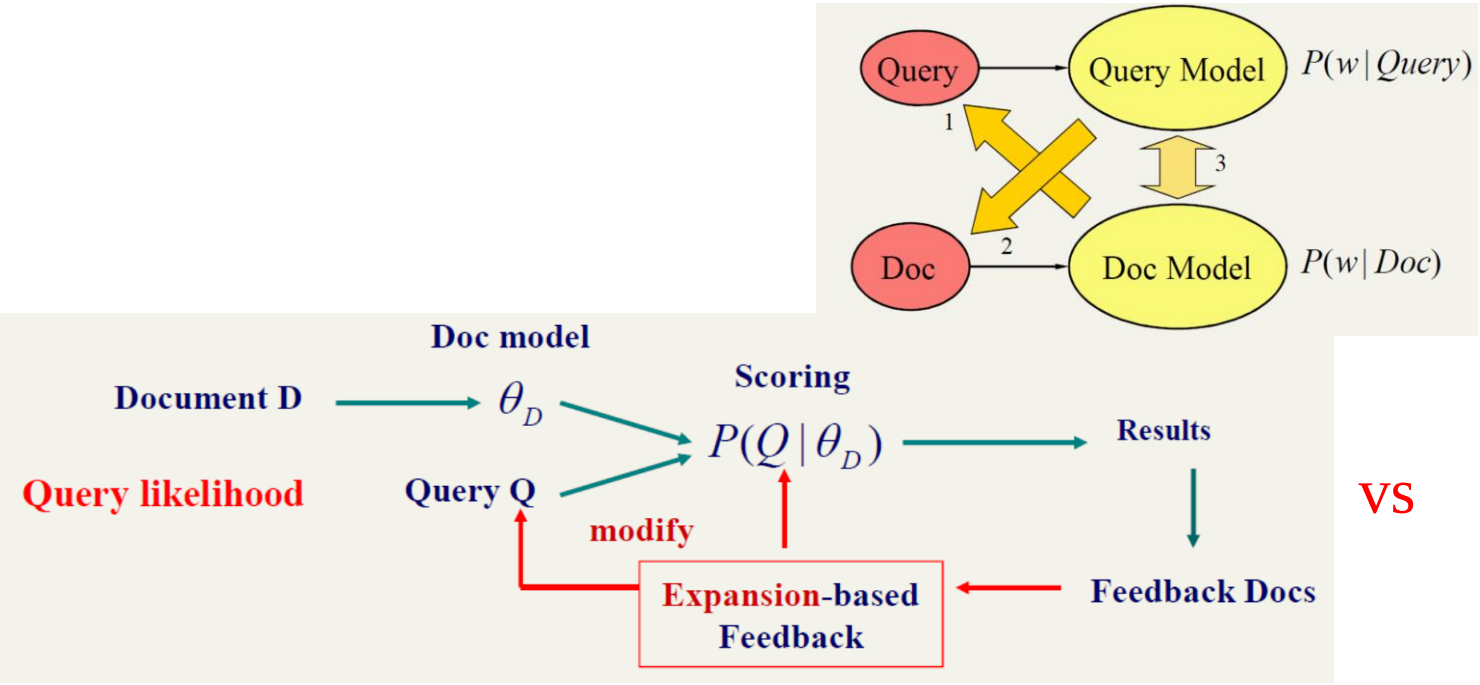


At the Point of Query/Doc Representation

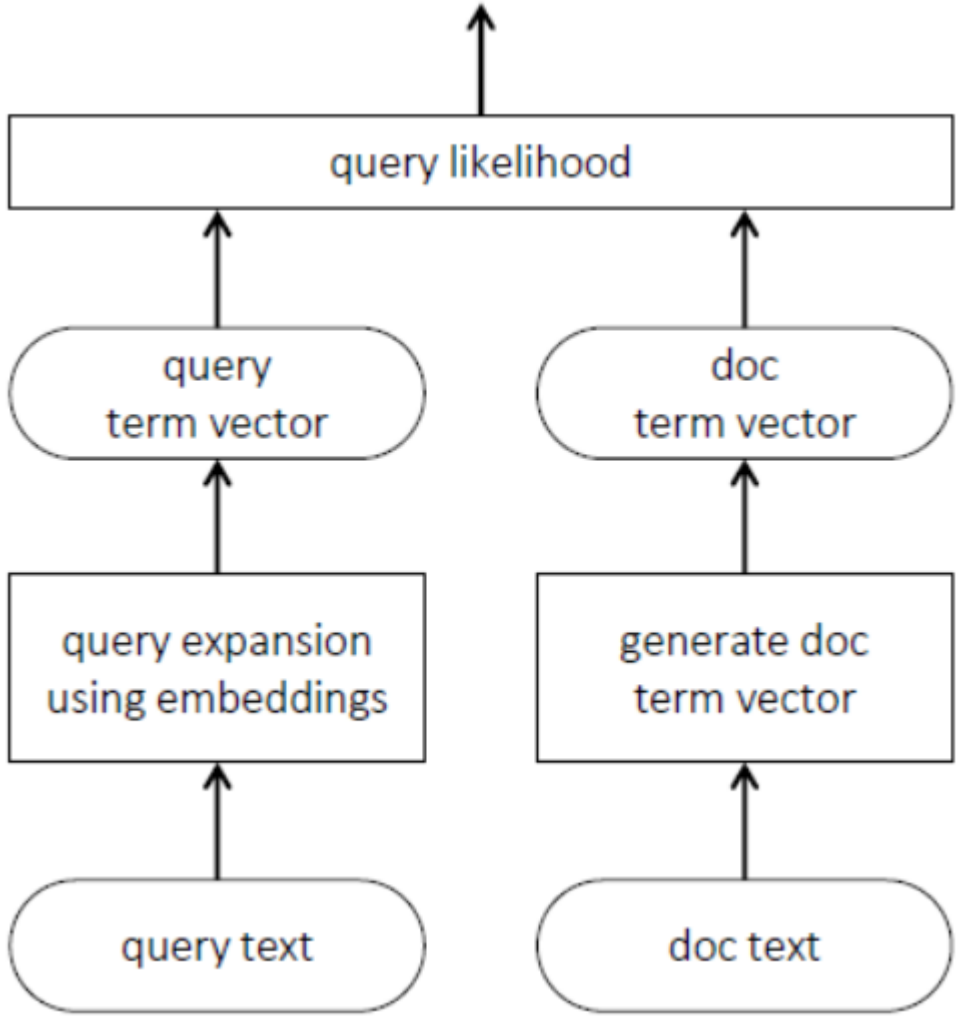
Learn embedding and use them within traditional IR models or in conjunction with simple similarity metrics such as cosine similarity



Query Expansion Using Neural Embedding



VS



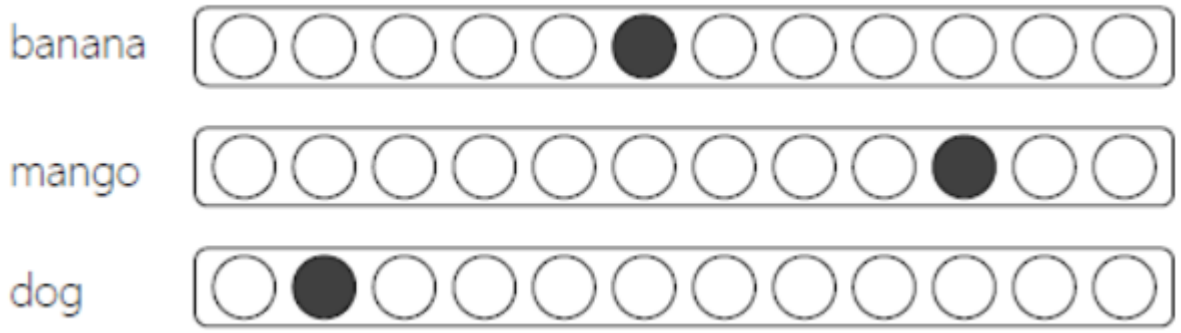
Unsupervised learning of term representations

Term Representation

- Terms are the smallest unit of representation for indexing and retrieval.
- Many IR models, both non-neural and neural, focus on learning good vector representations of terms.
- Different vector representation exhibit different levels of generalization.
- Different representation schemes derive different notions of similarity between terms.
  - Operate over fixed-size vocabularies?
  - Properties of compositionality (terms → passages → documents)

Local Representations

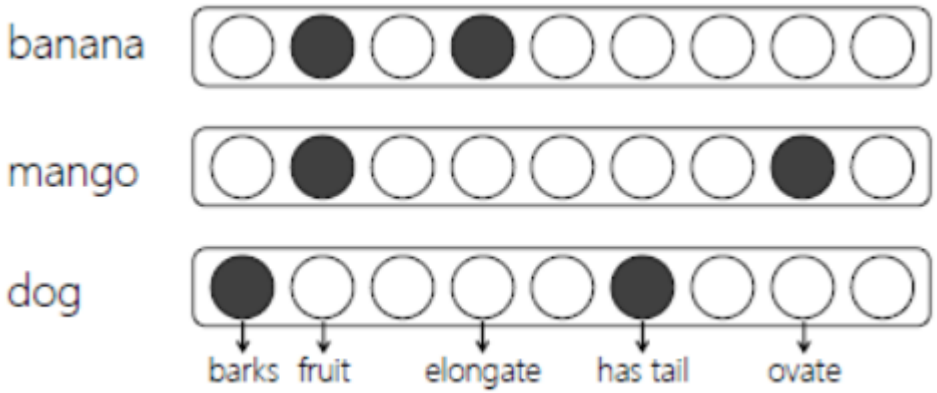
- One-hot representation
  - Every term in a fixed size vocabulary *T* is represented by a binary vector, $\vec{v} \in \{0,1\}^{|T|}$, where only one of the values in the vector is one and all the others are set to zero.
  - Each position in the vector $\vec{v}$ corresponds to a term.
  - Terms outside of the vocabulary have no representation or are denoted by a special “UNK” symbol.



Banana, mango, and dog are distinct items.

Distributed Representation

- Every term is represented by a vector $\vec{v} \in \mathbb{R}^{|k|}$.
- $\vec{v}$ can be a sparse or a dense vector.
- $\vec{v}$ can be a vector of hand-crafted features or a latent representation.
- Define similarity between terms



Banana and mango are both fruits, but dog is different.

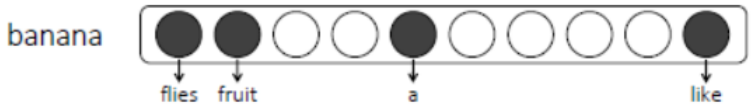
Feature-based Distributed Representations

- Distributional Hypothesis
  - Terms that are used (or occur) in similar context tend to be semantically similar.
- Distributional Semantics
  - A word is characterized by the company it keeps.
- The distribution of different types of context may model different semantics of a term.
  - Documents containing the term
  - Neighboring terms in a window
  - Neighboring terms with distance
- A term can have a distributed representation based on non-distributional features
  - The character trigraphs in the term itself
  - Parts of speech classification

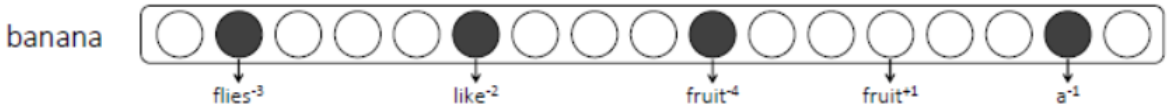
Examples of Feature-based Distributed Representations

- Example: *Time flies like an arrow*; *fruit flies like a banana*.

(a) In-document features



(b) Neighbouring-term features



(c) Neighbouring-term w/ distance features

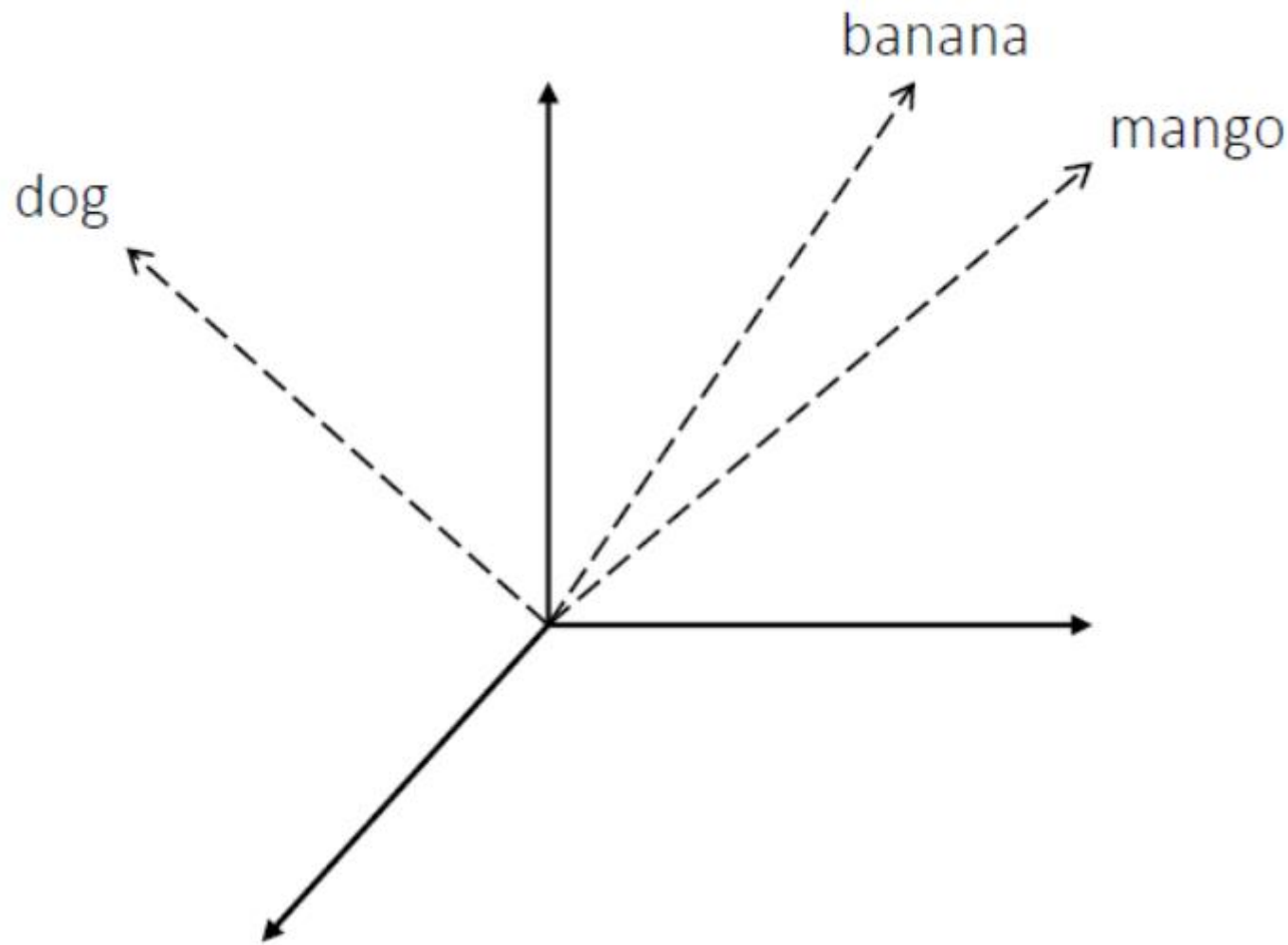
(d) Character-trigraph features

Under a local or one-hot representation every item is distinct. But when items have distributed or feature based representation, then the similarity between two items is determined based on the similarity between their features.

Observed vs. Latent

- Observed (or explicit) vector representation
  - Vectors are high-dimensional, sparse, and based on observable features.
- Latent vector spaces, or embeddings
  - Vectors are dense, small ($k \ll |T|$), and learnt from data.
- Distance metrics can be used to define the similarity between terms.
  - cosine similarity

$$sim(\vec{v}_i, \vec{v}_j) = cos(\vec{v}_i, \vec{v}_j) = \frac{\vec{v}_i^{\mathsf{T}} \vec{v}_j}{\|\vec{v}_i\| \|\vec{v}_j\|}$$


banana
mango
dog

Compositionality

- Distributed representations of items are derived from local or distributed representation of its parts.
  - A document can be represented by the sum of the on-hot vectors or embeddings corresponding to the terms in the document.
  - Distributed bag-of-terms representation
  - The character trigraph representation of terms is an aggregation over the one-hot representations of the constituent trigraphs.

Notions of Similarity and Choice of Features

- Typical or type-base notion of similarity
  - Compare “Seattle” with “Sydney”
  - Both cities
  - The “in documents” features naturally lead to a topical sense of similarity between the terms.
- Topical sense of relatedness
  - Compare “Seattle” with “Seahawks”
  - Seahawks: Seattle’s football team
  - The “neighboring terms with distance” features gives rise to a more typical notion of relatedness.

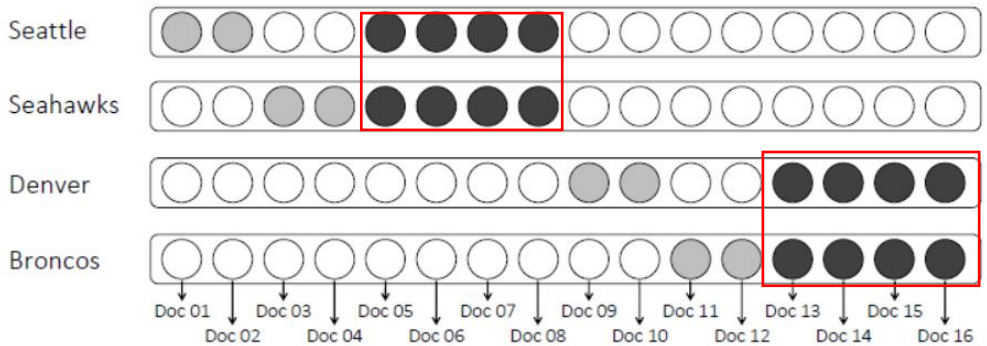
西雅圖海鷹隊（Seattle Seahawks）是一支專業的美式足球隊，成立於1976年，總部位於美國華盛頓州的西雅圖市。

Using “neighboring terms” without the inter-term distances as features???
A mixture of topical and typical relationships

主題：職業美式足球球隊

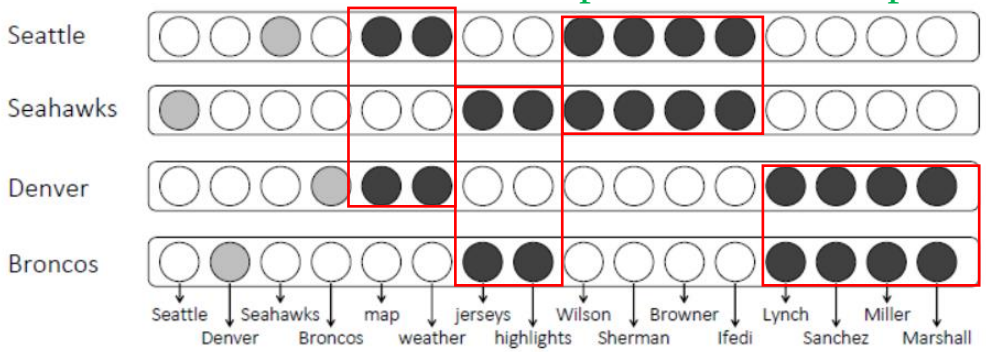
西雅圖海鷹隊 vs.丹佛野馬隊

Sample documents			
doc 01	Seattle map	doc 09	Denver map
doc 02	Seattle weather	doc 10	Denver weather
doc 03	Seahawks jerseys	doc 11	Broncos jerseys
doc 04	Seahawks highlights	doc 12	Broncos highlights
doc 05	Seattle Seahawks Wilson	doc 13	Denver Broncos Lynch
doc 06	Seattle Seahawks Sherman	doc 14	Denver Broncos Sanchez
doc 07	Seattle Seahawks Browner	doc 15	Denver Broncos Miller
doc 08	Seattle Seahawks Ifedi	doc 16	Denver Broncos Marshall



