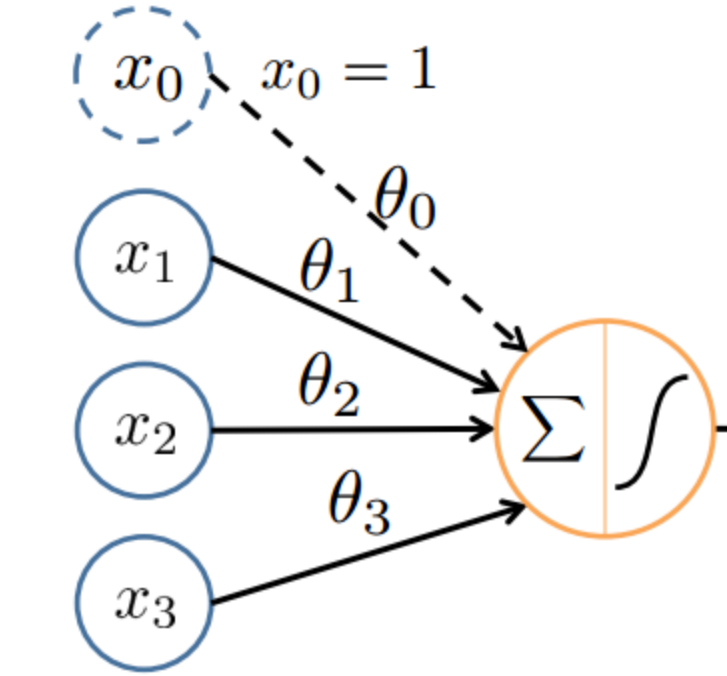
# Advanced Retrieval Models

Some slides courtesy Andrew Ng, Stanford

Bhaskar Mitra, Microsoft

# Neural Networks

# Neural Networks: Perceptron

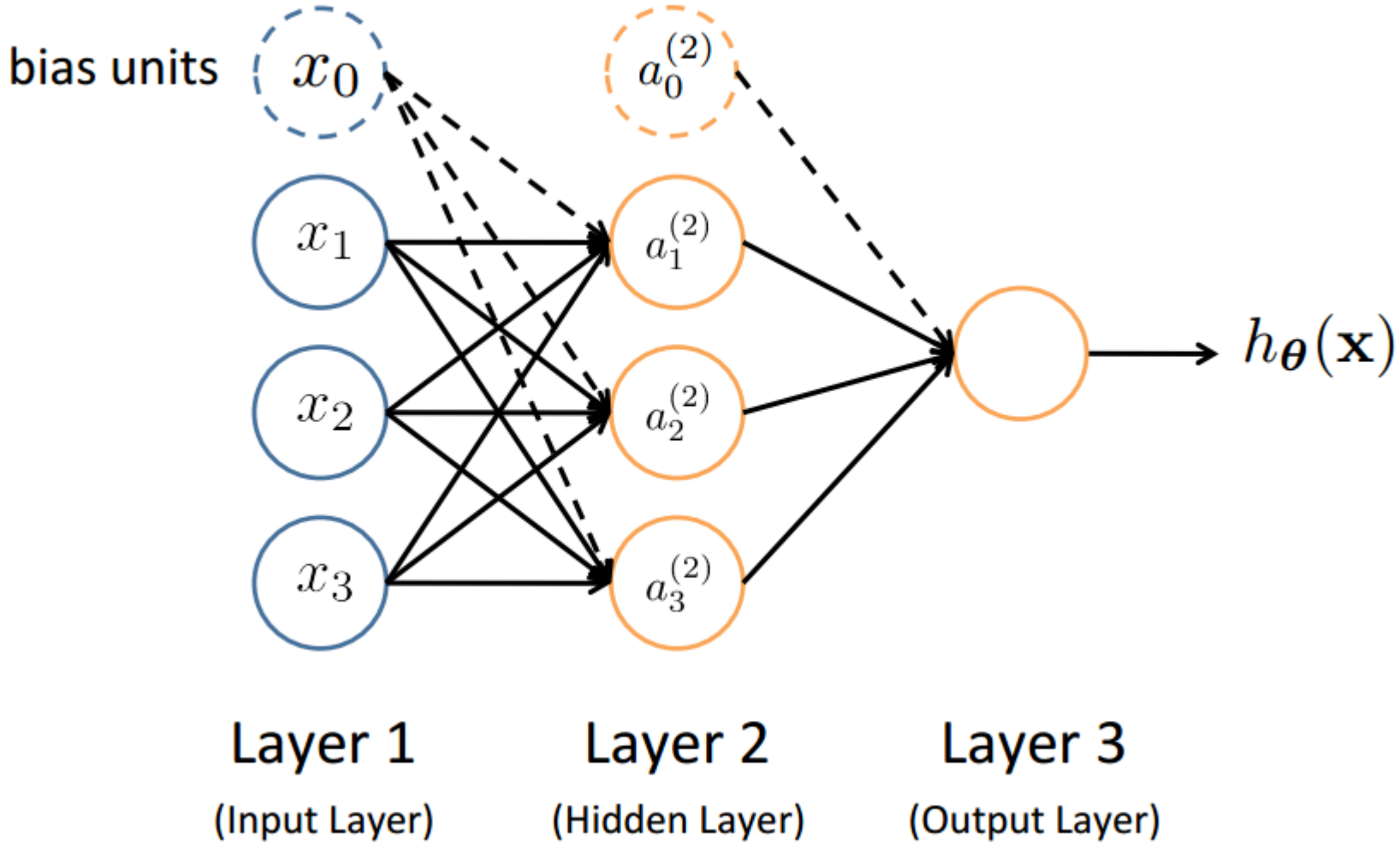



$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}\right) = \frac{1}{1 + e^{-\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}}}$$