(a) "In-documents" features

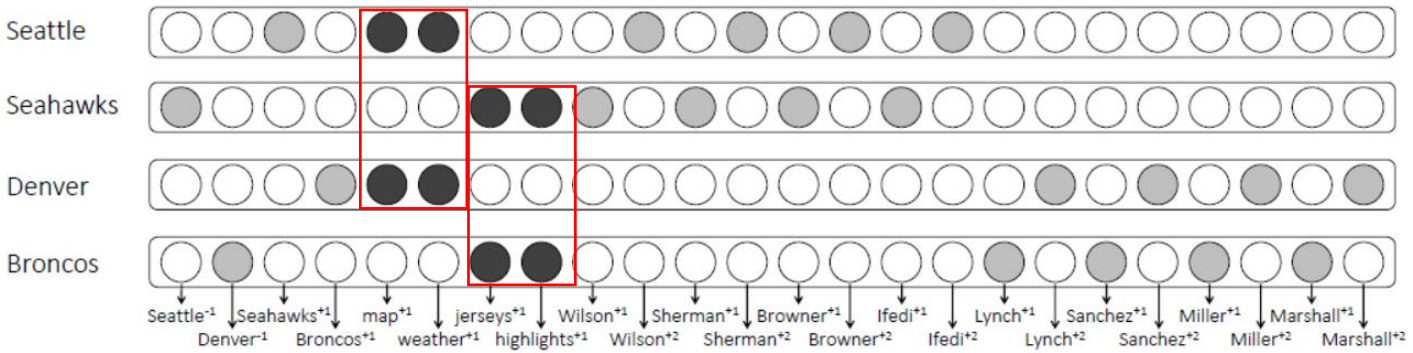
topical relationships



(b) "Neighbouring terms" features

型態：城市球隊

西雅圖海鷹隊 vs.丹佛野馬隊



(c) "Neighbouring terms w/ distances" features

typical relationships

綜合主題和型態關係

topical and typical relationships

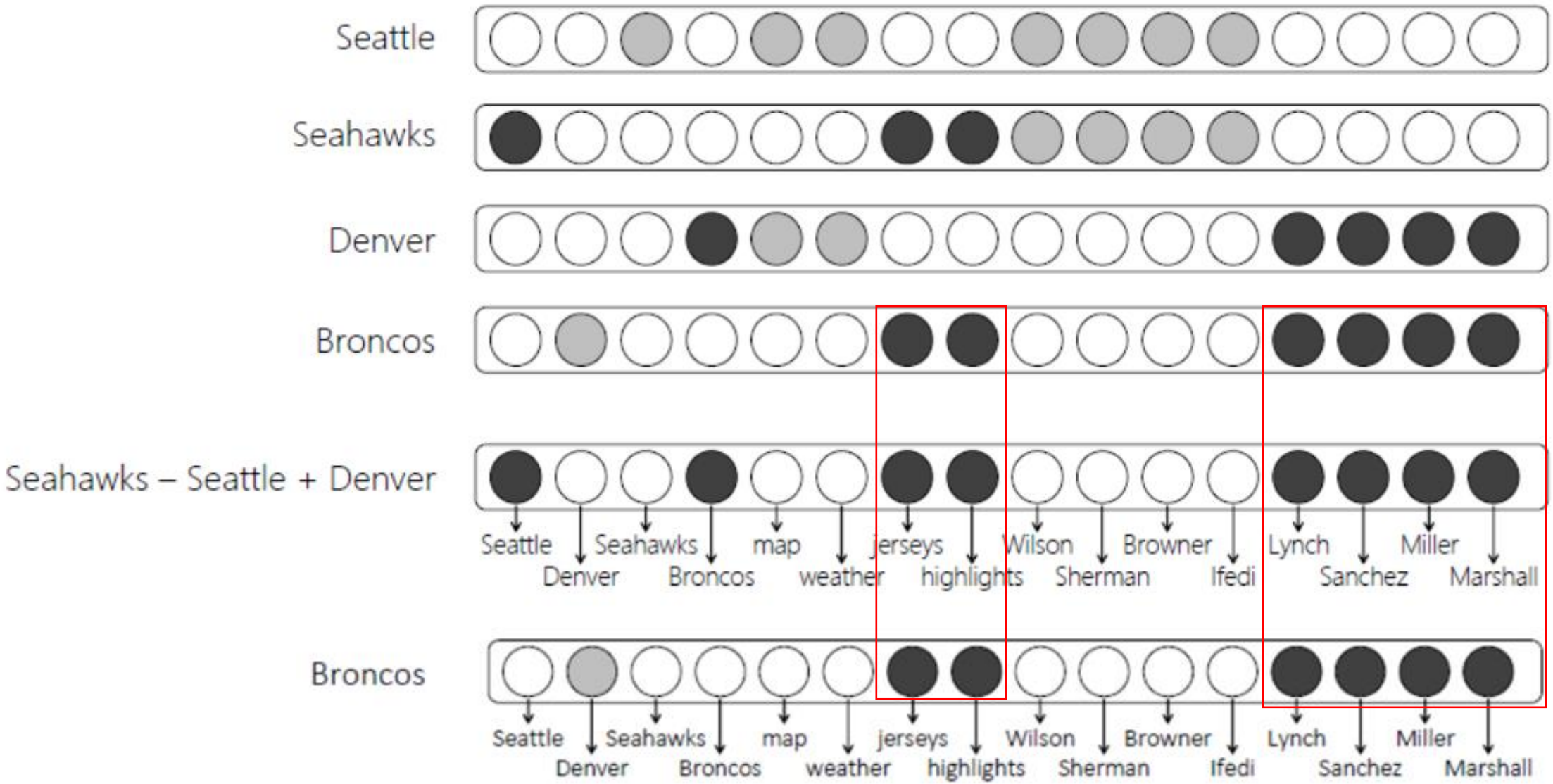
Different vector representations capture different notions of similarity between terms. "Seattle" may be closer to either "Sydney" (*typically* similar) or "Seahawks" (*topically* similar) depending on the choice of vector dimensions.

Observed Feature Spaces

- Categorized by distributional features
  - in document, neighboring terms with or without distances, etc.
- Categorized by weighting schemes
  - TF-IDF, positive pointwise mutual information, etc.
- Term analogy task using simple vector operations
  - "*man* is to *woman* as *king* is to _____ ?"
  - Performed by simple vector operations of the form followed by a nearest-neighboring search

$$\vec{v}_{Seahawks} - \vec{v}_{Seattle} + \vec{v}_{Denver} \approx \vec{v}_{Broncos}$$

丹佛野馬是一支位於科羅拉多州丹佛的職業美式足球隊


Seattle
Seahawks
Denver
Broncos
Seahawks – Seattle + Denver
Seattle
Denver
Seahawks
Broncos
map
weather
jerseys
highlights
Wilson
Sherman
Browner
Ifedi
Lynch
Sanchez
Miller
Marshall
Broncos
Seattle
Denver
Seahawks
Broncos
map
weather
jerseys
highlights
Wilson
Sherman
Browner
Ifedi
Lynch
Sanchez
Miller
Marshall

Latent Feature Space

- Observed feature space based on distributional features
  - The resultant representation is highly sparse and highly dimensional.
  - The number of dimensions may be the same as the vocabulary size.
- Embedding
  - A representation of items in a new space such that the properties of – and the relationships between – the items are preserved from the original representation.
- Approaches
  - Factorizing the term-feature matrix, such as LSA
  - Using gradient descent based methods

Latent Semantic Analysis (LSA)

- Perform singular value decomposition (SVD) on a term-document (or term-passage) matrix to obtain its low-rank approximation.

$$\underset{(\vec{t}_i^{\intercal})\rightarrow}{} \underset{(\vec{d}_j)\downarrow}{X} = \underset{(\vec{t}_i^{\intercal})\rightarrow}{} \underset{}{U} \cdot \underset{}{\Sigma} \cdot \underset{(\vec{d}_j)\downarrow}{V^{\intercal}}$$

$$\begin{bmatrix} x_{1,1} & \dots & x_{1,|D|} \\ \vdots & \ddots & \vdots \\ x_{|T|,1} & \dots & x_{|T|,|D|} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \\ \vec{u}_1 \\ \\ \end{bmatrix} & \dots & \begin{bmatrix} \\ \vec{u}_l \\ \\ \end{bmatrix} \end{bmatrix} \cdot \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_l \end{bmatrix} \cdot \begin{bmatrix} [\ \vec{v}_1\ ] \\ \vdots \\ [\ \vec{v}_l\ ] \end{bmatrix}$$

$\sigma_1, \dots, \sigma_l$, $\vec{u}_1, \dots, \vec{u}_l$, and $\vec{v}_1, \dots, \vec{v}_l$ are the singular values, and the left and the right singular vectors, respectively. The k largest singular values—and corresponding singular vectors from U and V—is the rank k approximation of X ($X_k = U_k \Sigma_k V_k^T$) and $\Sigma_k \vec{t}_i$ is the embedding for the i^{th} term.

Probabilistic Latent Semantic Analysis (PLSA)

- Learn low-dimensional representations of terms and documents by modelling their co-occurrence $p(t,d)$ as follows.

$$p(t,d) = p(d) \sum_{c \in C} p(c|d) P(t|c)$$

where, C is the set of latent topics—and the number of topics $|C|$ is a hyperparameter of the model. Both $p(c|d)$ and $P(t|c)$ are modelled as multinomial distributions and their parameters are typically learned using the EM algorithm (Dempster *et al.*, 1977).

- After learning the parameters of the model, a term t_i can be represented as a distribution over the latent topics

$$[p(c_0|t_i), \ldots, p(c_{|C|-1}|t_i)]$$

Latent Dirichlet Allocation (LDA)

- Each document is represented by a Dirichlet prior instead of a fixed variable.

M denotes the number of documents

N is number of words in a given document (document *i* has N_i words)

α is the parameter of the Dirichlet prior on the per-document topic distributions

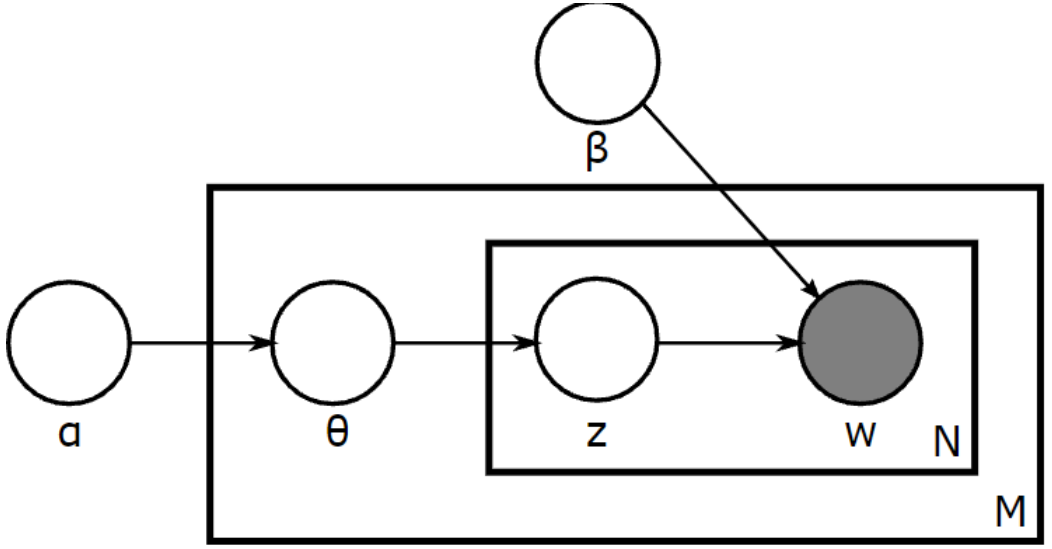
β is the parameter of the Dirichlet prior on the per-topic word distribution

θ_i is the topic distribution for document *i*

φ_k is the word distribution for topic *k*

z_{ij} is the topic for the *j*-th word in document *i*

w_{ij} is the specific word.



Documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over all the words.

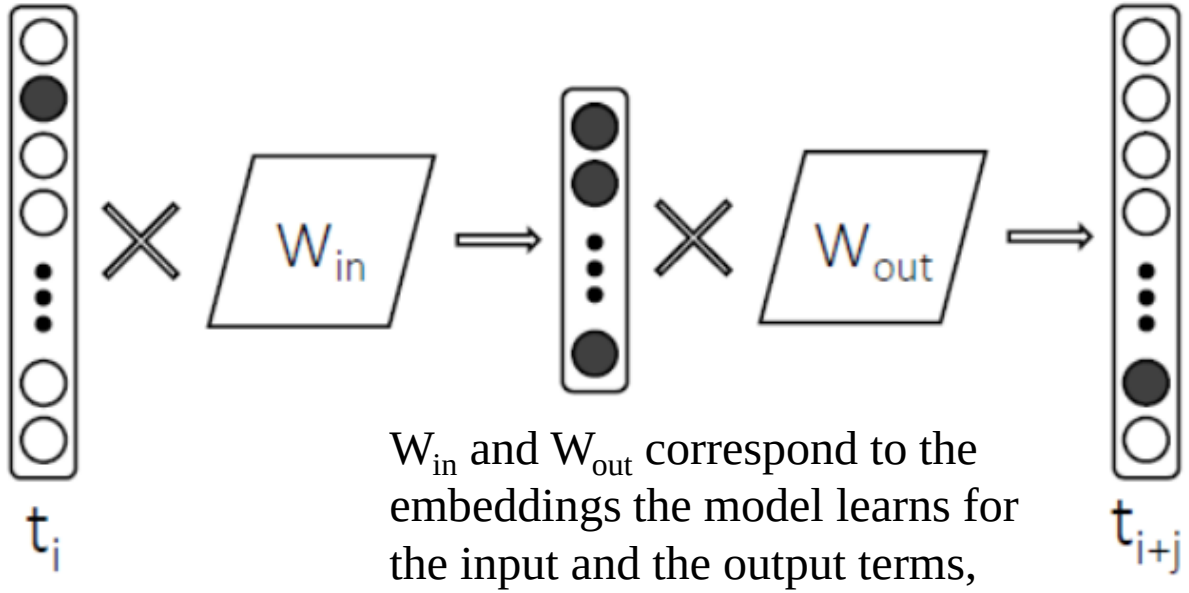
https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation

Neural Term Embedding

- Models are trained by setting up a prediction task.
- Instead of factorizing the term-feature matrix, neural models are trained to predict the term from its features.
- The model learns dense low-dimensional representations in the process of minimizing the prediction error.

word2vec: Skip-gram

- The features for a term are made up of its neighbors within a fixed size window over the text.
- Skip-gram



S is the set of all windows over the training text and c is the number of neighbours we want to predict on either side of the term t_i.

$$\mathcal{L}_{skip-gram} = -\frac{1}{|S|}\sum_{i=1}^{|S|}\sum_{-c\leq j\leq +c, j\neq 0} \log(p(t_{i+j}|t_i))$$

$$\text{where,} \qquad p(t_{i+j}|t_i) = \frac{\exp\left((W_{out}\vec{v}_{t_{i+j}})^\intercal (W_{in}\vec{v}_{t_i})\right)}{\sum_{k=1}^{|T|}\exp\left((W_{out}\vec{v}_{t_k})^\intercal (W_{in}\vec{v}_{t_i})\right)}$$

W_{in} and W_{out} correspond to the embeddings the model learns for the input and the output terms, respectively

Minimizing the error in predicting a term given one of its neighbors

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{d} e^{z_j}} \quad 1 \leq i \leq d$$

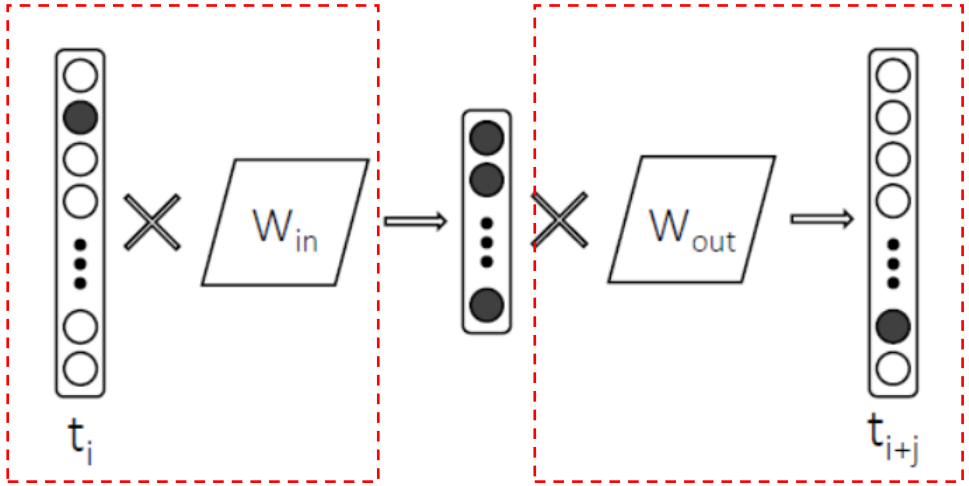
IN embedding and OUT embedding

- W_{in} and W_{out} are learnable parameters of the models.
- W_{in} gives us the IN embeddings corresponding to the input terms and W_{out} corresponds to the OUT embeddings for the output terms.
- Generally, only W_{in} is used and W_{out} is discarded after training.
- Both IN and OUT embeddings will be discussed in inter-term relationships.

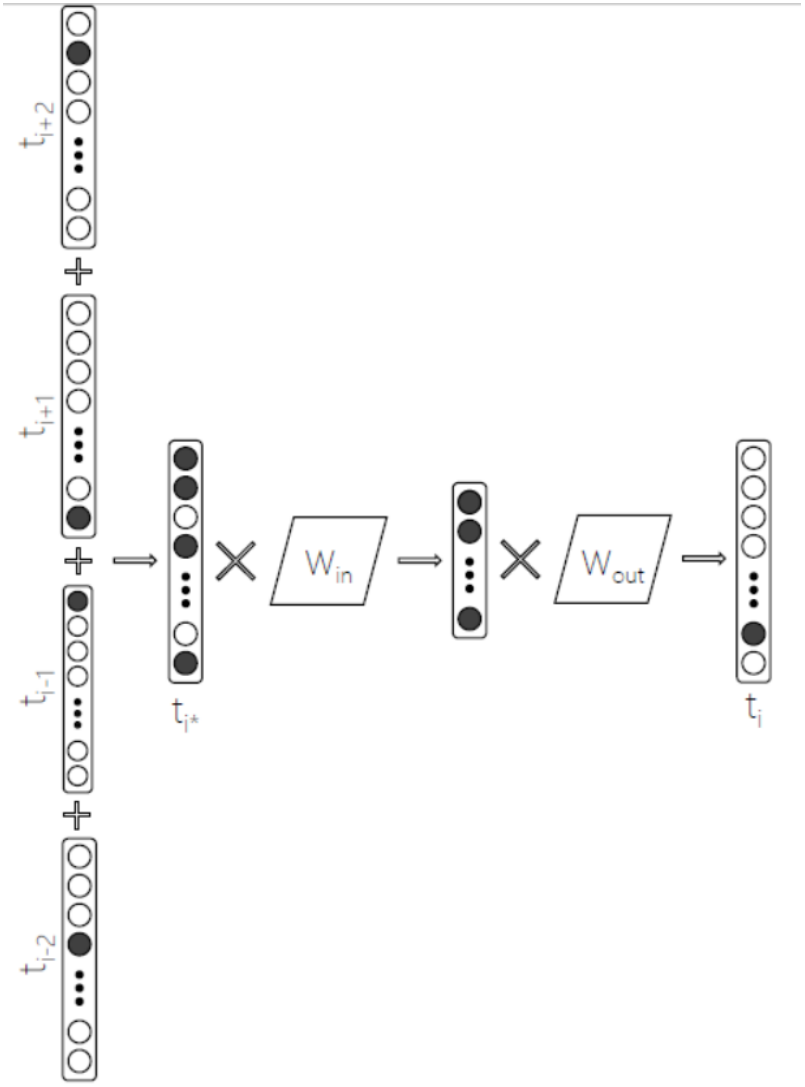
OUT embeddings IN embeddings

$$p(t_{i+j}|t_i) = \frac{\exp\left((W_{out}\vec{v}_{t_{i+j}})^\intercal (W_{in}\vec{v}_{t_i})\right)}{\sum_{k=1}^{|T|}\exp\left((W_{out}\vec{v}_{t_k})^\intercal (W_{in}\vec{v}_{t_i})\right)}$$

Degree of inter-term relationship



word2vec: CBOW (Continuous Bag of Word)



Predict a term from a bag of its neighboring terms

$$\mathcal{L}_{CBOW} = -\frac{1}{|S|}\sum_{i=1}^{|S|}\log(p(t_i|t_{i-c},\ldots,t_{i-1},t_{i+1},\ldots,t_{i+c}))$$

A Variant of Skip-Gram

- The skip-gram model trains on individual term-neighbor pairs.
- Aggregate all the training samples such that x_{ij} is the frequency of the pairs $<t_i,t_j>$ in the training data.
- Change loss function to

$$\begin{aligned}\mathcal{L}_{skip-gram} &= -\sum_{i=1}^{|T|}\sum_{j=1}^{|T|} x_{ij}\log(p(t_j|t_i)) \\ &= -\sum_{i=1}^{|T|} x_i \sum_{j=1}^{|T|} \frac{x_{ij}}{x_i}\log(p(t_j|t_i)) \\ &= -\sum_{i=1}^{|T|} x_i \sum_{j=1}^{|T|} \bar{p}(t_j|t_i)\log(p(t_j|t_i)) \\ &= \sum_{i=1}^{|T|} x_i H(\bar{p}(t_j|t_i), p(t_j|t_i))\end{aligned}$$

$H(\ldots)$ is the cross-entropy error between the actual co-occurrence probability $\bar{p}(t_j|t_i)$ and the one predicted by the model $p(t_j|t_i)$.