Sigmoid (logistic) activation function: $g(z) = \dfrac{1}{1 + e^{-z}}$

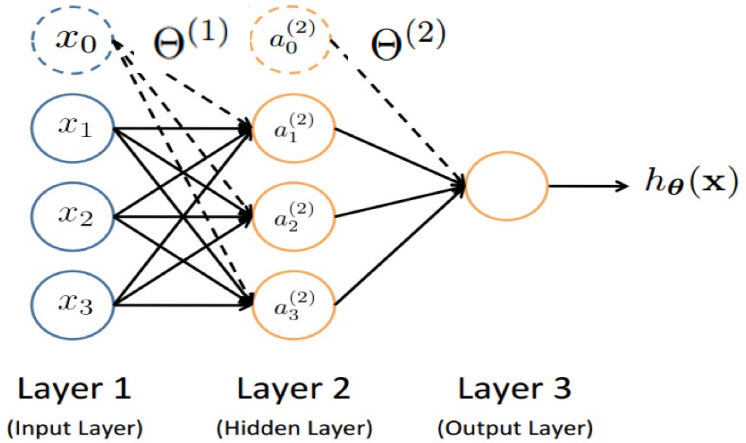
# Neural Network (Feed forward)




Slide by Andrew Ng

# Feed-Forward Process

- Input layer units are features (in NLP/IR, e.g., words)
  - Usually, one-hot vector or word embedding
- Working forward through the network, the **input function** is applied to compute the input value
  - E.g., weighted sum of the input
- The **activation function** transforms this input function into a final value
  - Typically a **nonlinear** function (e.g, **sigmoid**)




$a_i^{(j)} =$ "activation" of unit $i$ in layer $j$

$\Theta^{(j)} =$ weight matrix controlling function mapping from layer $j$ to layer $j+1$

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$
$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

If network has $s_j$ units in layer $j$ *and* $s_{j+1}$ units in layer $j+1$, then $\Theta^{(j)}$ has dimension $s_{j+1} \times (s_j+1)$ .

$$\Theta^{(1)} \in \mathbb{R}^{3\times 4} \qquad \Theta^{(2)} \in \mathbb{R}^{1\times 4}$$

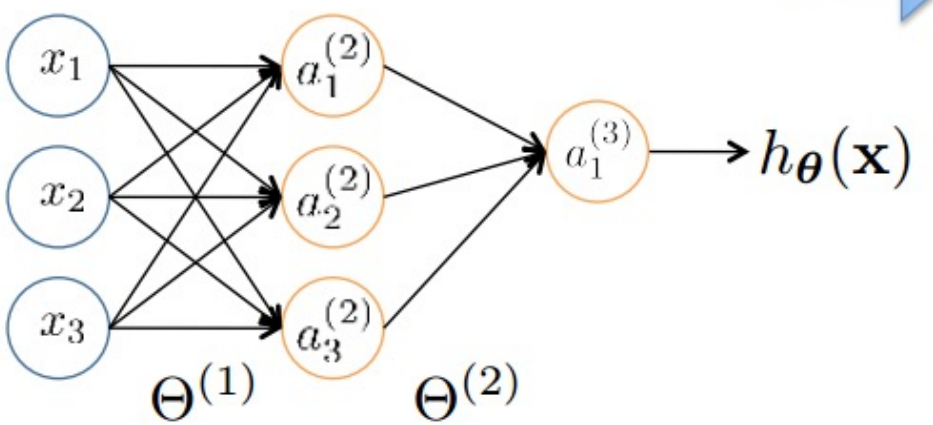
# Vector Representation

$$a_1^{(2)} = g\left(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3\right) = g\left(z_1^{(2)}\right)$$

$$a_2^{(2)} = g\left(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3\right) = g\left(z_2^{(2)}\right)$$

$$a_3^{(2)} = g\left(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3\right) = g\left(z_3^{(2)}\right)$$

$$h_\Theta(\mathbf{x}) = g\left(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)}\right) = g\left(z_1^{(3)}\right)$$




**Feed-Forward Steps:**

$$\mathbf{z}^{(2)} = \Theta^{(1)}\mathbf{x}$$

$$\mathbf{a}^{(2)} = g(\mathbf{z}^{(2)})$$

Add $a_0^{(2)} = 1$

$$\mathbf{z}^{(3)} = \Theta^{(2)}\mathbf{a}^{(2)}$$

$$h_\Theta(\mathbf{x}) = \mathbf{a}^{(3)} = g(\mathbf{z}^{(3)})$$

Based on slide by Andrew Ng

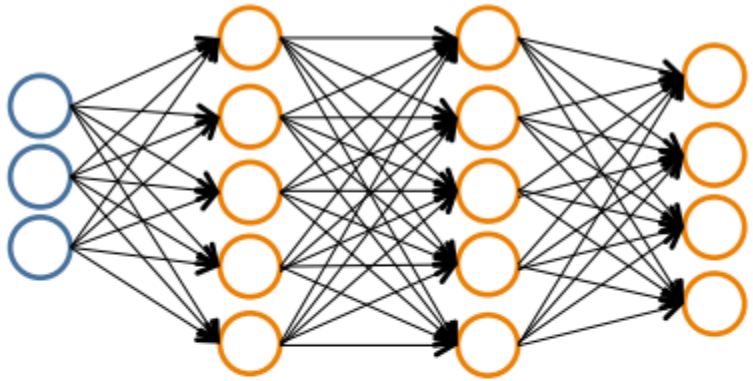
# Can extend to multi-class

Pedestrian

Car

Motorcycle

Truck



$$h_\Theta(\mathbf{x}) \in \mathbb{R}^K$$

We want:

$$h_\Theta(\mathbf{x}) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

when pedestrian

$$h_\Theta(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

when car

$$h_\Theta(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

when motorcycle

$$h_\Theta(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

when truck

Slide by Andrew Ng

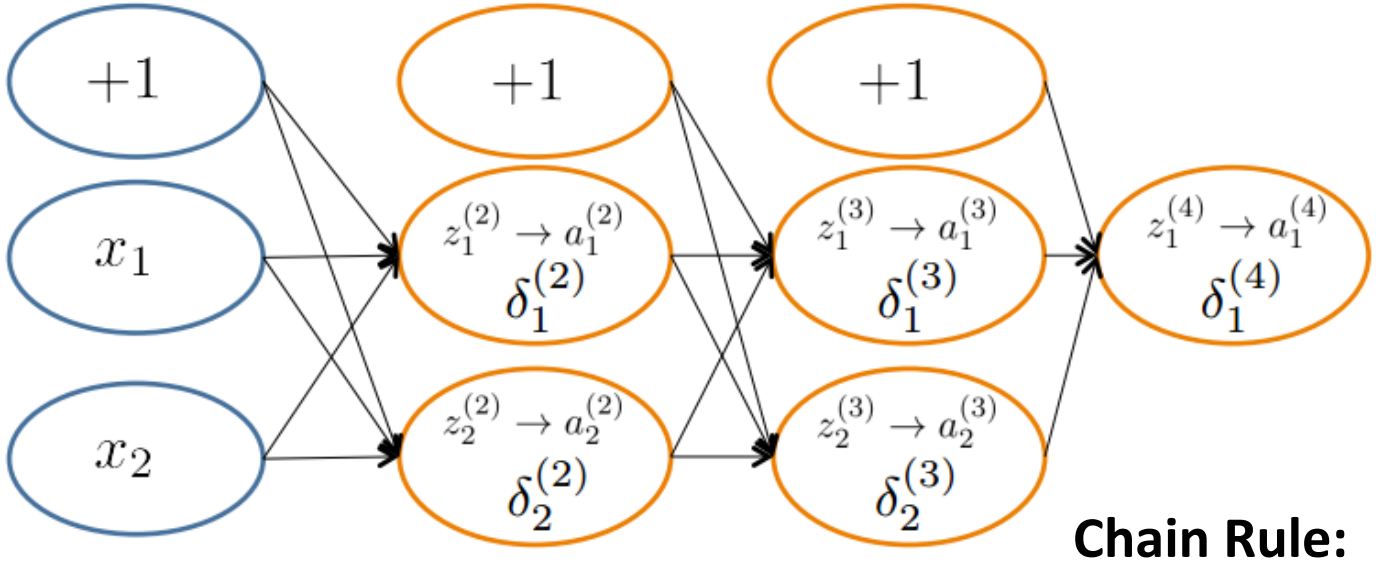
# Recap: Training in Logistic Regression

Let $\mathrm{h}_\theta(x_i) = 1/(1 + e^{-\theta^T x_i})$ (probability $y = 1$ given $x_i$)

$$\min_{\boldsymbol{\theta}} \frac{1}{n} \sum_i \mathrm{y_i} \log(\, h_\theta(x_i)) + (1 - y_i)\, (\, \log(1 - h_\theta(x_i))$$