GloVe

- Replace the cross-entropy error with a squared-error and apply a saturation function *f*(…) over the actual co-occurrence frequencies.

$$\mathcal{L}_{GloVe} = -\sum_{i=1}^{|T|}\sum_{j=1}^{|T|} f(x_{ij})(log(x_{ij} - \vec{v}_{w_i}^{\intercal}\vec{v}_{w_j}))^2$$

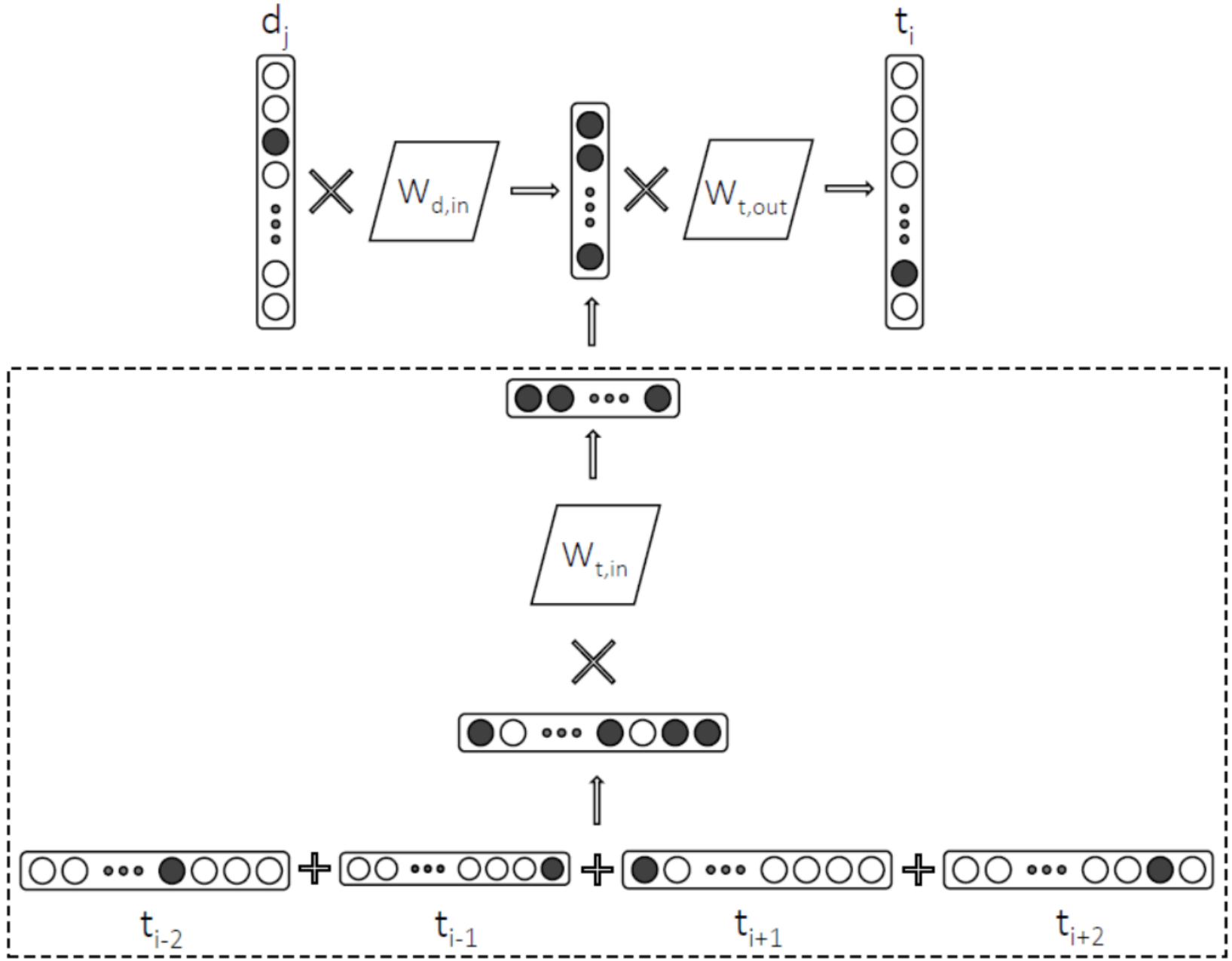
actual co-occurrence frequency

where,

$$f(x) = \begin{cases} (x/x_{max})^{\alpha}, & \text{if} x \leq x_{max} \\ 1, & \text{otherwise} \end{cases}$$

Paragraph2vec

- Train on term-document co-occurrences.
- Predict a term given the ID of a document or a passage that contains the term.
- In some variants, neighboring terms are also provided as input.
- Training on term-document pairs is to learn an embedding that is more aligned with a topical notion of term-term similarity, which is often more appropriate for IR tasks.


d_j
$W_{d,in}$
$W_{t,out}$
t_i
$W_{t,in}$
t_{i-2}
t_{i-1}
t_{i+1}
t_{i+2}

Term Embeddings for IR

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 Albuquerque metropolitan statistical area (or MSA) has a population of 907,301 according to the United States Census Bureau's most recently available estimate for 2015.

阿爾伯克基
位於新墨西哥州的城市

(a) About Albuquerque

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) Not about Albuquerque

a passage related to computer and technology

Inexact Matching

- Term embeddings can be useful for inexact matching
  - Deriving latent vector representations of the query and the document text for matching
  - As a mechanism for selecting additional terms for query expansion
- Query-document matching
  - Compare the query with the document directly in the embedding space
- Query expansion
  - Use embeddings to generate suitable query expansion candidates from a global vocabulary and then perform retrieval based on the expanded query

Query-Document Matching

- Deriving a dense vector representation for the query and the document from the embeddings of the individual terms in the corresponding texts.
- Aggregate the term embeddings
  - Average Word (or Term) Embeddings (AWE)
  - Non-linear combinations of term vectors
- Compare the query and the document embeddings
  - Cosine similarity
- Choice of term embeddings
  - LSA, word2vec, GloVe

Cosine Similarity

$$sim(q,d) = cos(\vec{v}_q, \vec{v}_d) = \frac{\vec{v}_q^{\intercal}\vec{v}_d}{\|\vec{v}_q\|\|\vec{v}_d\|}$$

$$\text{where,} \quad \vec{v}_q = \frac{1}{|q|}\sum_{t_q \in q}\frac{\vec{v}_{t_q}}{\|\vec{v}_{t_q}\|}$$

$$\vec{v}_d = \frac{1}{|d|}\sum_{t_d \in d}\frac{\vec{v}_{t_d}}{\|\vec{v}_{t_d}\|}$$

Choice of Term Embeddings for Retrieval

- Models, such as LSA and Paragraph2vec, that consider term-document pairs generally capture topical similarities in the learnt vector space.
- Word2vec and GloVe embeddings may incorporate a mixture of topical and typical notions of relatedness.
- The inter-term relationships modelled in the latent spaces of word2vec and GloVe may be closer to type-based similarities when trained with short window sizes or on short text, such as on keyword queries.

Revisit IN and OUT Embeddings

- The word2vec model learns two different embeddings, IN and OUT, corresponding to the input and the out terms.
- If a query contains a term t_i, then we may also consider the present of a different term t_j in the document to be a supporting evidence of relevance if the pair of terms < t_i, t_j> frequently co-occurs in the collection.
- In the skip-gram model, this probability of co-occurrence $p(t_j|t_i)$ is proportion to $(W_{out}\vec{v}_{t_j})^\intercal(W_{in}\vec{v}_{t_i})$, i.e., the dot product between the IN embeddings of t_i and the OUT embeddings of t_j.

OUT embeddings

IN embeddings

$$p(t_{i+j}|t_i) = \frac{\exp\left((W_{out}\vec{v}_{t_{i+j}})^\intercal(W_{in}\vec{v}_{t_i})\right)}{\sum_{k=1}^{|T|}\exp\left((W_{out}\vec{v}_{t_k})^\intercal(W_{in}\vec{v}_{t_i})\right)}$$

Degree of inter-term relationship

IN-OUT Similarity between the Query and the Document Terms

- Nalisnick et al. (2016) point out
  - When using word2vec embeddings for estimating the relevance of a document to a query, it is more appropriate to compute the IN-OUT similarity between the query and the document terms.
- The query terms should be represented using the IN embeddings and the document terms using the OUT embeddings.
- The difference between IN-IN or IN-OUT similarities between terms

Table 4.1: Different nearest neighbours in the word2vec embedding space based on whether we compute IN-IN, OUT-OUT, or IN-OUT similarities between the terms. The examples are from (Nalisnick *et al.*, 2016; Mitra *et al.*, 2016a) where the word2vec embeddings are trained on search queries. Training on short query text, however, makes the inter-term similarity more pronouncedly typical (where, "Yale" is closer to "Harvard" and "NYU") when both terms are represented using their IN vectors. In contrast, the IN-OUT similarity (where, "Yale" is closer to "faculty" and "alumni") mirrors more the topical notions of relatedness.

西雅圖海鷹 (西雅圖的職業美式足球球隊)

yale			seahawks			
IN-IN	OUT-OUT	IN-OUT	IN-IN	OUT-OUT	IN-OUT	
yale	yale	yale	seahawks	seahawks	seahawks	
harvard	uconn	faculty	49ers	broncos	highlights	美式足球釘鞋
nyu	harvard	alumni	broncos	49ers	jerseys	美式足球球衣
cornell	tulane	orientation	packers	nfl	tshirts	
tulane	nyu	haven	nfl	packers	seattle	
tufts	tufts	graduate	steelers	steelers	hats	

舊金山49人(美國國家美式足球聯盟球隊)
丹佛野馬(丹佛的職業美式足球隊)
綠灣包裝工(職業美式足球隊)
國家美式足球聯盟
匹茲堡鋼人(職業美式足球隊)

Dual Embedding Space Model (DESM)

- Estimate the query-document relevance as follows.

$$DESM_{in-out}(q,d) = \frac{1}{|q|}\sum_{t_q \in q} \frac{\vec{v}_{t_q,in}^{\intercal}\vec{v}_{d,out}}{\|\vec{v}_{t_q,in}\|\|\vec{v}_{d,out}\|}$$

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

Neural Translation Language Model (NTLM)

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

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

$$p(t_q|d) = \sum_{t_d \in d} p(t_q|t_d) \cdot p(t_d|d)$$

Generalized Language Model (GLM)

$$p(d|q) = \prod_{t_q \in q} \Bigg(\lambda \frac{tf(t_q, d)}{|d|} + \alpha \frac{\sum_{t_d \in d} (sim(\vec{v}_{t_q}, \vec{v}_{t_d}) \cdot tf(t_d, d))}{\sum_{t_{d_1} \in d} \sum_{t_{d_2} \in d} sim(\vec{v}_{t_{d_1}}, \vec{v}_{t_{d_2}}) \cdot |d|^2} + \beta \frac{\sum_{\bar{t} \in N_t} (sim(\vec{v}_{t_q}, \vec{v}_{\bar{t}}) \cdot \sum_{\bar{d} \in D} tf(\bar{t}, \bar{d}))}{\sum_{t_{d_1} \in N_t} \sum_{t_{d_2} \in N_t} sim(\vec{v}_{t_{d_1}}, \vec{v}_{t_{d_2}}) \cdot \sum_{\bar{d} \in D} |\bar{d}| \cdot |N_t|} + (1 - \alpha - \beta - \lambda) \frac{\sum_{\bar{d} \in D} tf(t_q, \bar{d})}{\sum_{\bar{d} \in D} |\bar{d}|} \Bigg)$$

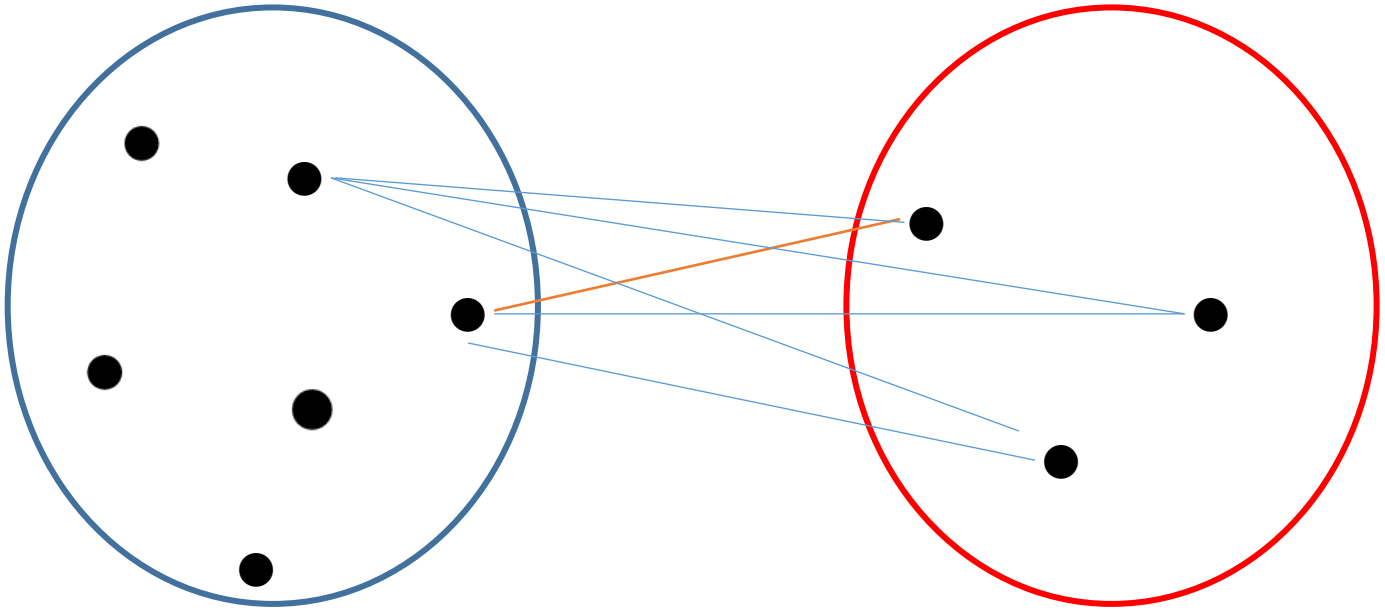
where, N_t is the set of nearest-neighbours of term t.

Dirichlet Prior Smoothing (MacKay and Peto, 1995)

$$p(t_q|d) = \left(tf(t_q, d) + \mu \frac{\sum_{\bar{d} \in D} tf(t_q, \bar{d})}{\sum_{\bar{d} \in D} |\bar{d}|} \right) / \left(|d| + \mu \right)$$

Word Mover's Distance (WMD)

- Measure the similarity between pairs of documents A and B
- Computing the minimum distance in the embedding space that each term in A needs to travel to reach the terms in B.



Non-linear Word Transportation (NWT)

Estimate the relevance between a query and a document: Term-alignment based distance metric

$$\max \sum_{t_q \in q} \log \left(\sum_{t_d \in u(d)} f(t_q, t_d) \cdot max(cos(\vec{v}_{t_q}, \vec{v}_{t_d}), 0)^{idf(t_q)+b} \right)$$

$u(d)$: set of unique terms in document d

$$\text{subject to} \quad f(t_q, t_d) \geq 0, \quad \forall t_q \in q, t_d \in d$$

$$\text{and} \quad \sum_{t_q \in q} f(t_q, t_d) = \frac{tf(t_d) + \mu \frac{\sum_{\bar{d} \in D} tf(t_q, \bar{d})}{\sum_{\bar{d} \in D} |\bar{d}|}}{|d| + \mu}, \quad \forall t_d \in d$$

$$\text{where,} \quad idf(t) = \frac{|D| - df(t) + 0.5}{df(t) + 0.5}$$

Dirichlet Prior Smoothing (MacKay and Peto, 1995)

$$p(t_q|d) = \left(tf(t_q, d) + \mu \frac{\sum_{\bar{d} \in D} tf(t_q, \bar{d})}{\sum_{\bar{d} \in D} |\bar{d}|} \right) / \left(|d| + \mu \right)$$

Saliency-Weighted Semantic Network (SWSN)

Compute the short-text similarity: Term-alignment based distance metric

$$swsn(s_l, s_s) = \sum_{t_l \in s_l} idf(t_l) \cdot \frac{sem(t_l, s_s) \cdot (k_1 + 1)}{sem(t_l, s_s) + k_1 \cdot \left(1 - b + b \cdot \frac{|s_s|}{avgsl}\right)}$$

s_l: longer of the two sentences (q)
s_s: shorter of the two sentences (d)

$$\text{where,} \quad sem(t, s) = \max_{\bar{t} \in s} cos(\vec{v}_t, \vec{v}_{\bar{t}})$$

Here s_s is the shorter of the two sentences to be compared, and s_l the longer sentence.

$$BM25(q, d) = \sum_{t_q \in q} idf(t_q) \cdot \frac{tf(t_q, d) \cdot (k_1 + 1)}{tf(t_q, d) + k_1 \cdot \left(1 - b + b \cdot \frac{|d|}{avgdl}\right)}$$

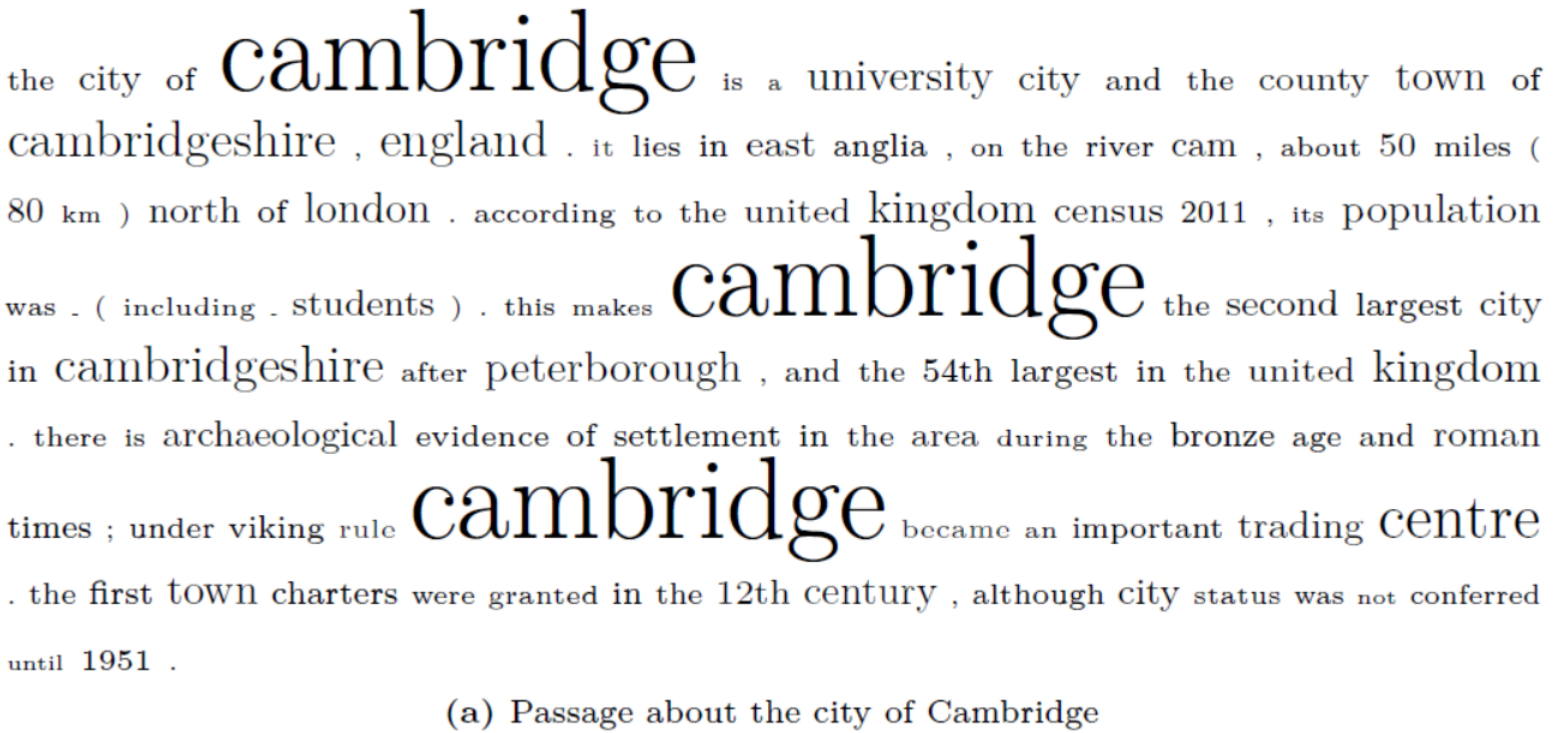
Embedding-Based Model vs. Exact Matching Model