Gradient descent for learning model parameters

# Neural Net Training: Backpropagation




**Chain Rule:**

$$\frac{d}{dt} f(g(t)) = f'(g(t))g'(t) = \frac{df}{dg} \cdot \frac{dg}{dt}$$

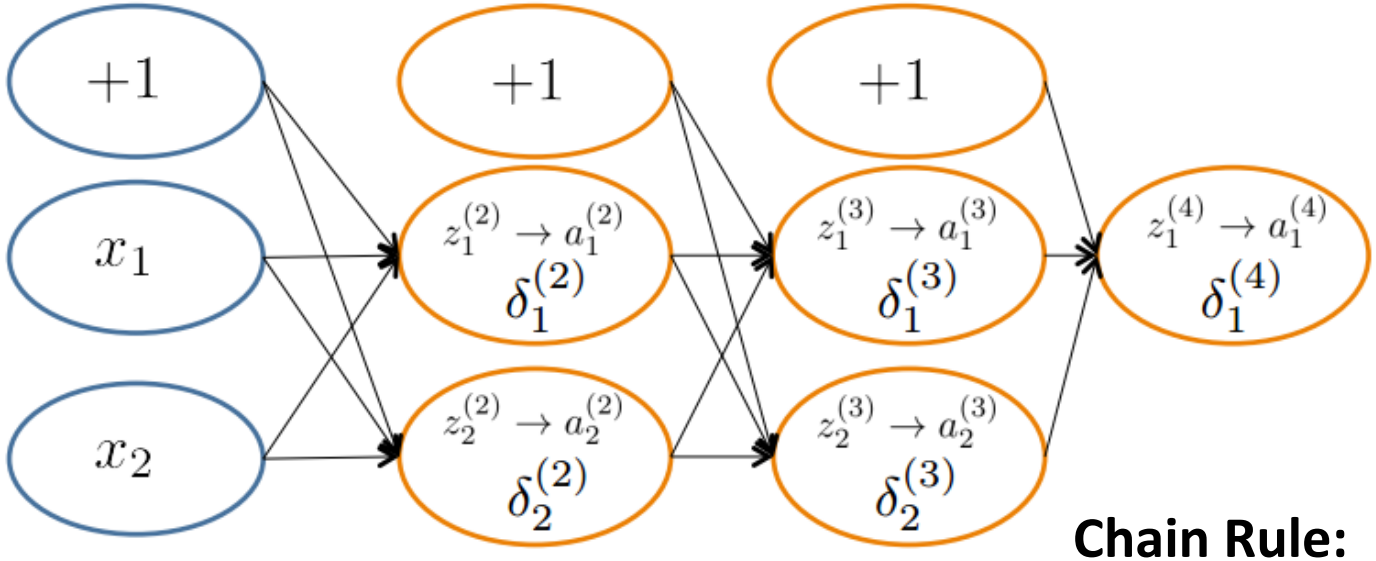
$\delta_j^{(l)}$ = "error" of node $j$ in layer $l$

Formally, $\delta_j^{(l)} = \dfrac{\partial}{\partial z_j^{(l)}} \text{cost}(\mathbf{x}_i)$

where $\text{cost}(\mathbf{x}_i) = y_i \log h_\Theta(\mathbf{x}_i) + (1 - y_i) \log(1 - h_\Theta(\mathbf{x}_i))$

http://ufldl.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/

# Neural Net Training: Backpropagation




**Chain Rule:**

$$\frac{d}{dt} f(g(t)) = f'(g(t))g'(t) = \frac{df}{dg} \cdot \frac{dg}{dt}$$

$\delta_j^{(l)}$ = "error" of node $j$ in layer $l$

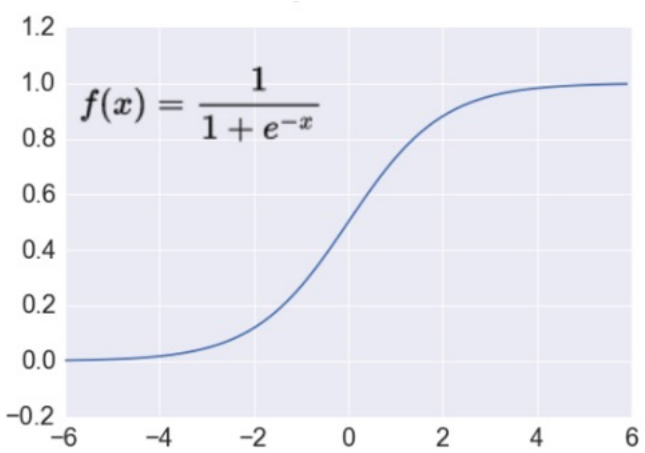
Formally, $\delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} \text{cost}(\mathbf{x}_i)$

Just one cost function, many different cost functions possible

where $\text{cost}(\mathbf{x}_i) = y_i \log h_\Theta(\mathbf{x}_i) + (1 - y_i) \log(1 - h_\Theta(\mathbf{x}_i))$

http://ufldl.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/

# Activation Functions: Sigmoid




*http://adilmoujahid.com/images/activation.png*

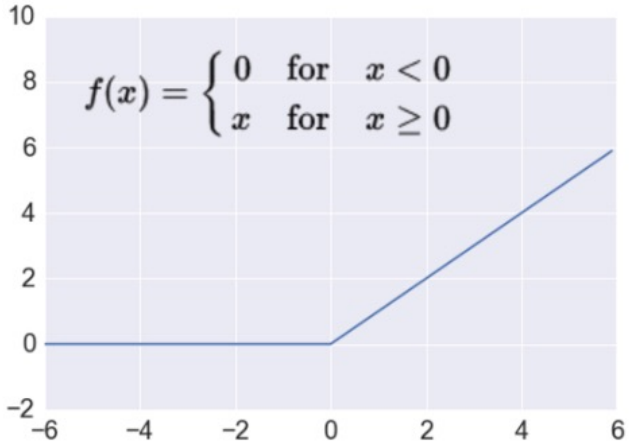
Takes a real-valued number and “squashes” it into range between 0 and 1.

$$R^n \rightarrow [0,1]$$

\+ Nice interpretation as the **firing rate** of a neuron
- 0 = not firing at all
- 1 = fully firing

\- Sigmoid neurons **saturate** and **kill gradients**, thus NN will barely learn
- when the neuron’s activation are 0 or 1 (saturate)
  - gradient at these regions almost zero
  - almost no signal will flow to its weights
  - if initial weights are too large then most neurons would saturate

# Activation Functions: ReLU




*http://adilmoujahid.com/images/activation.png*

Takes a real-valued number and thresholds it at zero $\mathrm{f}(x) = \max(0, x)$

$$R^n \rightarrow R_+^n$$

Most Networks use ReLU for internal units nowadays

Trains much **faster**

- accelerates the convergence of gradient descent
- due to linear, non-saturating form

Less expensive operations

- compared to sigmoid/tanh (exponentials etc.)
- implemented by simply thresholding a matrix at zero

More **expressive**

Prevents the **gradient vanishing problem**

# Learning to Rank

# Learning to Rank (LTR)

*"... the task to automatically construct a ranking model using training data, such that the model can sort new objects according to their degrees of relevance, preference, or importance."*

- Liu [2009]

L2R models represent a rankable item—e.g., a document—given some context—e.g., a user-issued query—as a numerical 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 relevant items are scored higher.

Tie-Yan Liu. Learning to rank for information retrieval. Foundation and Trends in Information Retrieval, 2009.

# Features

Traditional L2R models employ hand-crafted features that encode IR insights

They can often be categorized as:

Query-independent or static features
e.g., incoming link count and document length