Matching using
local (exact matching) and
distributed representations
(embedding space based matching)
of terms for retrieval

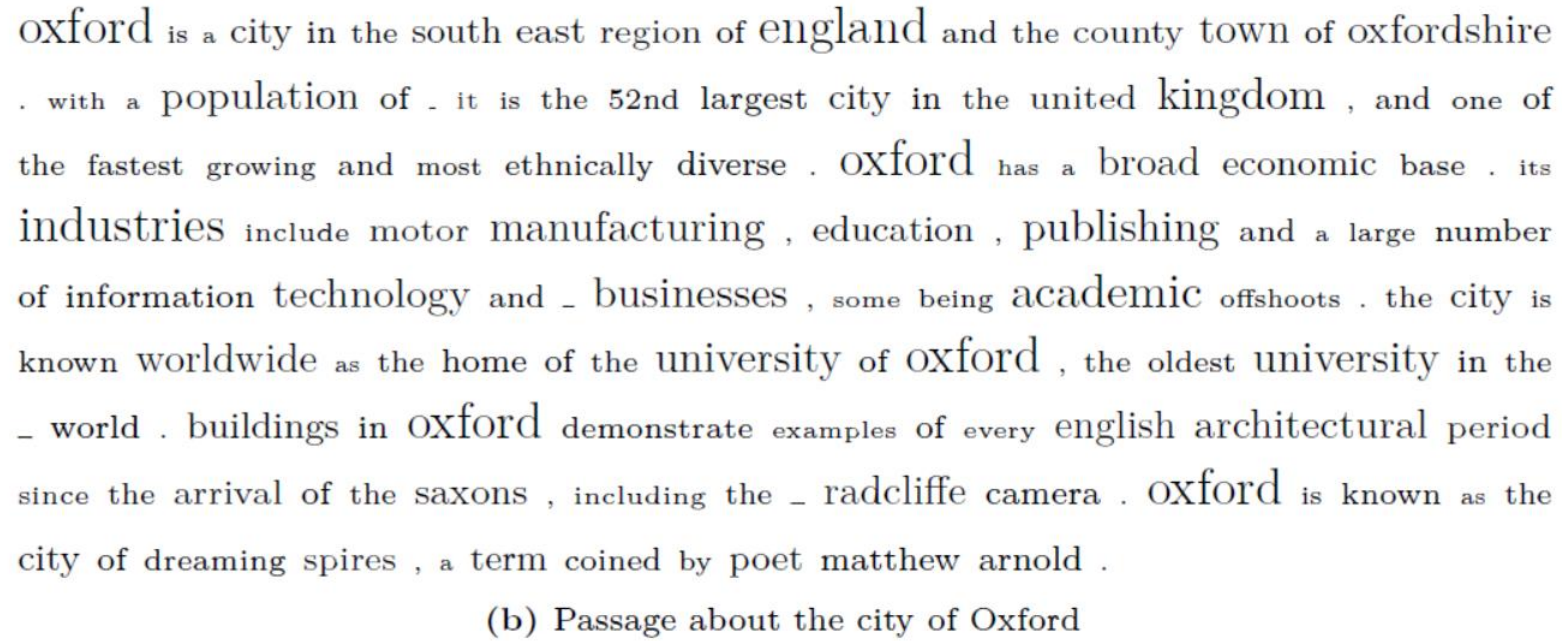
A visualization of IN-OUT similarities between terms in different passages with the query term “Cambridge”

An embedding based approach may be able to determine that passage (c) is non-relevant to the query “Cambridge”, but fail to realize that passage (b) is also non-relevant.

A term counting-based model can identify that passage (b) is non-relevant but may rank (c) incorrectly high.



(a) Passage about the city of Cambridge



(b) Passage about the city of Oxford

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 .

(c) Passage about giraffes, but 'giraffe' is replaced by 'Cambridge'

Shortcoming of Embedding based Models

- Perform poorly when the retrieval is performed over the full document collection.
- The errors made by embedding based models and exact matching models may be different.
- The combination of the two is often preferred.
- Rerank only a subset of the documents retrieved by a different – generally an exact matching based – IR model.

Telescoping (套疊；相嵌)

- Use the embedding based model to re-rank only a subset of the documents retrieved by a different – generally an exact matching-based IR model.
- The chaining of different IR models where each successive model re-ranks a smaller number of candidate documents.
- Telescoping evaluations are common in the neural IR literature.
- Good performances on re-ranking tasks may not be indicative how the model performs if the retrieval involves larger document collections.

Query Expansion

- Use term embeddings to find good expansion candidates from a global vocabulary, and then retrieve documents using the expanded query.
- Estimate the relevance of candidate terms to the query
  - Compare the candidate term individually to every query term using their vector representations
  - Aggregate the scores

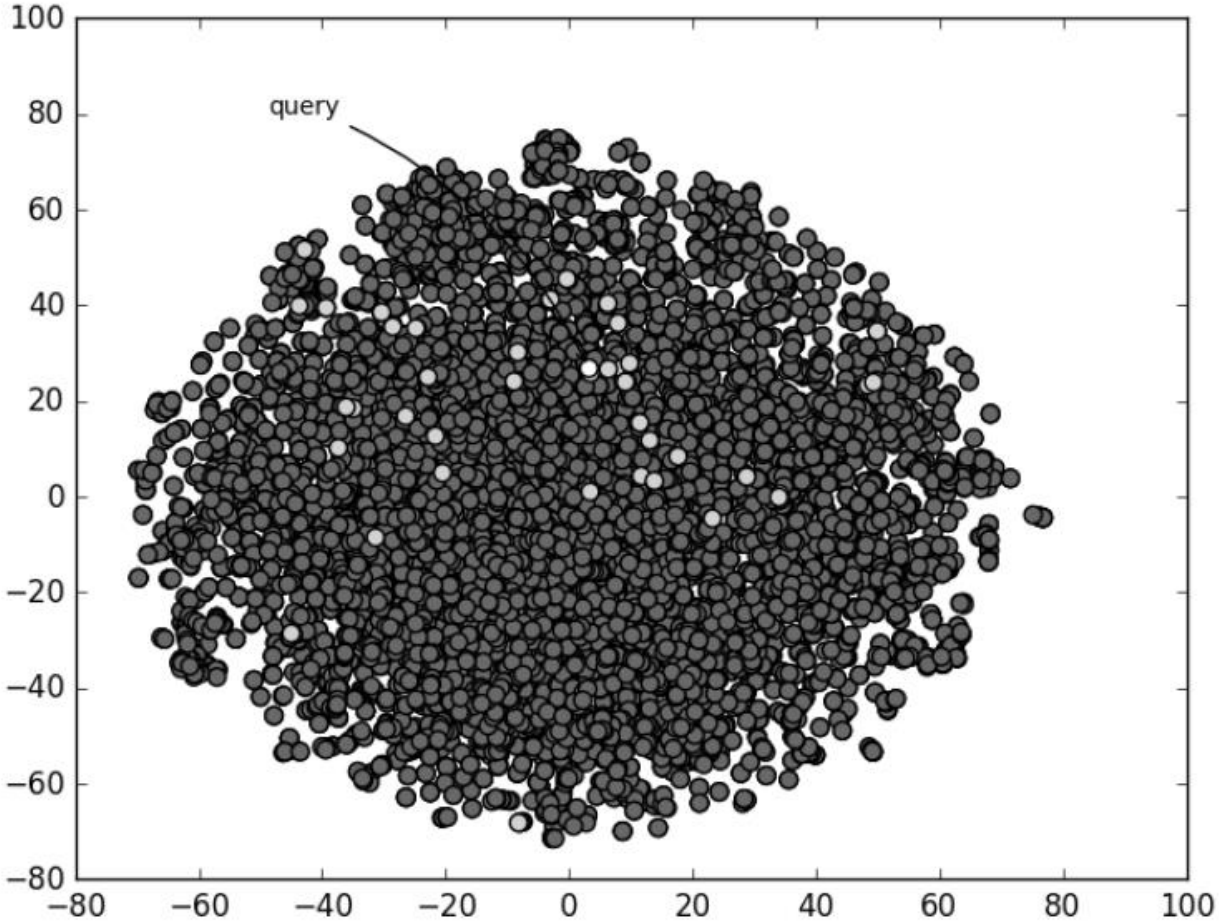
$$score(t_c, q) = \frac{1}{|q|} \sum_{t_q \in q} cos(\vec{v}_{t_c}, \vec{v}_{t_q})$$

Findings

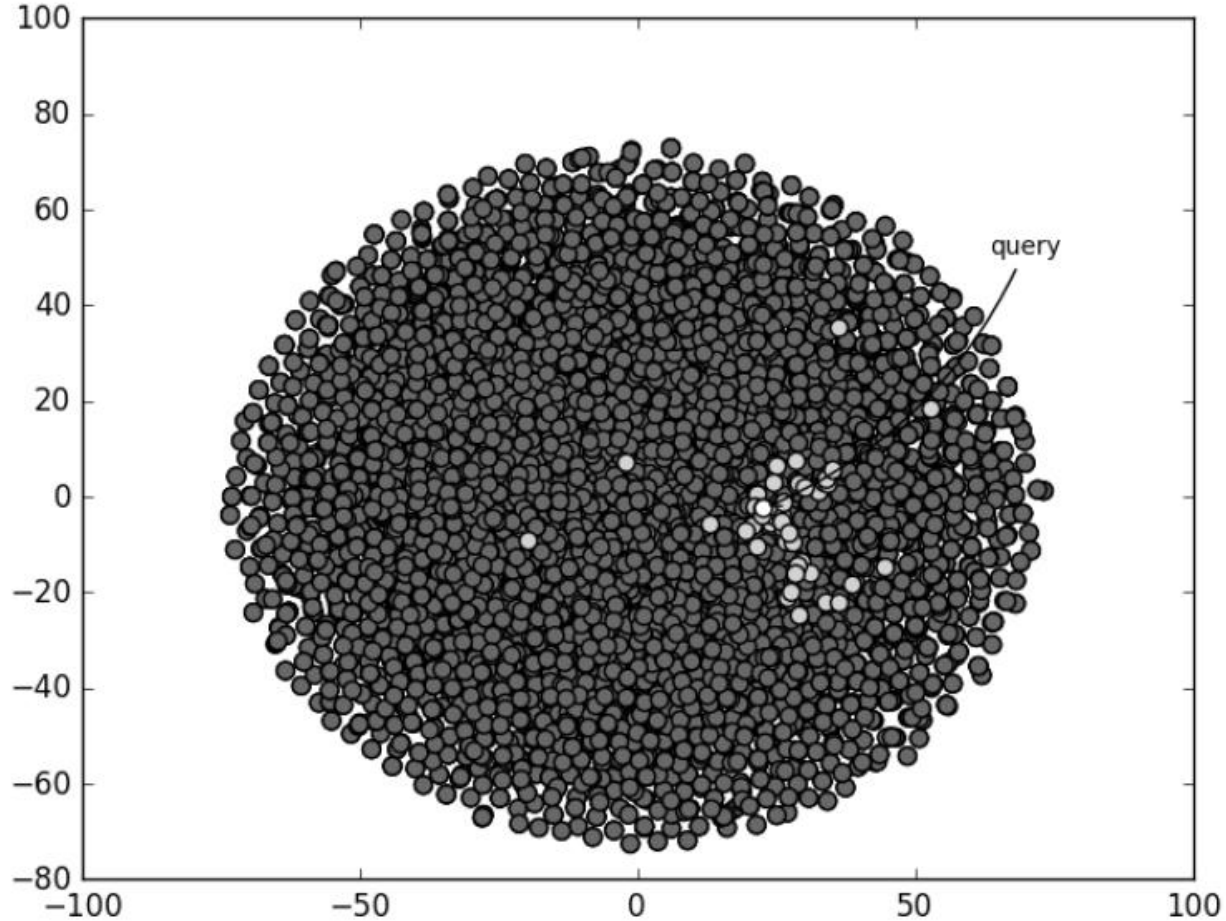
- Term embedding based query expansion on its own performs worse than pseudo-relevance feedback.
- Show better performances when used in combination with PRF.
- Term embeddings have been explored for re-weighting query terms and finding relevant query re-writes, as well as in the context of other IR tasks such as cross-lingual retrieval and entity retrieval.

Query-Specific Term Embeddings

- Incorporate relevance feedback in the process of learning the term embeddings.
  - A set of documents is retrieved for the query.
  - A query-specific term embedding model is trained.
  - The local embedding model is then employed for identifying expansion candidates for the query for a second round of document retrieval.
- The query–specific term embeddings are more effective in identifying good expansion terms than a global representation.



(a) Global embedding



(b) Local embedding

Figure 4.3: A two-dimensional visualization of term embeddings when the vector space is trained on a (a) global corpus and a (b) query-specific corpus, respectively. The grey circles represent individual terms in the vocabulary. The white circle represents the query "ocean remote sensing" as the centroid of the embeddings of the individual query terms, and the light grey circles correspond to good expansion terms for this query. When the representations are query-specific then the meaning of the terms are better disambiguated, and more likely to result in the selection of good expansion terms.

Supervised Learning to Rank

Learning to Rank (LTR)

- Use training data $rel_q(d)$, such as human relevance labels and click data, to train towards an IR objective.
  - LTR models represent a rankable item, e.g., a query-document pair, as a feature vector $\vec{x} \in \mathbb{R}^n$.
  - The ranking model $f : \vec{x} \rightarrow \mathbb{R}$ is trained to map the vector to a real-valued score such that for a given query more relevant documents are scored higher and some chosen rank-based metric is maximized.
- LTR approaches
  - pointwise approach
  - pairwise approach
  - listwise approach

Pointwise Approach

- The relevance information $rel_q(d)$ is a numerical value associated with every query-document pair with input vector $\vec{x}_{q,d}$.
- The numerical relevance label can be derived from binary or graded relevance judgements or from implicit user feedback, such as a clickthrough rate.
- A regression model is typically trained on the data to predict the numerical value $rel_q(d)$ given $\vec{x}_{q,d}$.

Pairwise Approach

- The relevance information is in the form of preferences between pairs of documents with respect to individual queries $d_i \succ_q d_j$.
- The ranking problem is a binary classification to predict the more relevant document.

Listwise Approach

- Optimizing for a rank-based metric such as NDCG.

In learning to rank, direct supervision is used to optimize the model for a ranking task.

Input Features

- Traditional LTR models
  - Hand-crafted features for representing query-document pairs in $\vec{x}$.
    - Query-independent or static features: e.g., incoming link count and document length
    - Query-dependent or dynamic features: e.g., BM25
    - Query-level features: e.g., query length
- Neural LTR model
  - The deep architecture is responsible for feature learning from simple vector representations of the input.
  - The features learnt from the query and document text can be combined with other features that may not be possible to infer from the context, such as document popularity.

Loss Function

- In ad-hoc retrieval, the LTR model needs to rank the documents in a collection D in response to a query.
- In the pointwise approach, the neural model is trained to directly estimate $rel_q(d)$, which can be a numeric value or a categorical label.

Regression Loss

- Given $\vec{x}_{q,d}$, estimating the relevance label $rel_q(d)$ can be cast as a regression problem.
- A standard loss function *square loss*

$$\mathcal{L}_{\text{squared}} = \|rel_q(d) - s(\vec{x}_{q,d})\|^2$$

where, $s(\vec{x}_{q,d})$ is the score predicted by the model and $rel_q(d)$ can either be the value of the relevance label (Cossock and Zhang, 2006) or the one-hot representation when the label is categorical (Fuhr, 1989).

Classification Loss

- When the relevance labels in the training data are categorical, the label prediction problem can be treated as a multiclass classification.
- The neural model estimates the probability of a label *y* given $\vec{x}_{q,d}$.
- The probability of the correct label $y_{q,d}\ (= rel_q(d))$ can be obtained by the softmax function.

$$p(y_{q,d}|q,d) = p(y_{q,d}|\vec{x}_{q,d}) = \frac{e^{\gamma \cdot s\left(\vec{x}_{q,d}\,,\, y_{q,d}\right)}}{\sum_{y \in Y} e^{\gamma \cdot s(\vec{x}_{q,d}\,,\, y)}}$$

- cross-entropy loss

$$\mathcal{L}_{\text{classification}} = -\log\Big(p(y_{q,d}|q,d)\Big) = -\log\Big(\frac{e^{\gamma \cdot s\left(\vec{x}_{q,d}\,,\, y_{q,d}\right)}}{\sum_{y \in Y} e^{\gamma \cdot s(\vec{x}_{q,d}\,,\, y)}}\Big)$$

Contrastive Loss

- In representation learning models, a relevant document should be closer to the query representation than a non-relevant document.
- The contrastive loss learns the model parameters by minimizing the distance between a relevant pair, while increasing the distance between dissimilar items.

$$\mathcal{L}_{\text{Contrastive}}(q, d, y_{q,d}) = y_{q,d} \cdot \mathcal{L}_{pos}(\text{dist}_{q,d}) + (1 - y_{q,d}) \cdot \mathcal{L}_{neg}(\text{dist}_{q,d})$$

- Contrastive loss assumes that the relevance label $y_{q,d} \in \{0, 1\}$ is binary.

$$\mathcal{L}_{\text{Contrastive}}(q, d, y_{q,d}) = y_{q,d} \cdot \frac{1}{2}(\max(0, m - \text{dist}_{q,d}))^2 + (1 - y_{q,d}) \cdot (\text{dist}_{q,d})^2$$

m: a margin

Rank the relevant documents D^+

- A ranking model does not need to estimate the true relevance label accurately as long as it ranks the relevant documents D^+ over all the other candidates in D.
- Only a few documents from D are relevant to q.
- If a binary notion of relevance is assumed, then the problem is similar to multi-label classification, where the candidate documents are the classes.
- We will discuss loss functions for LTR models that tries to predict the relevant document by maximizing $p(d^+|q)$, where d^+ is a relevant document, rather than maximizing $p(y_{q,d}|q,d)$.

Cross-Entropy Loss over Documents

- The probability of ranking d^+ over all the other documents in the collection is given by the softmax function

$$p(d^+|q) = \frac{e^{\gamma \cdot s\left(q,d^+\right)}}{\sum_{d \in D} e^{\gamma \cdot s(q,d)}}$$

- The cross-entropy loss maximizes the difference between scores generated by the model for relevant and less relevant documents

$$\mathcal{L}_{\mathrm{CE}}(q, d^+, D) = -\log\left(p(d^+|q)\right)$$

$$= -\log\left(\frac{e^{\gamma \cdot s\left(q,d^+\right)}}{\sum_{d \in D} e^{\gamma \cdot s(q,d)}}\right)$$

Hierarchical Softmax

- Instead of computing $p(d^+|q)$ directly, group the candidates D into a set of classes C, and then the correct class c^+ given q followed by predicting d^+ given $< c^+, q >$.

$$p(d^+|q) = p(d^+|c^+, q) \cdot p(c^+|q)$$

- The computational cost is a function of $|C|+|c^+|$ which is much smaller than $|D|$.
- The hierarchy of classes is based on either similarity between candidates or frequency binning.

Computing the cross-entropy loss with a softmax is costly because the softmax normalization involves scoring every candidate in the collection using the trained model. Alternatives include applying hierarchical categories over candidates or approximating the loss function by considering only a sample of candidates from the collection.

Pairwise Approach

- Multiple documents may be relevant to the same query q, the notion of relevance among this set of documents D^+ may be further graded.
- Consider pairs of documents for the same query and minimize the average number of inversions in ranking, i.e., $d_i \succ_q d_j$ but d_j is ranked higher than d_i.
- Pairwise loss

$$\mathcal{L}_{pairwise} = \phi(s_i - s_j)$$

where, some possible choices for ϕ include,

- Hinge function $\phi(z) = \max(0, 1 - z)$
- Exponential function $\phi(z) = e^{-z}$
- Logistic function $\phi(z) = \log\,(1 + e^{-z})$

RankNet Loss

- The model is trained on triples $\langle q, d_i, d_j \rangle$ consisting of a query q and a pair of documents d_i and d_j with different relevance labels such that d_i is more relevant than d_j ($d_i \succ_q d_j$) and the corresponding feature vectors $\langle \vec{x}_i, \vec{x}_j \rangle$.
- The model $f : \mathbb{R}^n \to \hat{\mathbb{R}}$, a neural network whose output is differentiable with respect its parameters, computes the score $s_i = f(\vec{x}_i)$ and $s_j = f(\vec{x}_j)$, where $s_i > s_j$.
- Given the scores $\langle s_i, s_j \rangle$, the probability that d_i would be ranked higher than d_j is given by

$$p_{ij} \equiv p(s_i > s_j) \equiv \frac{1}{1 + e^{-\sigma(s_i - s_j)}}$$

where, the constant σ determines the shape of the sigmoid.