Query-dependent or dynamic features
e.g., BM25

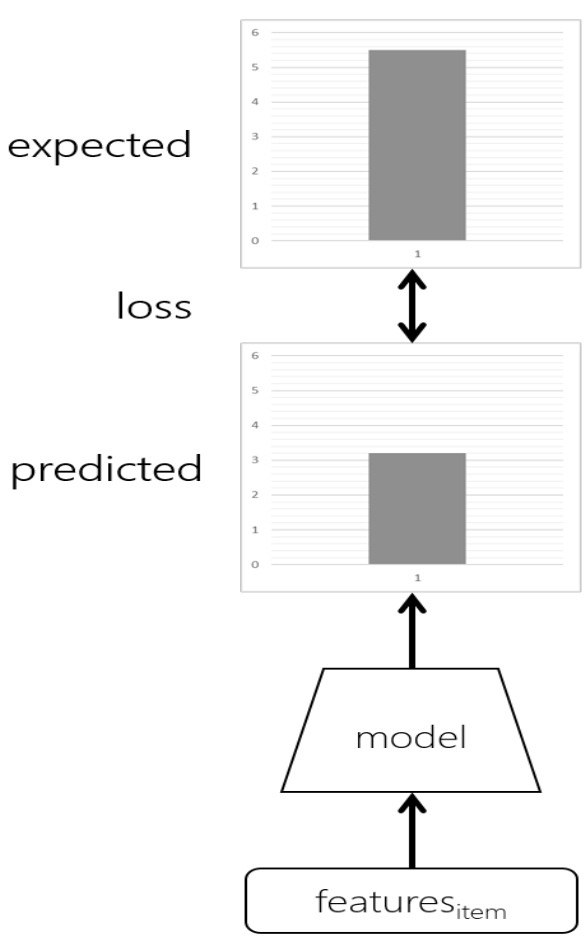
Query-level features
e.g., query length

# Features

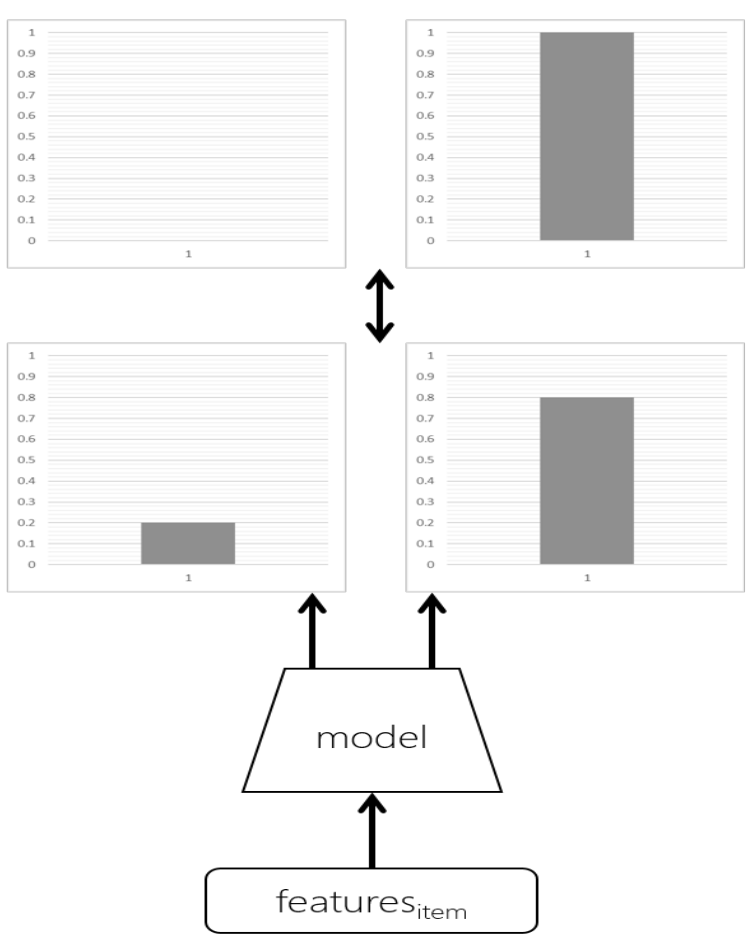
| ID | Feature description | Category |
|---|---|---|
| 1 | $\sum_{q_i \in q \cap d} c(q_i, d)$ in body | Q-D |
| 2 | $\sum_{q_i \in q \cap d} c(q_i, d)$ in anchor | Q-D |
| 3 | $\sum_{q_i \in q \cap d} c(q_i, d)$ in title | Q-D |
| 4 | $\sum_{q_i \in q \cap d} c(q_i, d)$ in URL | Q-D |
| 5 | $\sum_{q_i \in q \cap d} c(q_i, d)$ in whole document | Q-D |
| 6 | $\sum_{q_i \in q} idf(q_i)$ in body | Q |
| 7 | $\sum_{q_i \in q} idf(q_i)$ in anchor | Q |
| 8 | $\sum_{q_i \in q} idf(q_i)$ in title | Q |
| 9 | $\sum_{q_i \in q} idf(q_i)$ in URL | Q |
| 10 | $\sum_{q_i \in q} idf(q_i)$ in whole document | Q |
| 11 | $\sum_{q_i \in q \cap d} c(q_i, d) \cdot idf(q_i)$ in body | Q-D |
| 12 | $\sum_{q_i \in q \cap d} c(q_i, d) \cdot idf(q_i)$ in anchor | Q-D |
| 13 | $\sum_{q_i \in q \cap d} c(q_i, d) \cdot idf(q_i)$ in title | Q-D |
| 14 | $\sum_{q_i \in q \cap d} c(q_i, d) \cdot idf(q_i)$ in URL | Q-D |
| 15 | $\sum_{q_i \in q \cap d} c(q_i, d) \cdot idf(q_i)$ in whole document | Q-D |
| 16 | $\|d\|$ of body | D |
| 17 | $\|d\|$ of anchor | D |
| 18 | $\|d\|$ of title | D |
| 19 | $\|d\|$ of URL | D |
| 20 | $\|d\|$ of whole document | D |
| 21 | BM25 of body | Q-D |
| 22 | BM25 of anchor | Q-D |
| 23 | BM25 of title | Q-D |
| 24 | BM25 of URL | Q-D |
| 25 | BM25 of whole document | Q-D |
| 26 | LMIR.ABS of body | Q-D |
| 27 | LMIR.ABS of anchor | Q-D |
| 28 | LMIR.ABS of title | Q-D |
| 29 | LMIR.ABS of URL | Q-D |
| 30 | LMIR.ABS of whole document | Q-D |
| 31 | LMIR.DIR of body | Q-D |
| 32 | LMIR.DIR of anchor | Q-D |
| 33 | LMIR.DIR of title | Q-D |
| 34 | LMIR.DIR of URL | Q-D |
| 35 | LMIR.DIR of whole document | Q-D |
| 36 | LMIR.JM of body | Q-D |
| 37 | LMIR.JM of anchor | Q-D |
| 38 | LMIR.JM of title | Q-D |
| 39 | LMIR.JM of URL | Q-D |
| 40 | LMIR.JM of whole document | Q-D |
| 41 | Sitemap based term propagation | Q-D |
| 42 | Sitemap based score propagation | Q-D |
| 43 | Hyperlink based score propagation: weighted in-link | Q-D |
| 44 | Hyperlink based score propagation: weighted out-link | Q-D |
| 45 | Hyperlink based score propagation: uniform out-link | Q-D |
| 46 | Hyperlink based propagation: weighted in-link | Q-D |
| 47 | Hyperlink based feature propagation: weighted out-link | Q-D |
| 48 | Hyperlink based feature propagation: uniform out-link | Q-D |
| 49 | HITS authority | Q-D |
| 50 | HITS hub | Q-D |
| 51 | PageRank | D |
| 52 | HostRank | D |
| 53 | Topical PageRank | Q-D |
| 54 | Topical HITS authority | Q-D |
| 55 | Topical HITS hub | Q-D |
| 56 | Inlink number | D |
| 57 | Outlink number | D |
| 58 | Number of slash in URL | D |
| 59 | Length of URL | D |
| 60 | Number of child page | D |
| 61 | BM25 of extracted title | Q-D |
| 62 | LMIR.ABS of extracted title | Q-D |
| 63 | LMIR.DIR of extracted title | Q-D |
| 64 | LMIR.JM of extracted title | Q-D |

Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. LETOR: A Benchmark Collection for Research on Learning to Rank for Information Retrieval, Information Retrieval Journal, 2010

# ML models for other tasks




regression

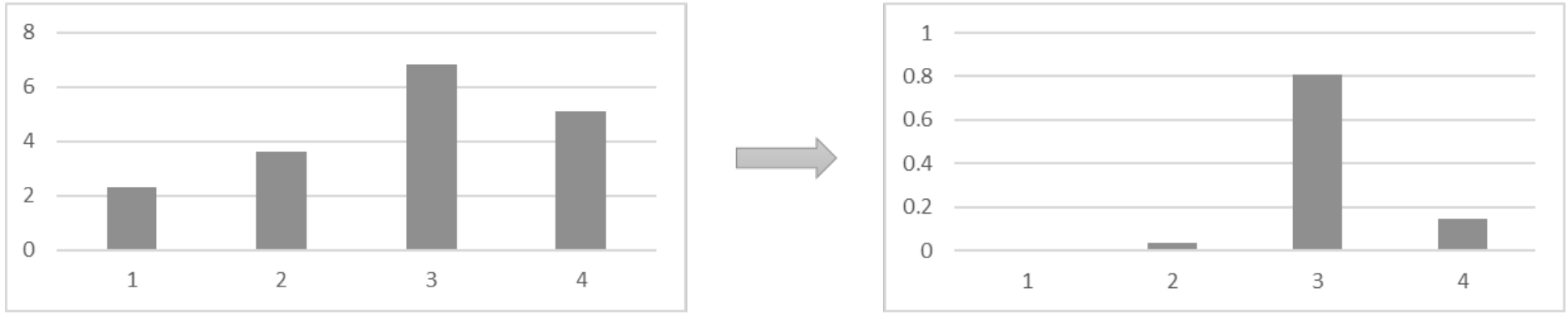



classification

# Background: The softmax function

In classification models, the softmax function is popularly used to normalize the output scores of the machine learning model across all the classes

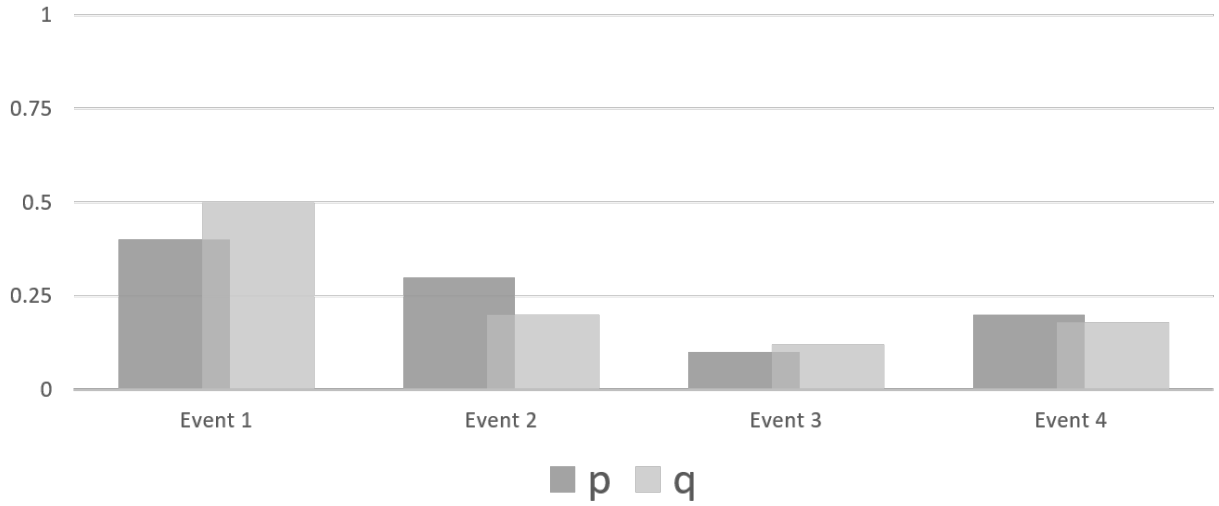
$$p(z_i) = \frac{e^{\gamma z_i}}{\sum_{z \in Z} e^{\gamma z}} \qquad (\gamma \text{ is a constant})$$



# Background: Cross entropy

The cross entropy between two probability distributions $p$ and $q$ over a discrete set of events is given by,