Let $S_{ij} \in \{-1, 0, +1\}$ be the true preference label between d_i and d_j for the training sample— denoting d_i is less, equal, or more relevant than d_j, respectively. Then the desired probability of ranking d_i over d_j is given by $\bar{p}_{ij} = \frac{1}{2}(1 + S_{ij})$. The cross-entropy loss $\mathcal{L}$ between the desired probability $\bar{p}_{ij}$ and the predicted probability p_{ij} is given by,

$$\begin{aligned}
\mathcal{L} &= -\bar{p}_{ij} \log(p_{ij}) - (1 - \bar{p}_{ij}) log(1 - p_{ij}) \\
&= \frac{1}{2}(1 - S_{ij})\sigma(s_i - s_j) + \log(1 + e^{-\sigma(s_i - s_j)}) \\
&= \log(1 + e^{-\sigma(s_i - s_j)}) \quad \text{if, } d_i \succ_q d_j (S_{ij} = 1)
\end{aligned}$$

Limitation of Pairwise Objective Function

- Rank inversion of any pair of documents is considered equally harmful.
- For most IR metrics where a significantly large penalty is associated with inversions at the top rank positions.
  - Result list A ranks two relevant documents at positions one and 50.
  - Result list B ranks the same two relevant documents at positions three and 40.
  - Result set A has more rank inversions compared to result set B.
  - On typical IR metrics, such as NDCG, it would be fare better.
- The rank-based metrics are generally non-continuous and non-differentiable.

LambdaRank Loss

- The gradient should be bigger for pairs of documents that produce a bigger impact in NDCG by swapping positions.
- To train a model, we do not need the costs themselves, only gradients.
- LambdaRank loss
  - Weight the gradients from the RankNet loss by the NDCG delta

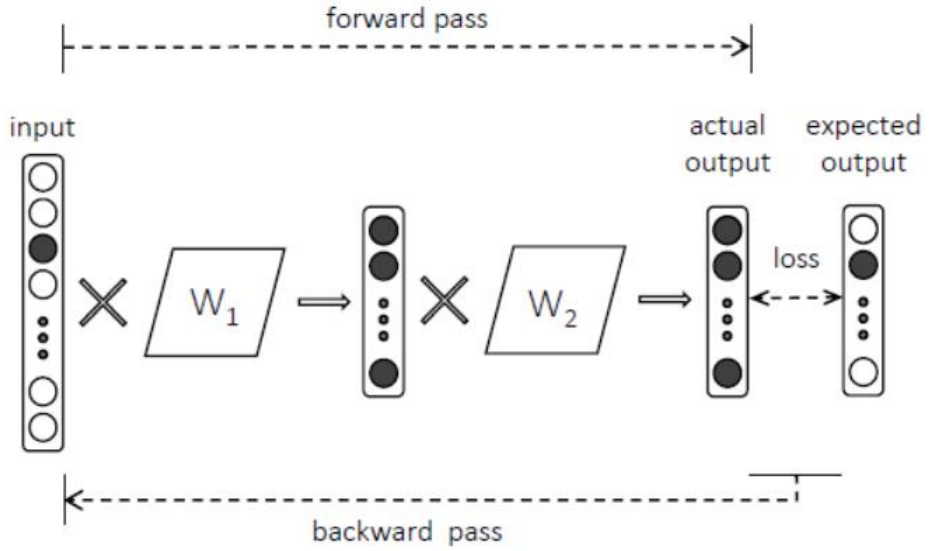
$$\lambda_{LambdaRank} = \lambda_{RankNet} \cdot |\Delta NDCG|$$

Deep Neural Network

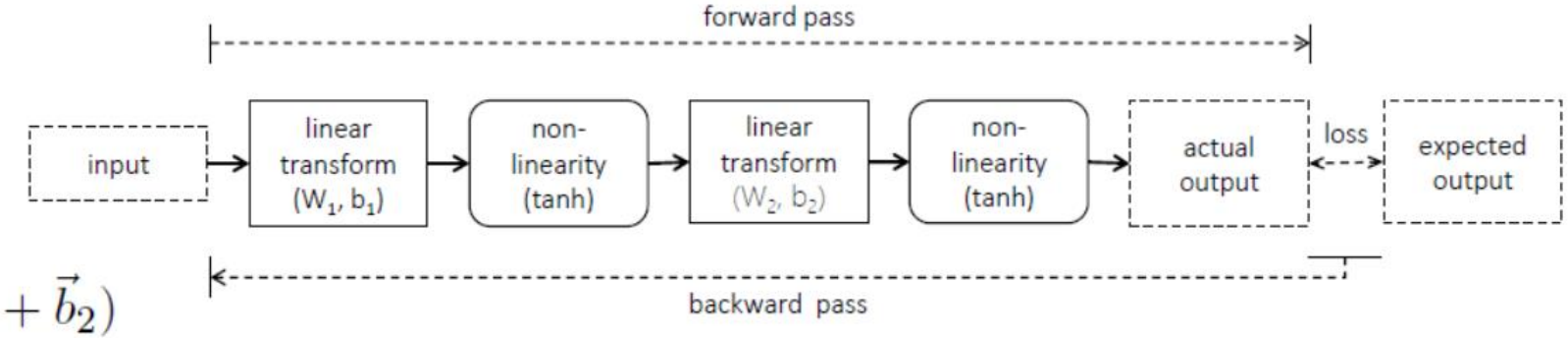
Fundamentals

- Deep neural network models consist of chains of tensor operations.
- Tensor operation
  - Parameterized linear transformations (e.g., multiplication with a weight matrix, or the addition of a bias vector)
  - Elementwise application of non-linear functions, such as *tanh* or *rectified linear units* (ReLU).

Feed-Forward Neural Network with Fully-Connected Layers



(a) A neural network with a single hidden layer



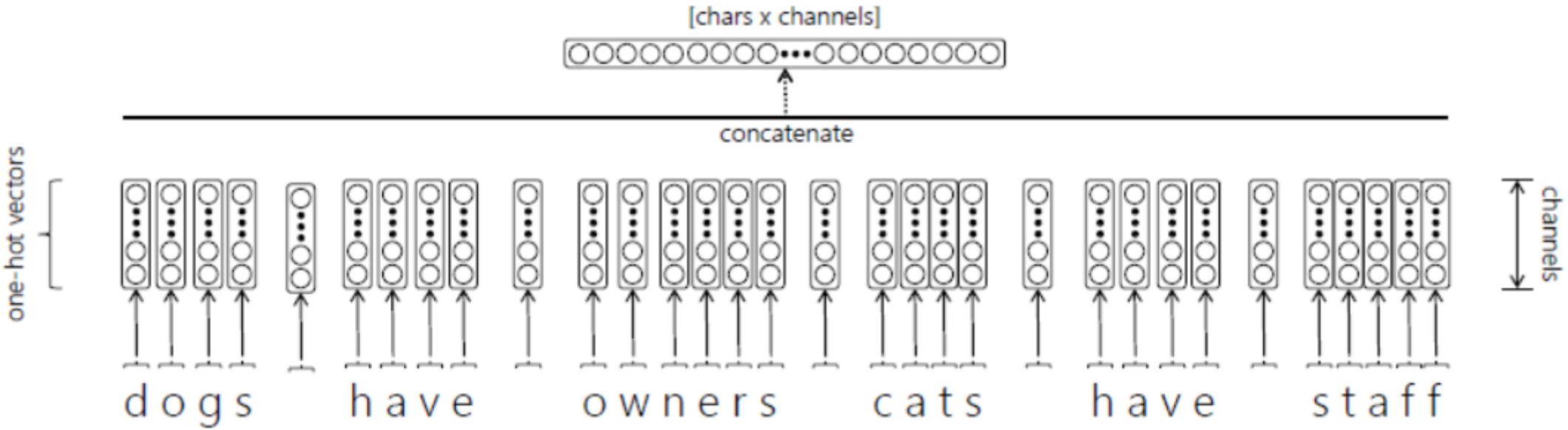
$$\vec{y} = tanh(W_2 \cdot tanh(W_1 \cdot \vec{x} + \vec{b}_1) + \vec{b}_2)$$

(b) The same neural network viewed as a chain of computational steps

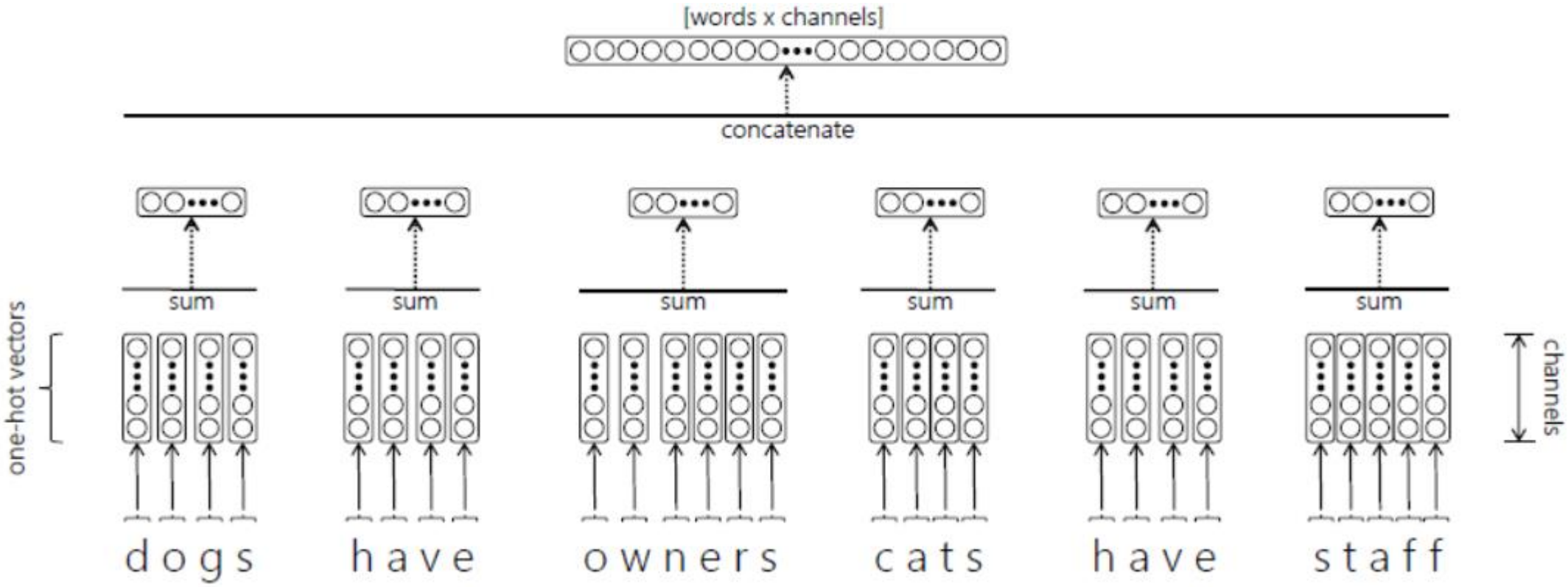
Input Text Representation

- Character-level input
  - Each character is represented by a one-hot vector.
  - The vector dimensions (channels) equals to the number of allowed character in the vocabulary.
  - Incorporate the least amount of prior knowledge about the language in the input representation.
  - The representation of longer texts can be derived by concatenating or summing the character-level vectors.
- Term-level input
  - Term-level input w/ bag-of-characters per term
  - Term-level input w/ bag-of-trigraphs per term
  - Term-level input w/ pre-trained term embeddings
  - The term vectors are further aggregated to obtain the representation of longer chunks.

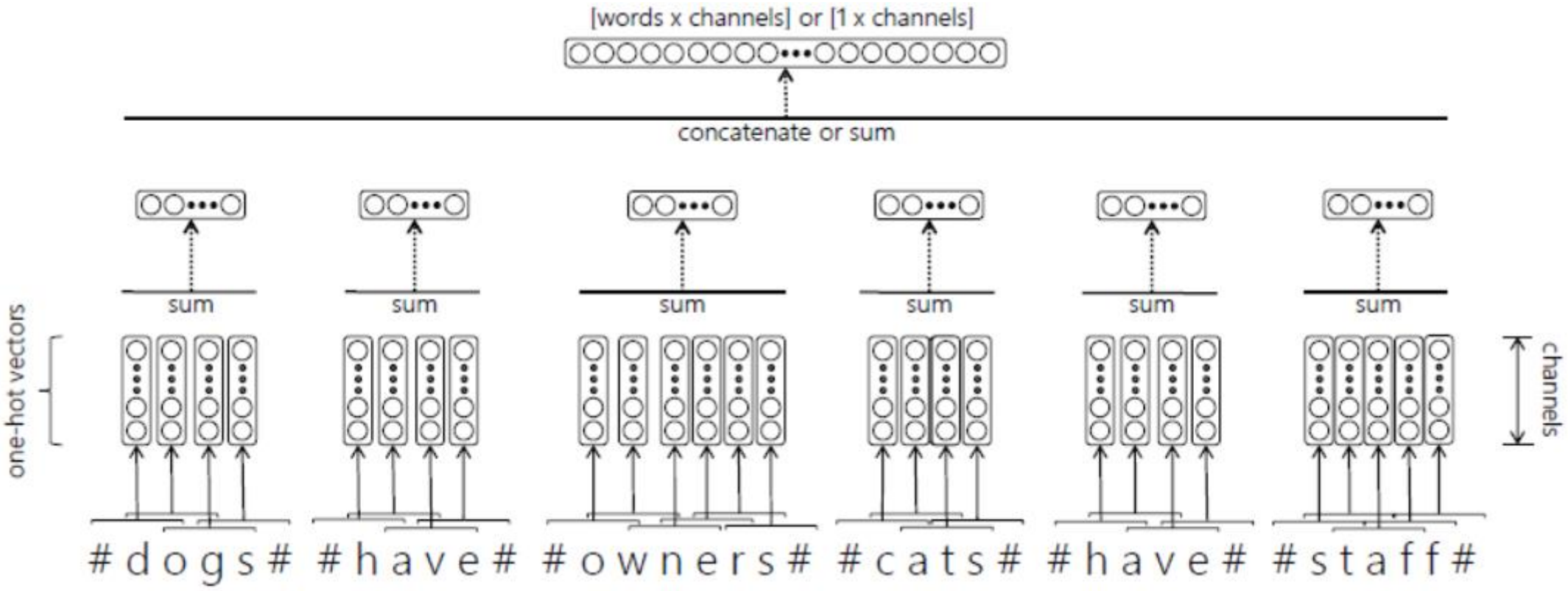
Character-Level Input



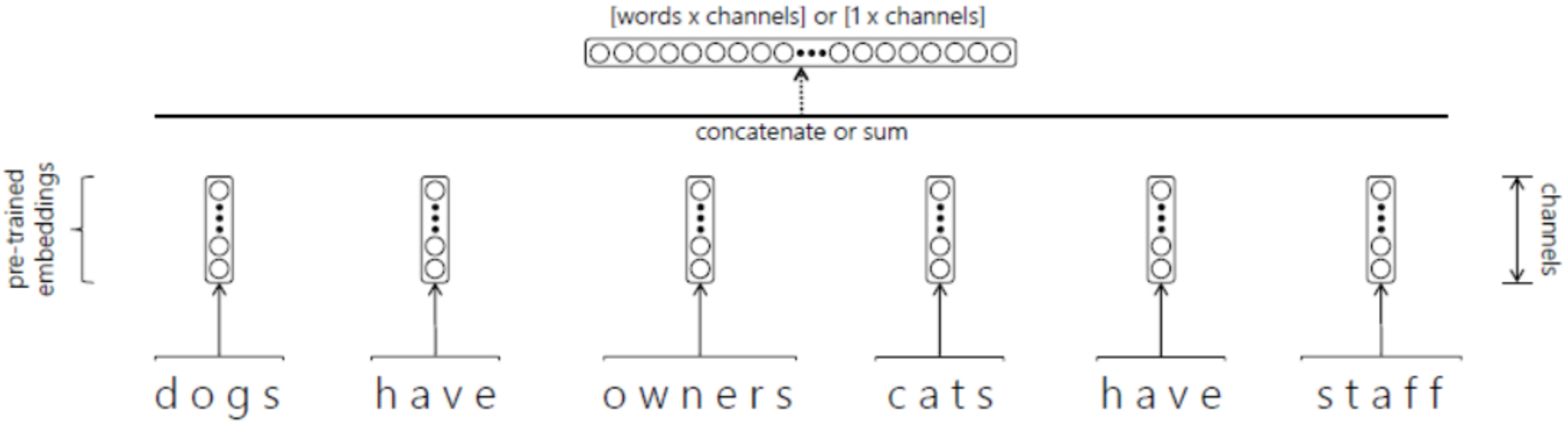
Term-level input w/ bag-of-characters per term



Term-level input w/ bag-of-trigraphs per term



Term-level input w/ pre-trained term embeddings



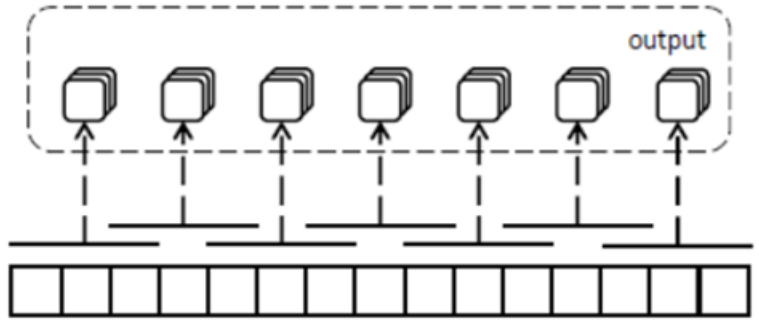
Standard Architectures

- Shift-invariant neural operations
  - Convolutional and recurrent architectures
  - The architectures stem from the natural regularities observable in most inputs.
  - The task of detecting a face should be invariant to whether the image is shifted, rotated, or scaled.
  - The meaning of an English sentence should, in most cases, stay consistent independent of which part of the document it appears in.
  - A fixed size window moves over the input space with fixed stride in each step.
  - A kernel, or a filter, or a cell is applied over each instance of the window.
  - The parameters of the cell are shared across all the instances of the input window.

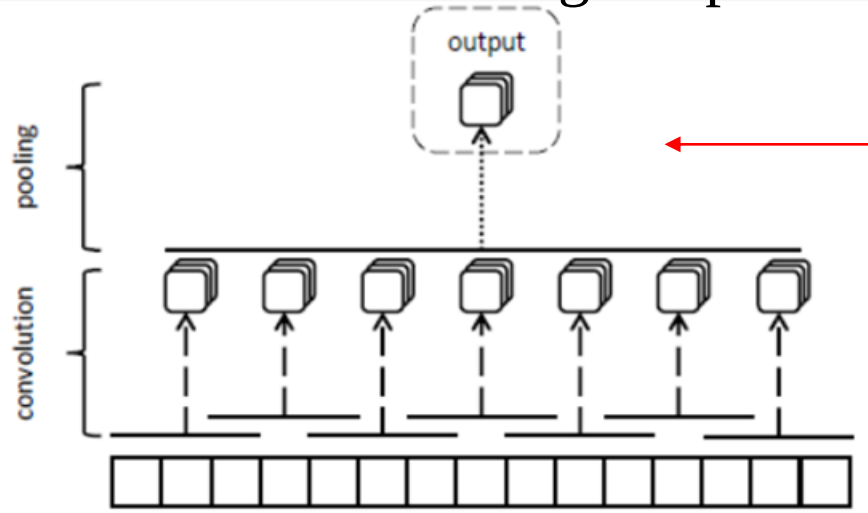
All shift-invariant neural operations, including convolutional and recurrent layers, move a fixed size window over the input space with fixed stride. Each window is projected by a parameterized neural operation, or *cell*, often followed by an aggregation step, such as pooling.

Convolution or Pooling

- Convolution, a cell, is applied on a sequence of terms with a window size of (three) terms in each step.
- Pooling, a cell without any parameters, consists of aggregating (max or average per channel) over all the terms in the window.
- The length of the output of a convolutional (or pooling) layer is a function of the input length
- Global pooling, where the window spans over the whole input, is applied on the top.
- Global pooling generates a fixed size of output from a variable length input.



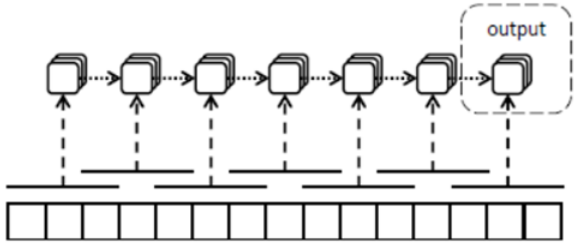
(a) Convolution or pooling



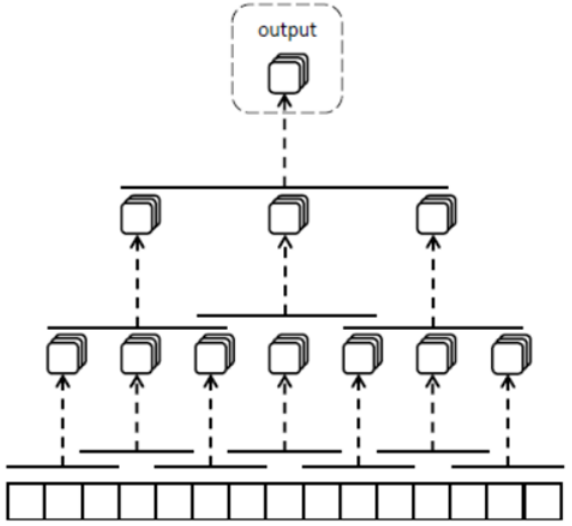
(b) Convolution w/ global pooling

Recurrent Architecture

- In convolution or pooling, each window is applied independently.
- The cell in the recurrent architecture considers both the input window and the output of previous instance.
- Many different cell architectures such as Elman network, LSTM, and GRU have been explored.
- In tree-structured (or recursive) NN,
  the same cell is applied at multiple levels
  in a tree-like hierarchical fashion.



(c) Recurrent



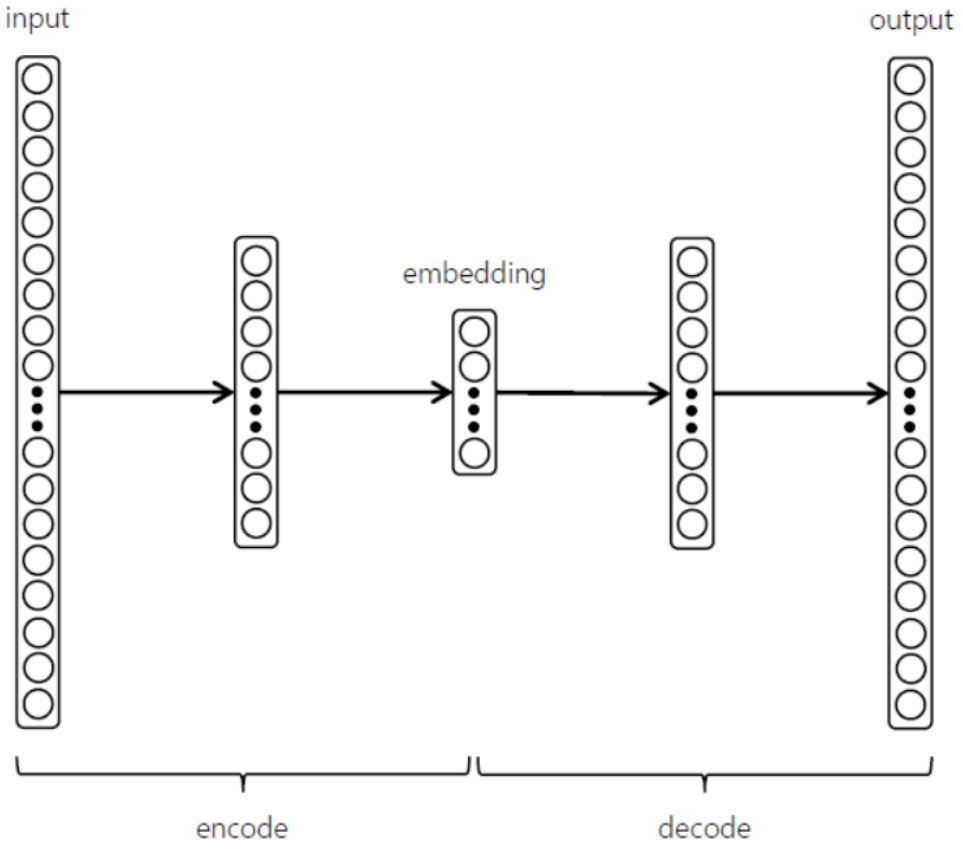
(d) Recursive or tree

How the window outputs are aggregated?

- Convolutional layers are typically followed by pooling or fully-connected layers that perform global aggregation over all the window instances.
  - Position issue
    - A fully-connected layer is aware of each window position.
    - A global pooling layer is typical agnostic to it.
  - Length issue
    - Unlike a fully-connected layer, a global max-pooling operation can applied to a variable size input.
  - Long sequence
    - RNN with memory and/or attention may be useful.

Autoencoder

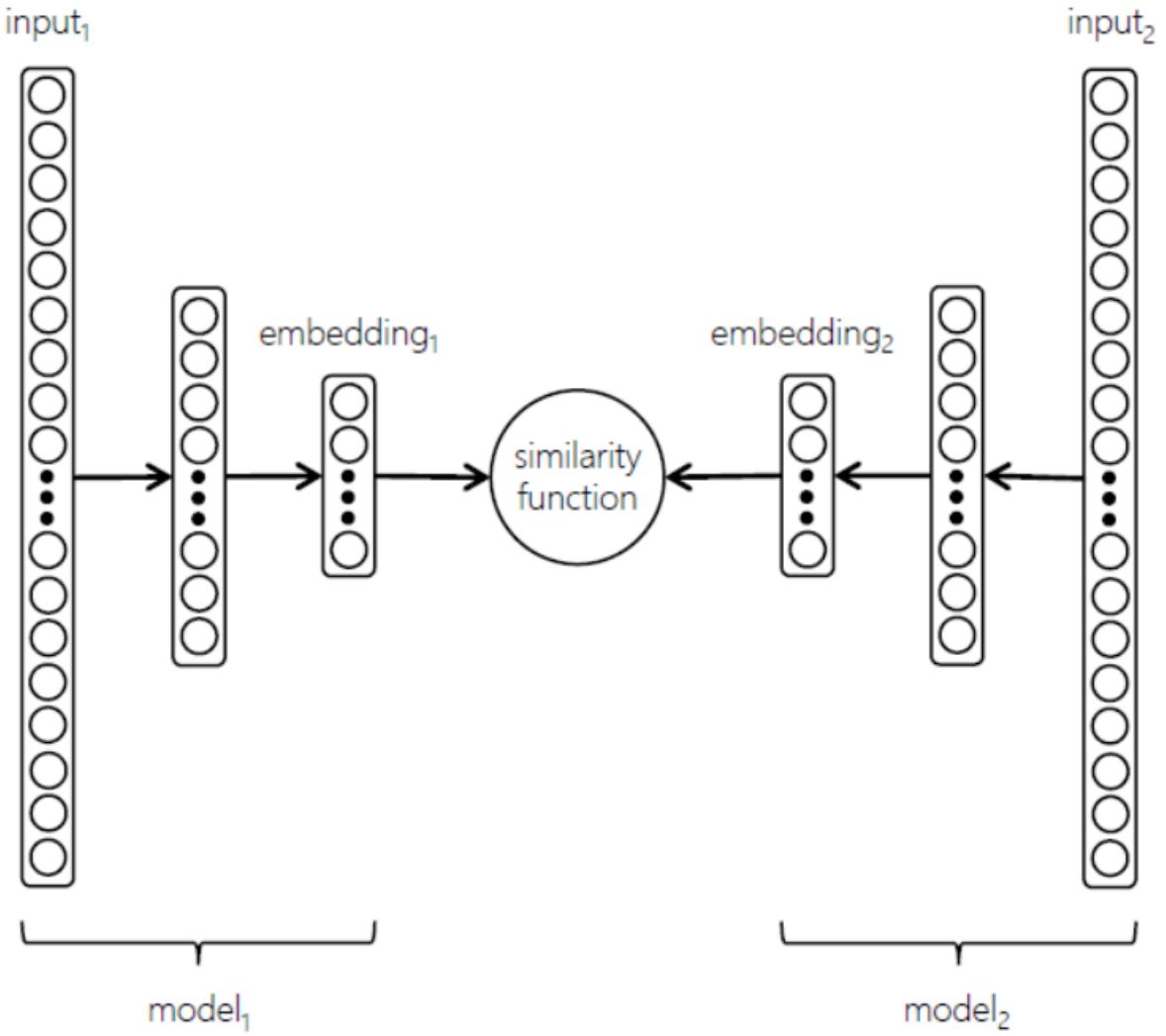
- Learn a compressed representation $\vec{x} \in \mathbb{R}^k$ of items from their higher-dimensional vector representation $\vec{v} \in \mathbb{R}^K$ such that $k \ll K$.



Minimizing the self-reconstruction error.

$$\mathcal{L}_{autoencoder}(\vec{v}, \vec{v'}) = \|\vec{v} - \vec{v'}\|^2$$

Siamese Networks



Retaining the information that is necessary
for determining the similarity between a pair of items
(a query and a document)

Consisting of two models *model*$_1$ and *model*$_2$ that
project *input*$_1$ and *input*$_2$ to $\vec{v_1}$ and $\vec{v_2}$ in common latent space.

A predefined metric (cosine similarity) is used to compute
the similarity between $\vec{v_1}$ and $\vec{v_2}$.

The model parameters are optimized such that $\vec{v_1}$ and $\vec{v_2}$
are closer when the two inputs are expected to be similar,
and further away otherwise.

One possible loss function is the logistic loss. If each training sample consist of a triple $\langle \vec{v_q}, \vec{v_{d1}}, \vec{v_{d2}} \rangle$, such that $sim(\vec{v_q}, \vec{v_{d1}})$ should be greater than $sim(\vec{v_q}, \vec{v_{d2}})$, then we minimize,

$$\mathcal{L}_{siamese}(\vec{v_q}, \vec{v_{d1}}, \vec{v_{d2}}) = \log\left(1 + e^{-\gamma(sim(\vec{v_q},\vec{v_{d1}}) - sim(\vec{v_q},\vec{v_{d2}}))}\right)$$

where, γ is a constant that is often set to 10.

Both autoencoders and Siamese networks learn compressed representations of the input. In autoencoders, the learnt representation is optimized for reducing reconstruction errors, whereas Siamese networks optimize to better discriminate similar pairs from dissimilar ones.

Deep Neural Networks for IR

Corpus for Ad-Hoc Retrieval

- Corpus of search queries
- Corpus of candidate documents
- Ground truth for query-document pairs

- Both large scale corpora of search queries and documents are publicly available for IR research.
- The amount of relevance judgements that can be associated with them are often limited outside of large industrial research labs.

- Data regime: large corpus of documents or queries is available, but limited (or no) labelled data.

Strike a Balance

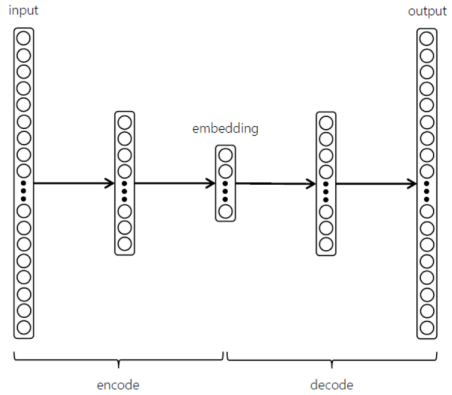
- The number of model parameters and the size of the data available for training.
- The proportion of labelled and unlabeled data that is available influences the level of supervision that can be employed for training these deep models.
- Unsupervised approach
  - The unlabeled document (or query) corpus is used to learn good text representations.
  - These learnt representations are incorporated into an existing retrieval model or a query-document similarity metric.
  - Leverage the available small amounts of labelled data to train a retrieval model with few parameters using text representation pre-trained on larger unlabeled corpus.

DNNs for IR can learn text representations *in situ*, or use pre-trained embeddings. When pre-trained embeddings are employed, care should be taken to make sure the representations are suitable for the task.

Long Document vs. Short Text

- Short texts
  - Question-answering task
  - Document ranking where document representation is based on a short text field like title
- Challenge from short text point of view
  - Vocabulary mismatches are more likely than when the retrieved items contain long text descriptions
  - Matching in a latent space vs. Lexical term-based matching
- Challenge from long document point of view
  - Documents with long body texts may contain mixture of many topics
  - Query matches may be spread over the whole document
  - A neural document ranking model must effectively aggregate the relevant matches from different parts of a long document.

Document Autoencoders

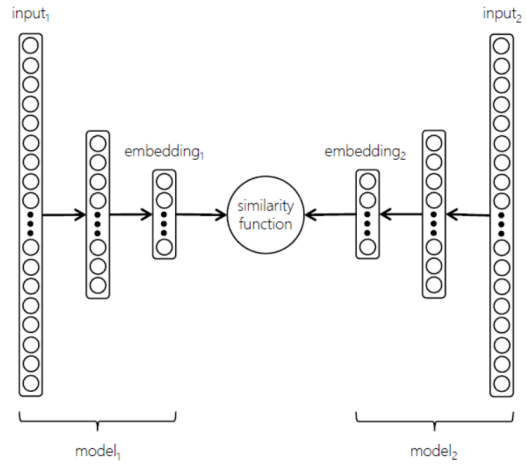


- Semantic Hashing
  - The earliest deep neural models for ad-hoc retrieval
  - A deep autoencoder trained under unsupervised setting on unlabeled document collection.
  - Consider each document as a bag-of-terms and use one-hot vector representation for the terms.
  - Consider only top two thousand most popular terms in the corpus after removing stopwords.
  - Pretrain first the model layer-by-layer, and then train it further end-to-end for additional tuning.
  - After fine tuning, the output of the model is thresholded to generate binary vector encoding of the documents.
  - Given a search query, a corresponding hash is generated, and the relevant candidate documents quickly retrieved that match the same hash vector.
  - A standard IR model can then be employed to rank between the selected documents.

Semantic Hashing

- An example of a document encoder based approach to IR
- Shortcoming of autoencoder
  - Vocabulary sizes of few thousand distinct terms may be too small for most practical IR tasks.
  - It minimizes the document reconstruction error which may not align well with the goal of the target IR task.
- Solutions
  - A larger vocabulary or a different term representation strategy, e.g., the character trigraph based representation may be considered in practice.
  - To train on query-document paired data where the choice of what constitutes as the minimal sufficient statistics of the document is influenced by what is important for determining relevance of the document to likely search queries.

Siamese Networks for Short-Text Matching



- Deep Semantic Similarity Model (DSSM)
  - Train on query and document title pairs where both the pieces of texts are represented as bags-of-character-trigraphs.
  - Consist of two deep models for the query and the document with all fully-connected layers and cosine distance as the choice of similarity function in the middle. (Note: convolutional layers, recurrent layers, and tree structured networks have been explored)
  - Minimizing the cross-entropy loss

$$\mathcal{L}_{dssm}(q, d^+, D^-) = -\log\left(\frac{e^{\gamma \cdot cos\left(\vec{q},\vec{d^+}\right)}}{\sum_{d \in D} e^{\gamma \cdot cos\left(\vec{q},\vec{d}\right)}}\right)$$

$$\text{where,} \quad D = \{d^+\} \cup D^-$$

SERP: Search Engine Result Page

a query q, a positive document d^+ (a document that was clicked by a user on the SERP for that query), and a set of negative documents D^- randomly sampled with uniform probability from the full collection

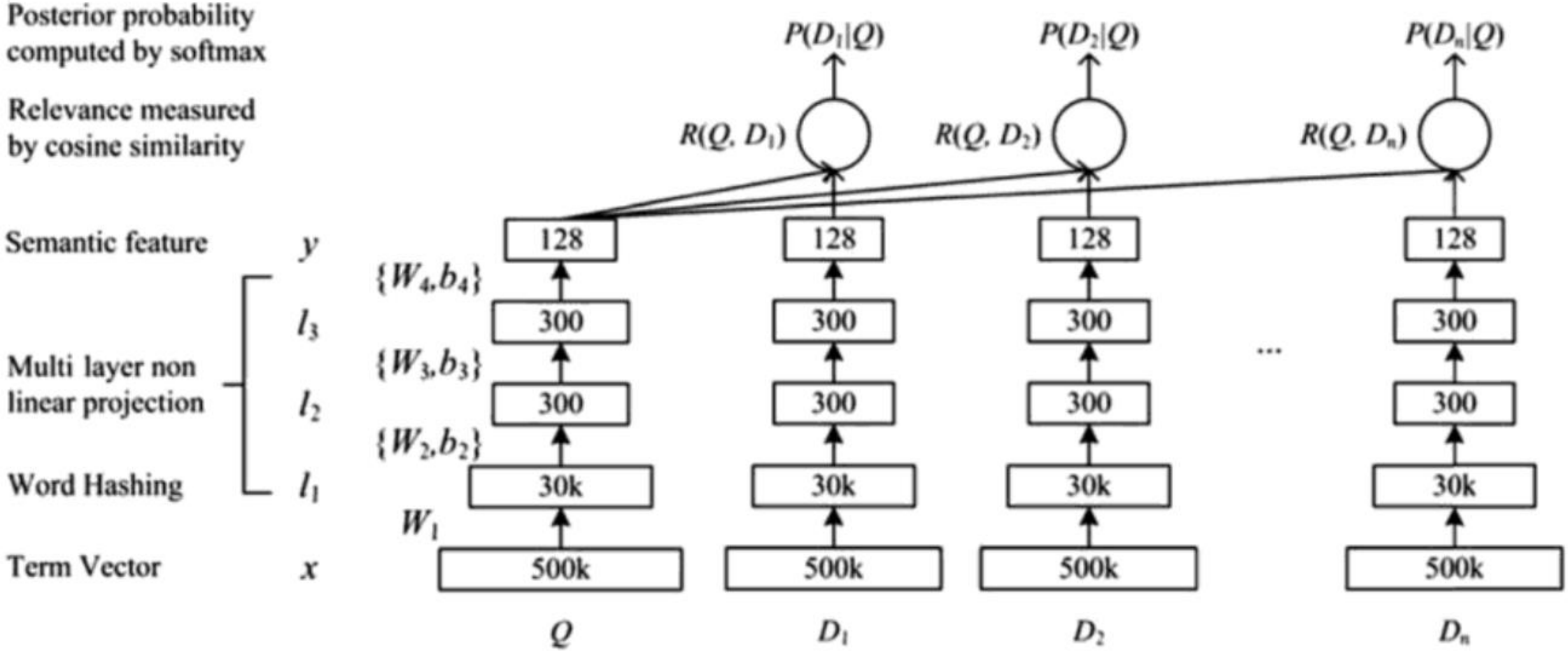


Figure 7.2 Illustration of the DSSM. From *P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, L. Heck, Learning deep structured semantic models for web search using clickthrough data, in: Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, CIKM '13, ACM, New York, NY, USA, 2013, pp. 2333–2338*

Alternatives

- Instead of deep fully-connected architecture for the query and the document models, convolution layers, recurrent layers, and tree-structured networks have also been explored.
- The similarity function can also be parameterized and implemented as additional layers of the neural network.

Convolutional DSSM (CDSSM)

- Replace the adoption of bag of words with the concatenation of term vectors in a sequence on the input.
- Learn each term within a given context window in the order of a word sequence to capture the n-gram based contextual features.
- Based on the significance learned from the n-gram features, the CDSSM structures the feature vectors up onto the sentence level.
- Extract the local contextual features from n-gram word level and the global contextual features from the max-pooling the sentence level in the text corpus.

Notions of Similarity for Typical and Topical

- In the case of Siamese networks, such as CDSSM, the notion of similarity being modelled depends on the choice of the paired data that the model is trained on.
- When the CDSSM is trained on query and document title pairs, the notion is more topical in nature.
- The same CDSSM trained on query prefix-suffix pairs captures a more typical notion of similarity.
- The CDSSM trained on session-query pairs is amenable to vector-based analogies.

$$\vec{v}_{\text{things to do in london}} - \vec{v}_{\text{london}} + \vec{v}_{\text{new york}} \approx \vec{v}_{\text{new york tourist attractions}}$$

$$\vec{v}_{\text{university of washington}} - \vec{v}_{\text{seattle}} + \vec{v}_{\text{denver}} \approx \vec{v}_{\text{university of colorado}}$$

$$\vec{v}_{\text{new york}} + \vec{v}_{\text{newspaper}} \approx \vec{v}_{\text{new york times}}$$

泰勒·艾莉森·絲薇芙特，是一名美國創作歌手、音樂製作人、慈善家、導演及演員。

seattle		taylor swift	
Query-Document	Prefix-Suffix	Query-Document	Prefix-Suffix
weather seattle	chicago	taylor swift.com	lady gaga
seattle weather	san antonio	taylor swift lyrics	meghan trainor
seattle washington	denver	how old is taylor swift	megan trainor
ikea seattle	salt lake city	taylor swift twitter	nicki minaj
west seattle blog	seattle wa	taylor swift new song	anna kendrick

梅根·伊莉莎白·崔娜，是一個美國創作歌手。

奧妮卡·譚雅·邁拉婕·佩禔，是一名美國饒舌、詞曲作家、歌手、演員及模特兒。

安娜·庫克·坎卓克，是一名美國演員及歌手。

Lessons

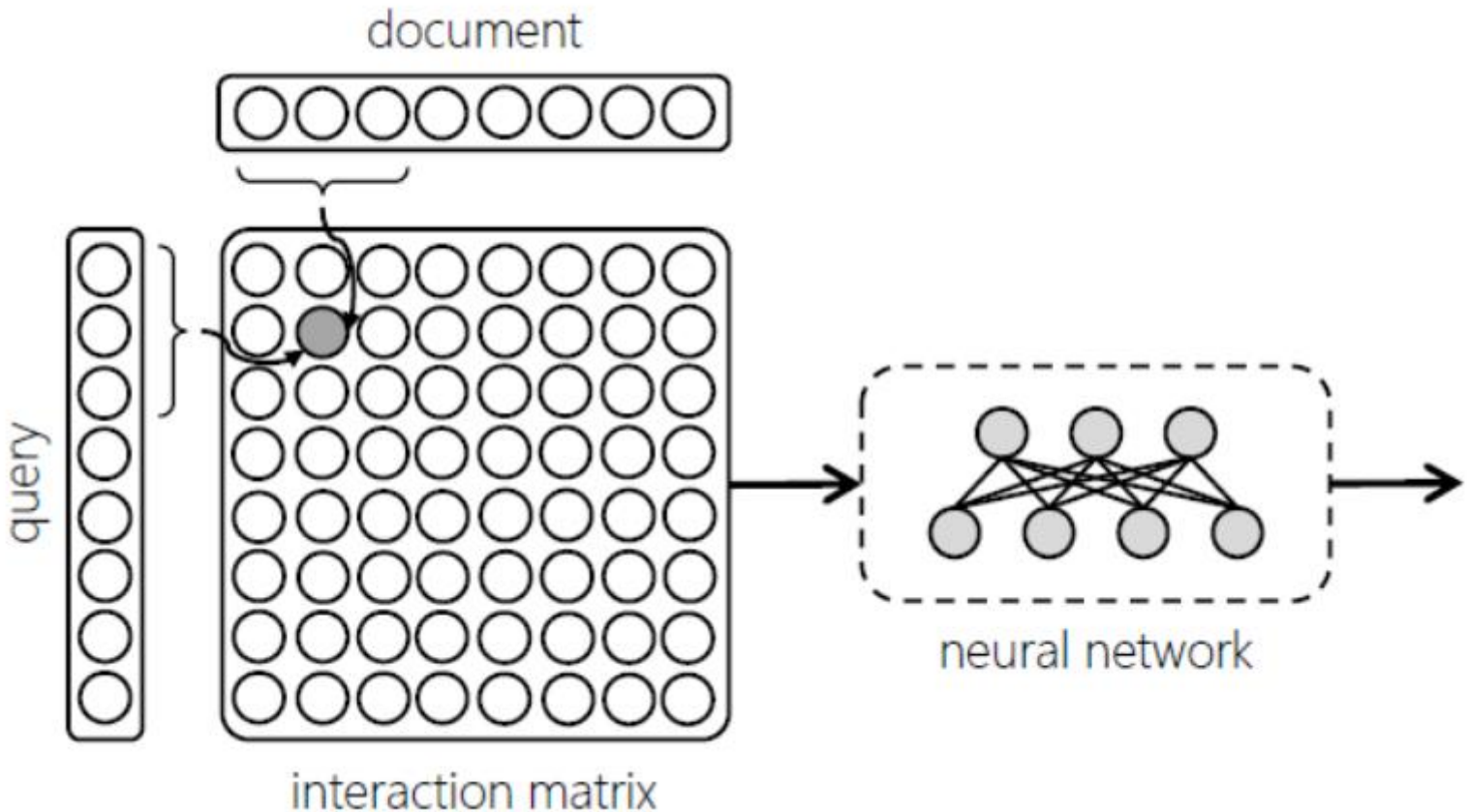
- Siamese networks represent both the query and the document using single embedding vectors.
- Compare different parts of the query with different parts of the documents, and aggregating these partial evidences of relevance.

Interaction-based Networks

Long documents may contain a mixture of many topics. So, comparing the query representation individually to windows of text from the document may be more effective than learning a single vector representation for the document.

Interaction-based Networks

- A sliding window is moved over both the query and the document text.
- Each instance of the window over the query is compared (interacts) over the document text.
- The terms with each window can be represented in different ways, i.e., one-hot vectors, pre-trained embedings, or embedings updated during the model training.
- A neural model (e.g., convolutional) operates over the generated interaction matrix and aggregates the evidence across all the pairs of windows compared.
- Explored both for short text matching and for ranking long documents.


document
query
interaction matrix
neural network

Lexical and Semantic Matching

- The representation learning models tend to perform poorly when dealing with rare terms and search intents.
  - Lexical matching: it is easier to estimate relevance based on patterns of exact matches of the rare term
  - Semantic matching: a neural model focusing on matching in the latent space is unlikely to have good representation for this rare term

The President of the United States of America (POTUS) is the elected head of state and head of government of the United States. The president leads the executive branch of the federal government and is the commander in chief of the United States Armed Forces. Barack Hussein Obama II (born August 4, 1961) is an American politician who is the 44th and current President of the United States. He is the first African American to hold the office and the first president born outside the continental United States.

(a) Lexical model

The President of the United States of America (POTUS) is the elected head of state and head of government of the United States. The president leads the executive branch of the federal government and is the commander in chief of the United States Armed Forces. Barack Hussein Obama II (born August 4, 1961) is an American politician who is the 44th and current President of the United States. He is the first African American to hold the office and the first president born outside the continental United States.

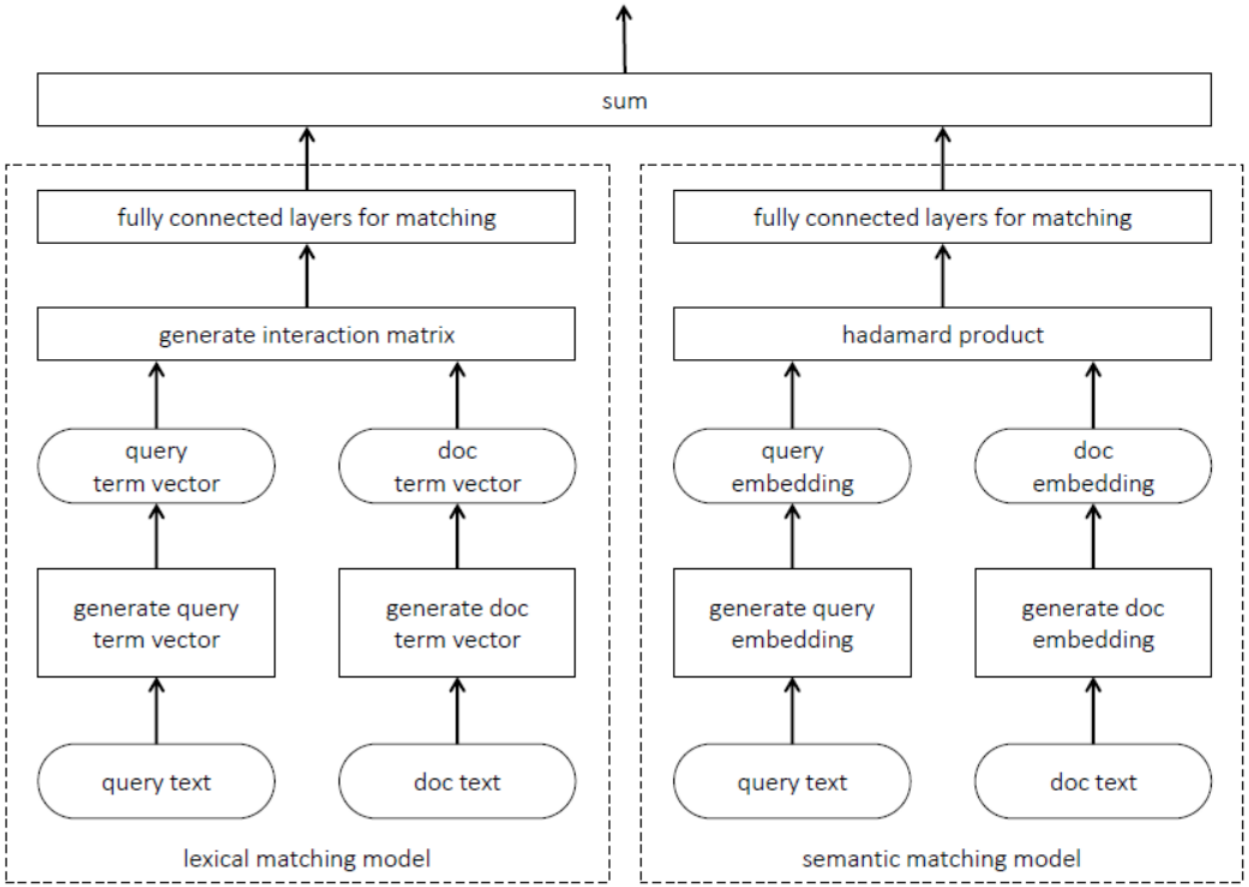
(b) Semantic model

Figure 7.2: Analysis of term importance for estimating the relevance of a passage to the query "United States President" by a lexical and a semantic deep neural network model. The lexical model only considers the matches of the query terms in the document but gives more emphasis to earlier occurrences. The semantic model is able to extract evidence of relevance from related terms such as "Obama" and "federal".

Duet Architecture

- Learn to identify good patterns of both lexical and semantic matches jointly.

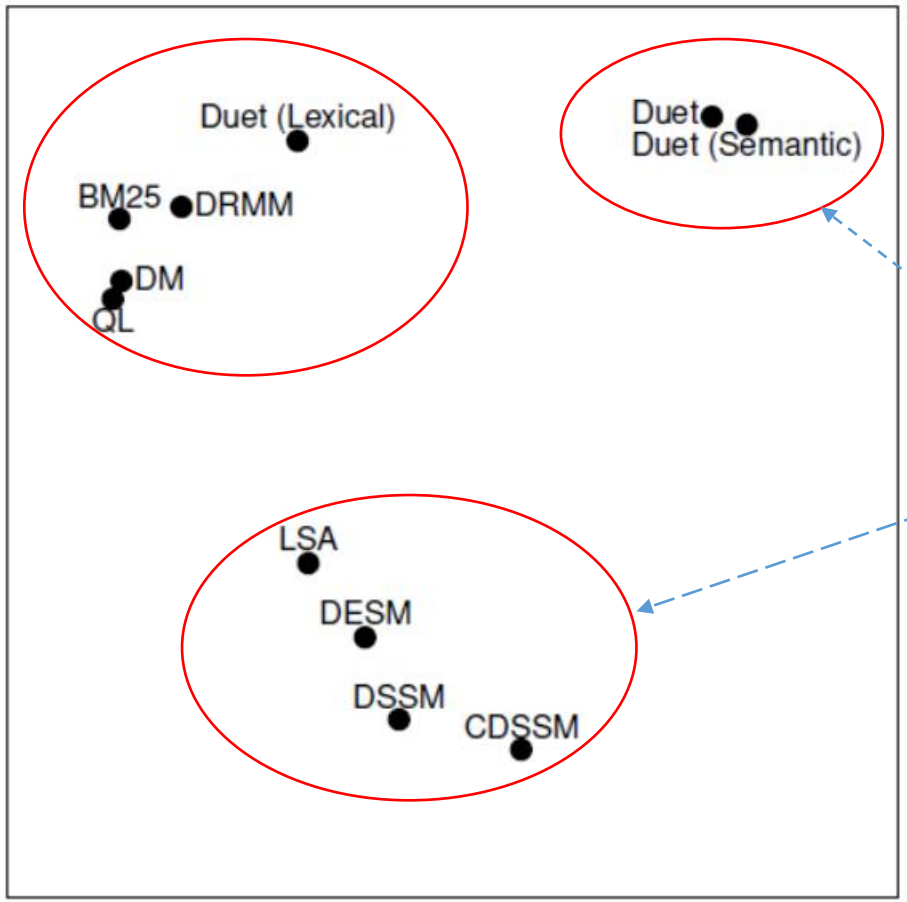
Use a convolutional model that operates over a binary interaction matrix



$$(A \circ B)_{ij} = (A)_{ij}(B)_{ij}.$$

Learn representations of query and document text for effective matching in the latent space

Lexical matching models



Models focusing on representation learning

Figure 7.4: A demonstration that IR models that focus on lexical matching tend to perform well on queries that are distinct from queries on which semantic matching models achieve good relevance. Each model is represented by a vector of NDCG scores achieved on a set of test queries. For visualization, t-SNE (Maaten and Hinton, 2008) is used to plot the points in a two-dimensional space. Lexical matching models (BM25, QL, DM, DRMM, and Duet-Lexical) are seen to form a cluster—as well as the models that focus on representation learning.

IR models that use exact matching tend to perform well on different queries than models that employ inexact matching. Models that combine both performs best.

Matching with Multiple Document Fields

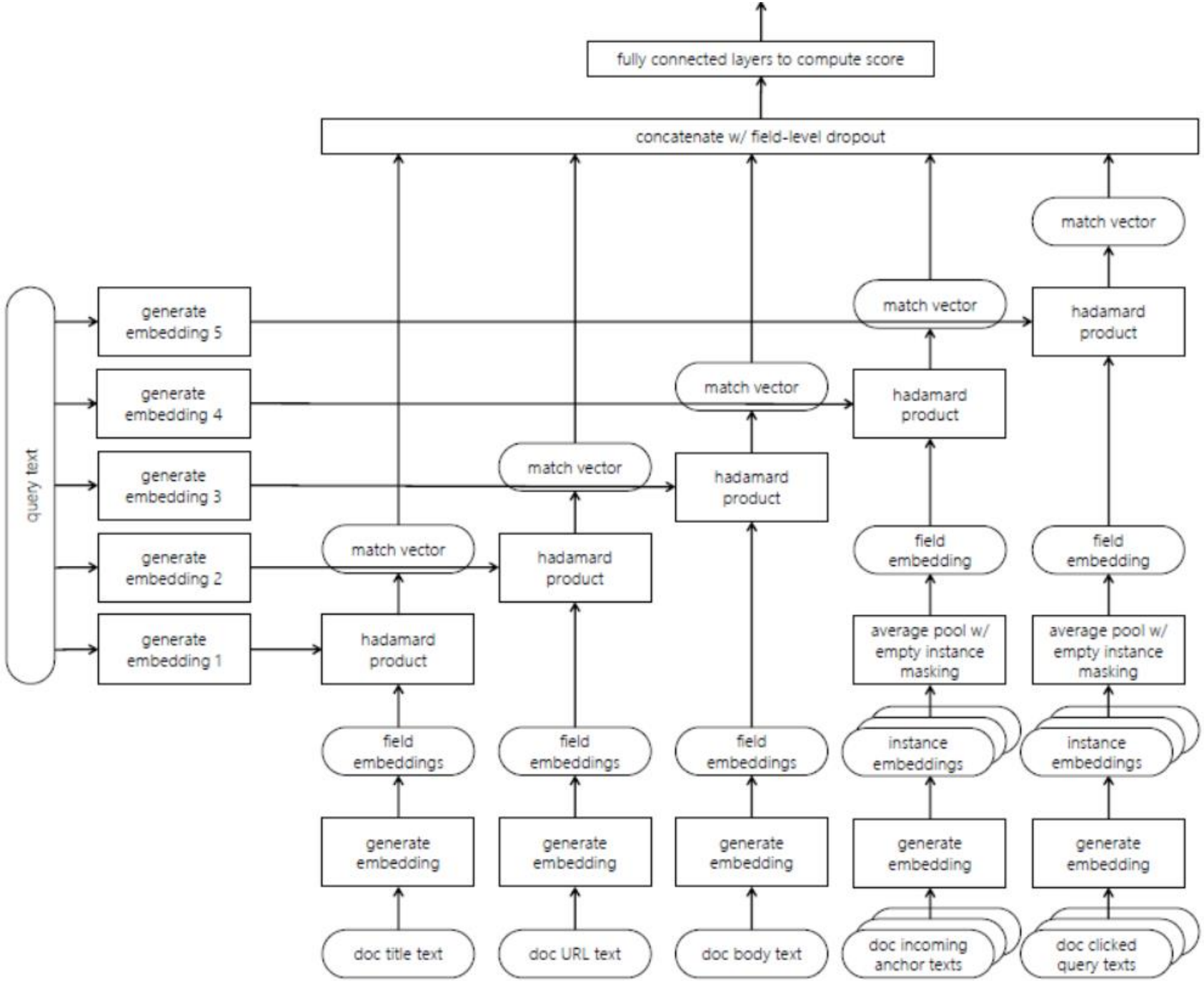
- Consider more than just the document content for matching in commercial web search engines.
  - Anchor texts corresponding to incoming hyperlinks
  - The query text for which the document may have been previous viewed
- Learning separate latent spaces for matching the query against the different document fields is more effective than using a shared latent space for all the fields.

《怒吼》（英語：Roar）是美國歌手凱蒂·佩芮的一首歌曲，
收錄於她的第四張專輯《超炫光》。

Table 7.2: Different document fields may be useful for matching different aspects of the query intent. For example, for the query "roar music video" the document title may be useful for matching the name of the song, whereas the URL may be more informative on whether the document is likely to satisfy the video intent. A neural representation learning model may emphasize on the video intent in the query embedding that it learns to match against the URL field and pay more attention to matching the name of the song against the title field. By combining the two signals the model can better discriminate the relevant document from the rest.

	Text fields		Field matches		
	URL	Title	URL	Title	Combined
Doc #1	youtube.com/watch?v=CevxZvSJLk8	Katy Perry - Roar	✓	✓	✓
Doc #2	youtube.com/watch?v=nfWlot6h_JM	Taylor Swift - Shake It Off	✓	✗	✗
Doc #3	wikipedia.org/wiki/Roar_(song)	Roar (song)	✗	✓	✗
Doc #4	wikipedia.org/wiki/Shake_It_Off	Shake It Off	✗	✗	✗

Taylor Swift於2014年發布的歌曲


fully connected layers to compute score
concatenate w/ field-level dropout
query text
generate embedding 5
generate embedding 4
generate embedding 3
generate embedding 2
generate embedding 1
match vector
hadamard product
field embeddings
average pool w/ empty instance masking
instance embeddings
field embedding
generate embedding
doc title text
doc URL text
doc body text
doc incoming anchor texts
doc clicked query texts

Methods of Matching Functions Learning

Runjie Zhu, Xinhui Tu, Jimmy Xiangji Huang, Deep Learning on Information Retrieval and Its Applications, Chapter 7, Deep Learning for Data Analytics, Academic Press, 2020, pages 125-153.

Typical Flow of a Representation Learning Matching System

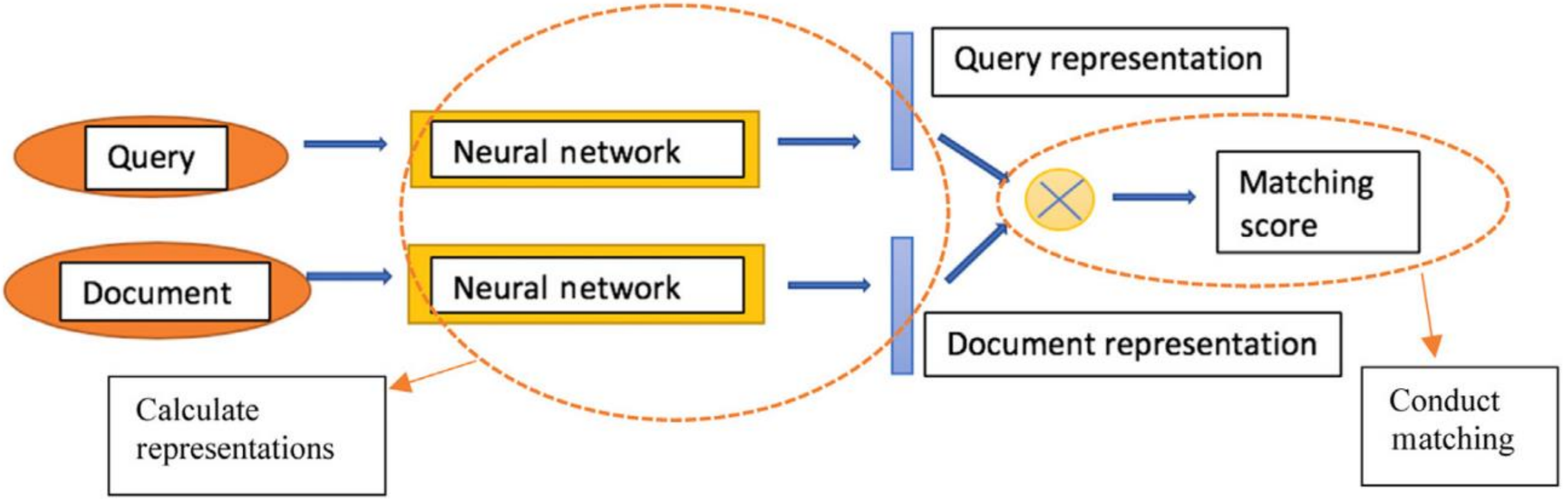


Figure 7.1 Typical flow of a representation learning matching system.

DNN-based Methods

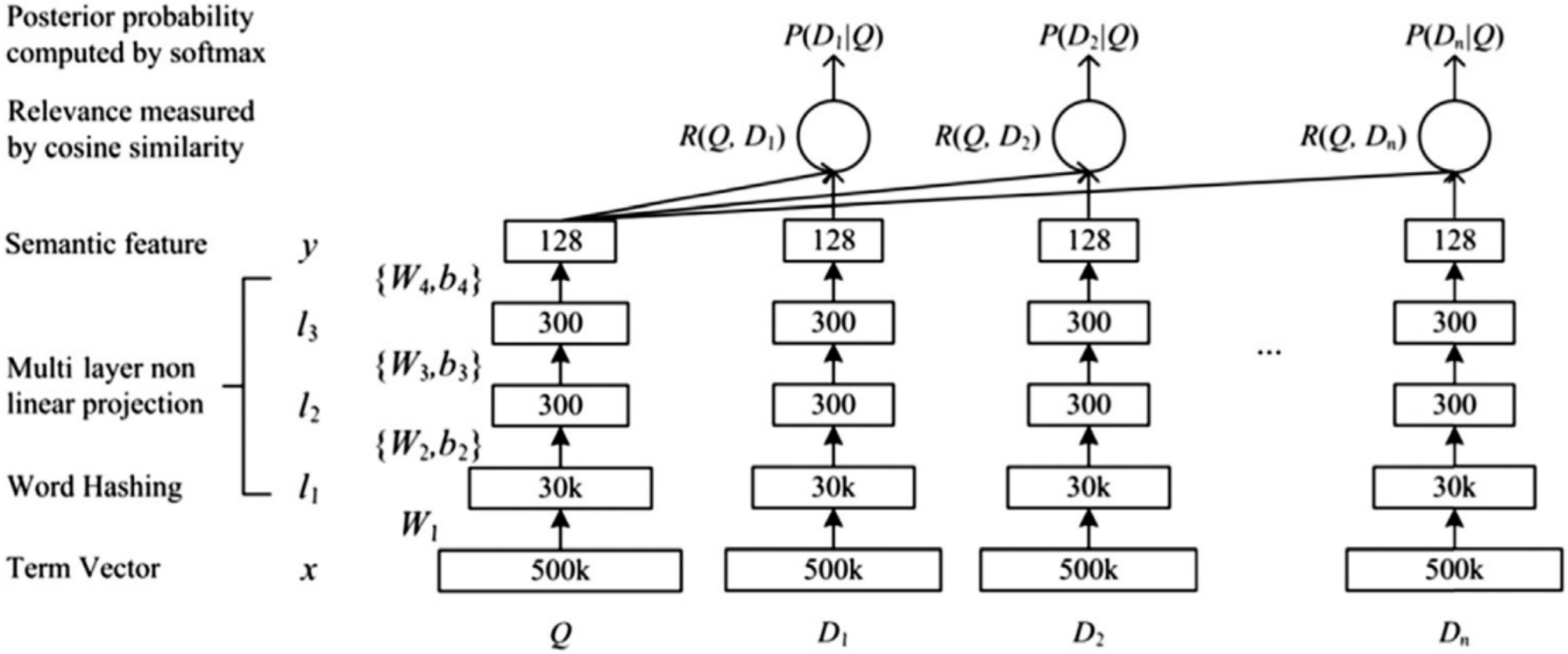


Figure 7.2 Illustration of the DSSM. From *P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, L. Heck, Learning deep structured semantic models for web search using clickthrough data, in: Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, CIKM '13, ACM, New York, NY, USA, 2013, pp. 2333—2338*

CNN-Based Methods

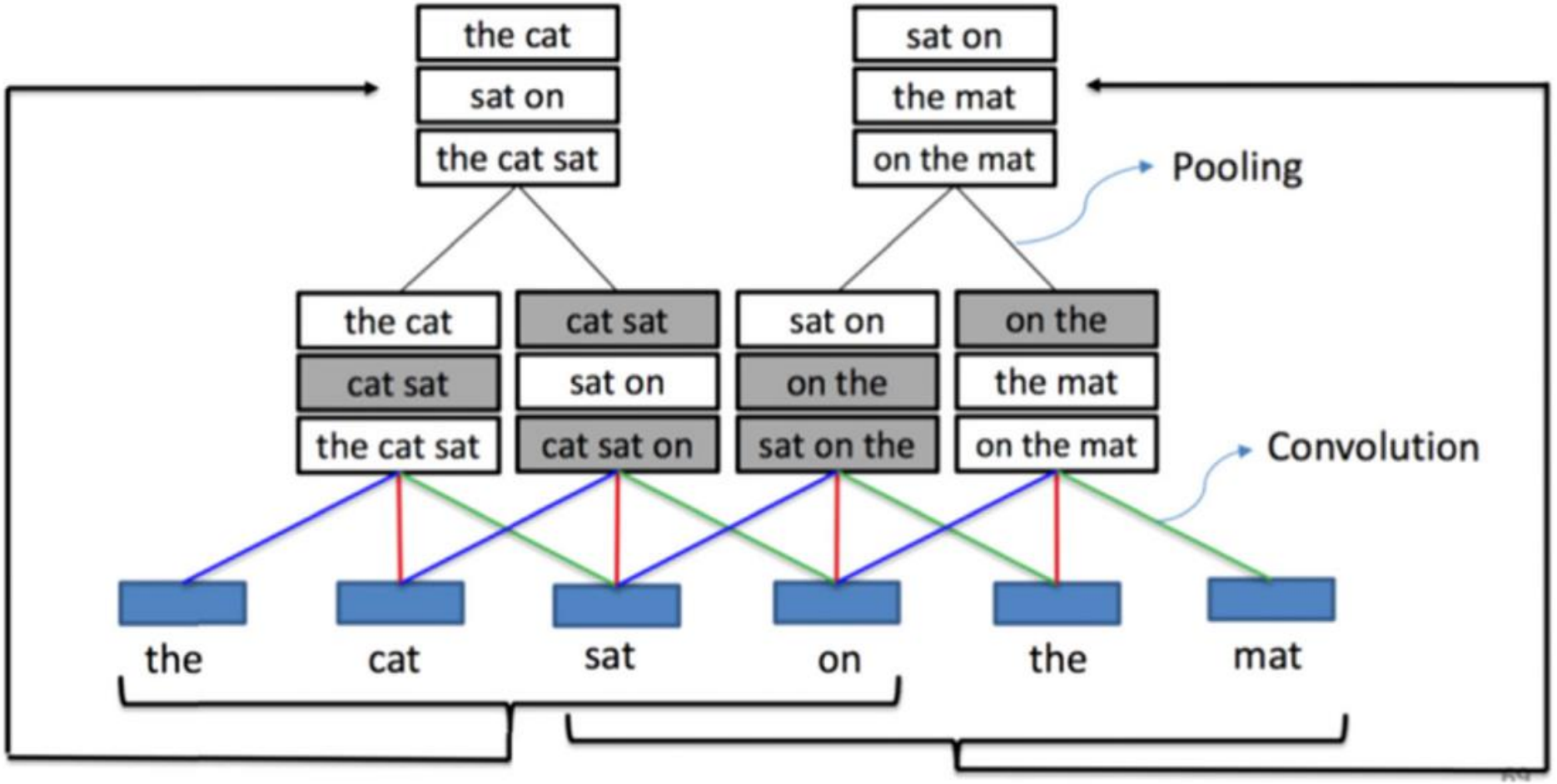


Figure 7.3 Example of CNN-based representation learning methods. *X. He, J. Gao, L. Deng, Deep learning for natural language processing: theory and practice tutorial, in: CIKM'14 Tutorial.*

RNN-based Methods

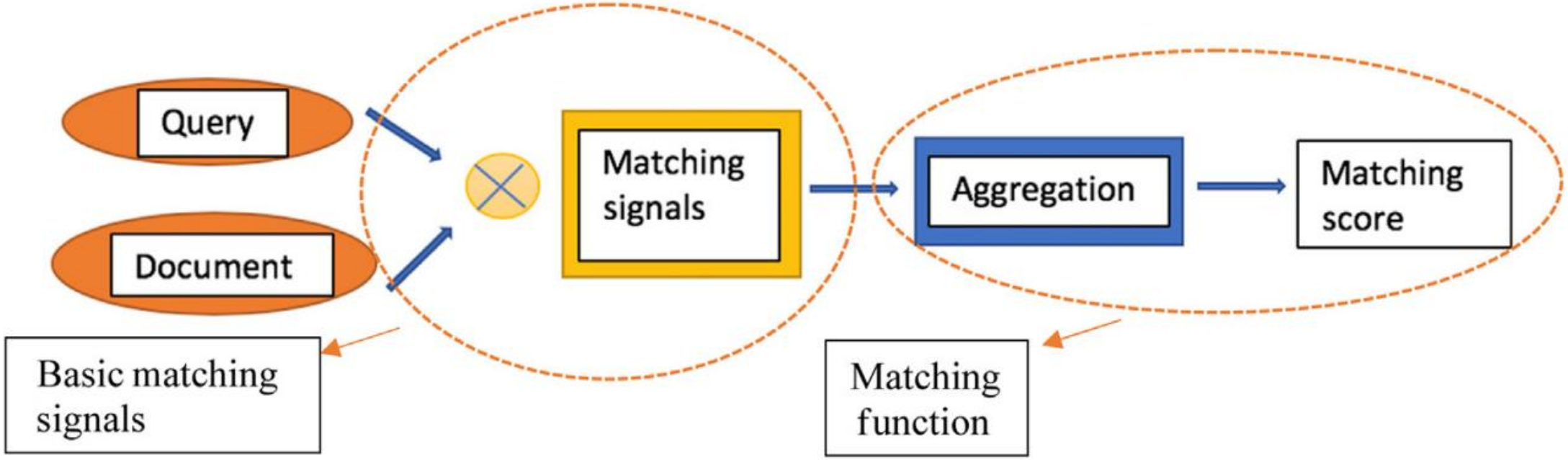
- RNN-based methods are able to capture and store long-dependence relationships.
- The long short-term memory (LSTM) model and the gated recurrent unit (GRU) are the popular variations in RNN-based representation learning methods.
- In real applications, these RNN models are with certain limitations in that they can only look a few steps back.

LSTM-based RNN Model

- To solve the sentence representation learning issue in IR.
- Capture the semantic meaning of the entire sentence as well as the most salient and less important words of the sentence.
- Takes in and extracts each term within a sentence in a sequential way, in order to embed them into a semantic vector.
- Use a sequence of letter trigrams as inputs to generate the embedding vector of the whole sentence.

Matching Function Learning

- Pull raw data inputs from queries and documents directly to the neural networks to seek and construct basic low-level matching signals, and then the algorithms aggregate the results to generate the final matching patterns and scores.

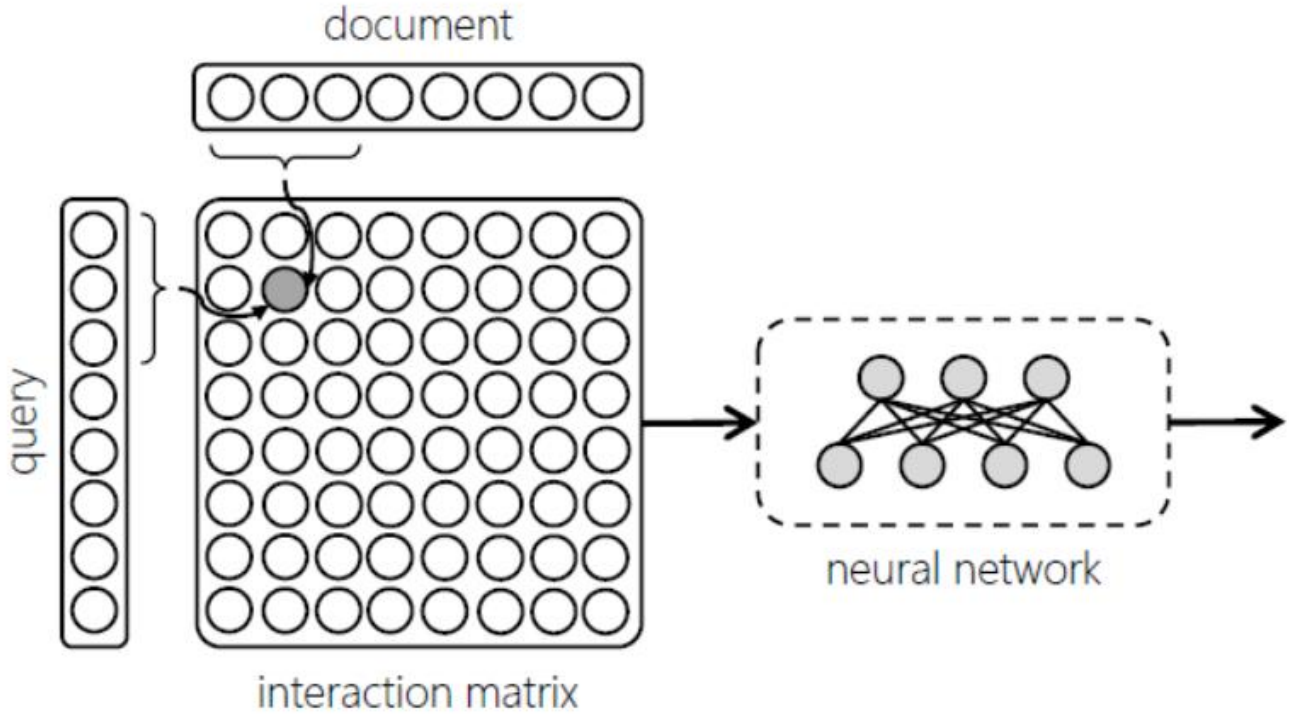


Four Major Classes

- Matching with word-level similarity matrix
- Matching with attentional models
- Matching with transformer models
- Methods of combining matching function learnings along with representation learnings

Matching with Word-Level Similarity Matrix

- The two target sentences interact before generating the higher-level word representations directly from context.
- The basic matching signal is functioned by the phrase sum interaction matrix



Matching with Attention Model

- The model takes attendance to the soft-alignment of the words that appeared in both the user’s queries and documents.
- The model uses the parallel structure to evaluate and compare the word-aligned in the subphrase to calculate the matching scores respectively.
- The model collects and consolidates all the output matching signals from subphrases to generate a final matching scores of the two target sentences.

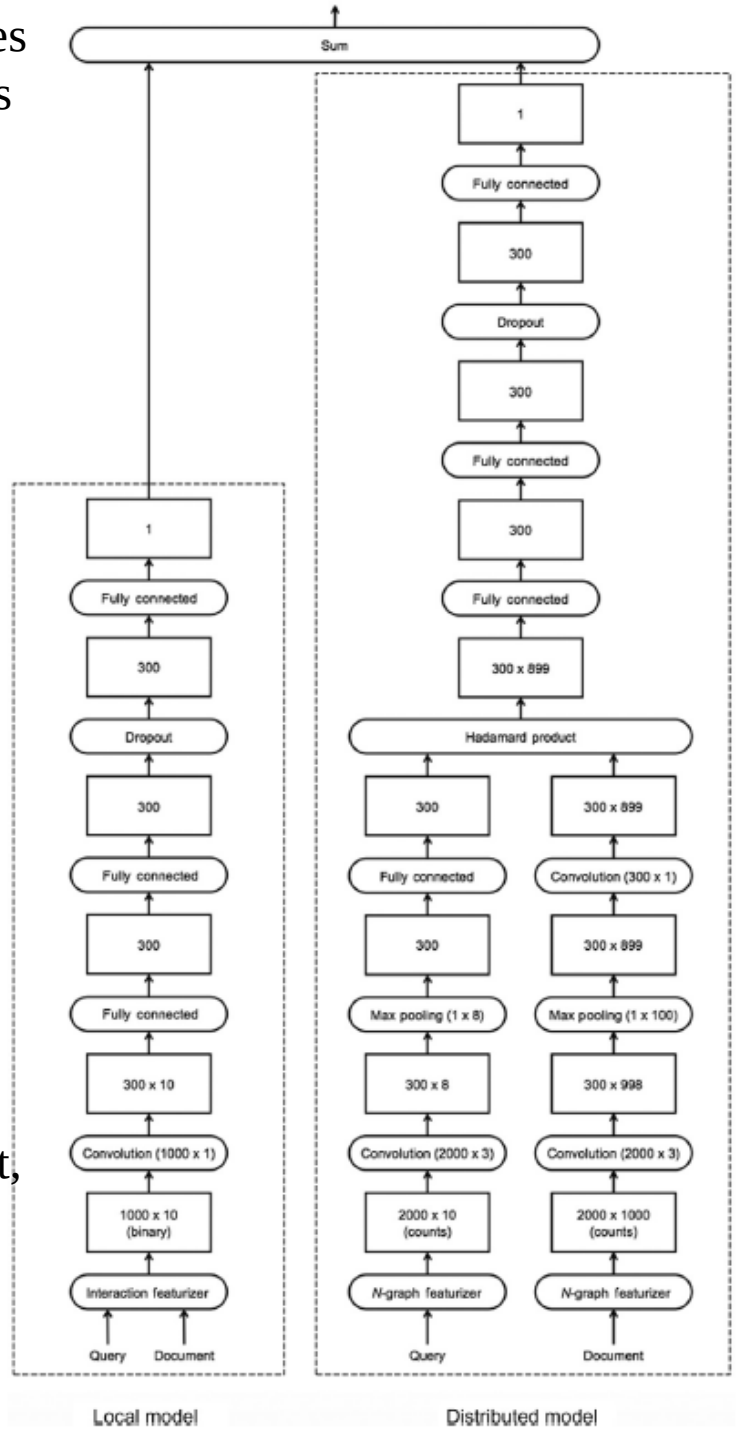
Matching with Transformer Model

- BERT Model in IR
  - The feature-based task
  - The fine-tuning task
- The feature-based BERT model
  - The BERT structure serves to generate representations of users' queries and documents, respectively
  - Combine them with cosine similarities to compute the final ranking score.
- The tuning-based BERT model
  - Predict the degree of relevance of a query-document pair.

Combining Matching Function Learning and Representation Learning

- The matching function learning is constructed for learning local matchings while the representation learning is applied to distributed matchings.

A linear combination of the scores generated by the two components



The representation learning-based component uses simple feedforward neural networks to compute the similarity scores.

Applies an indicator matrix to indicate where the term occurrence happens within a document, and then it uses convolutional layers to extract the salient features.

Methods of Relevance Learning

Similarity Matching vs. Relevance Matching

- Relevance matching is especially important as query and document comparisons on similarity do not always reflect the degree of relevance between them.
- For similarity matching, the models usually adopt a symmetric matching function and are limited to the same text formats comparison.
- Relevance matching usually contains an asymmetric matching function and could contain all different forms, ranging from keywords to documents, to phrases and sentences, and to documents.

Relevance Learning

- (1) Relevance learning based on global distribution of matching strengths.
  - Calculate each query term's matching signals across the entire document, followed by the calculation of the distribution of global matching strengths
  - Consolidate the matching strengths' distributions
- (2) Relevance learning based on local context of matched terms.
  - Detect the local contexts around the target term in the documents first, before it conducts relevance matching between the user's queries and the contexts of the term.
  - The relevance matching scores of the local matching signals are consolidated and output as a final result.

References

- Bhaskar Mitra and Nick Craswell, *An Introduction to Neural Information Retrieval*, Foundations and Trends in Information Retrieval, vol. 13, no. 1, pp. 1-126, 2018.
- Runjie Zhu, Xinhui Tu, Jimmy Xiangji Huang, Deep Learning on Information Retrieval and Its Applications, Chapter 7, Deep Learning for Data Analytics, Academic Press, 2020, pages 125-153.