$$CE(p,q) = -\sum_i p_i \log(q_i)$$



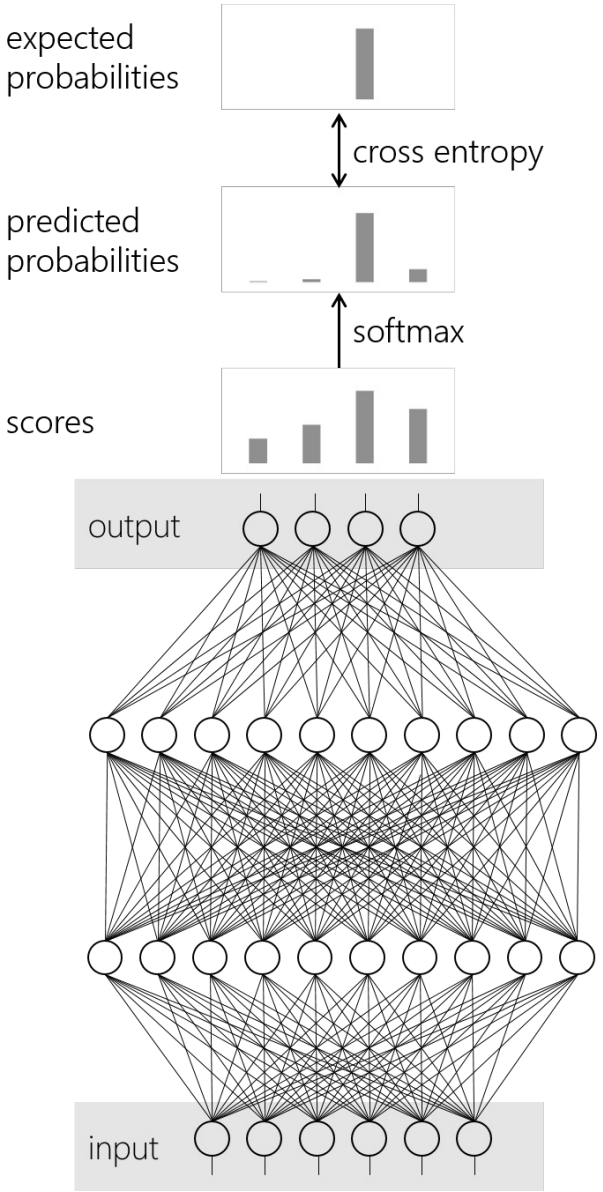

If $p_{correct} = 1$and $p_i = 0$ for all other values of $i$ then,

$$CE(p,q) = -\log(q_{correct})$$

# Background:
# Cross entropy with softmax loss

Cross entropy with softmax is a popular loss function for classification

$$\mathcal{L}_{\text{CE}} = -log\Big(\frac{e^{\gamma z_{correct}}}{\sum_{z \in Z} e^{\gamma z}}\Big)$$



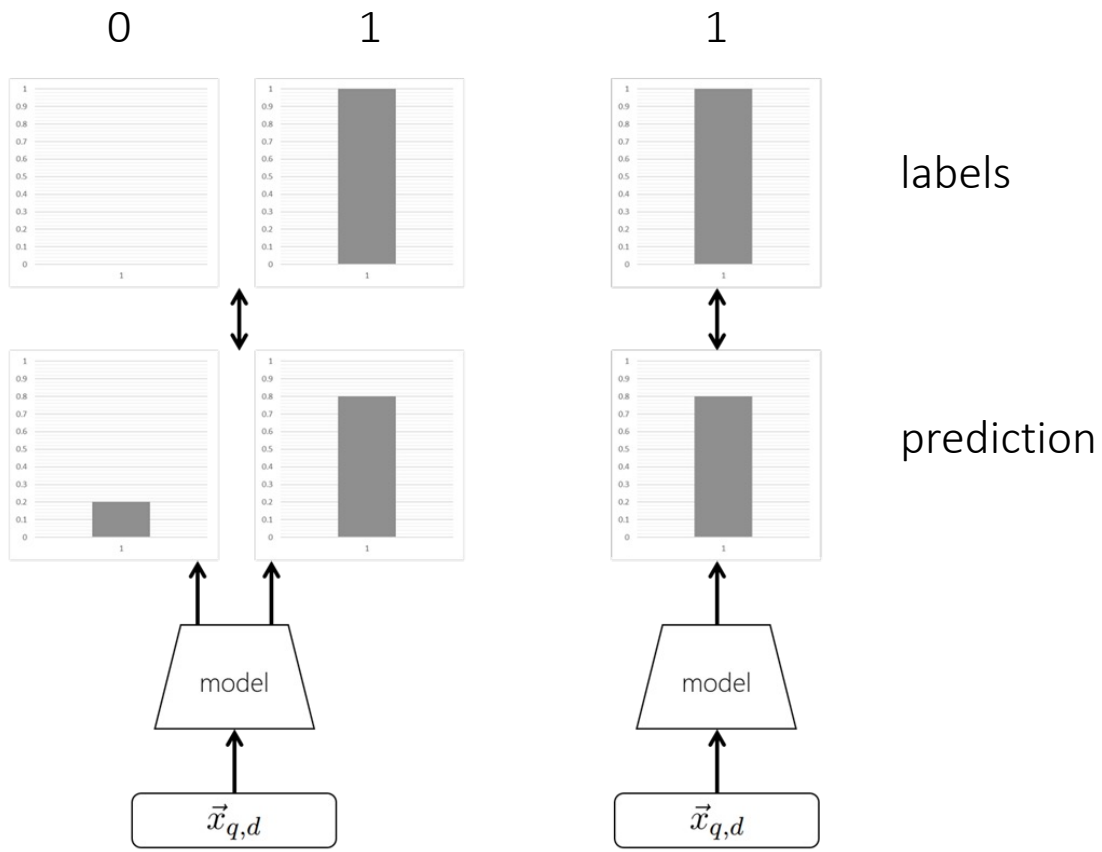
# Pointwise objectives

Regression loss

Given $\langle q, d \rangle$ predict the value of $y_{q,d}$

e.g., square loss for binary or categorical labels,

$$\mathcal{L}_{Squared} = \|y_{q,d} - f(\vec{x}_{q,d})\|^2$$





Norbert Fuhr. Optimum polynomial retrieval functions based on the probability ranking principle. ACM TOIS, 1989.

David Cossock and Tong Zhang. Subset ranking using regression. In COLT, 2006.

# Pointwise objectives

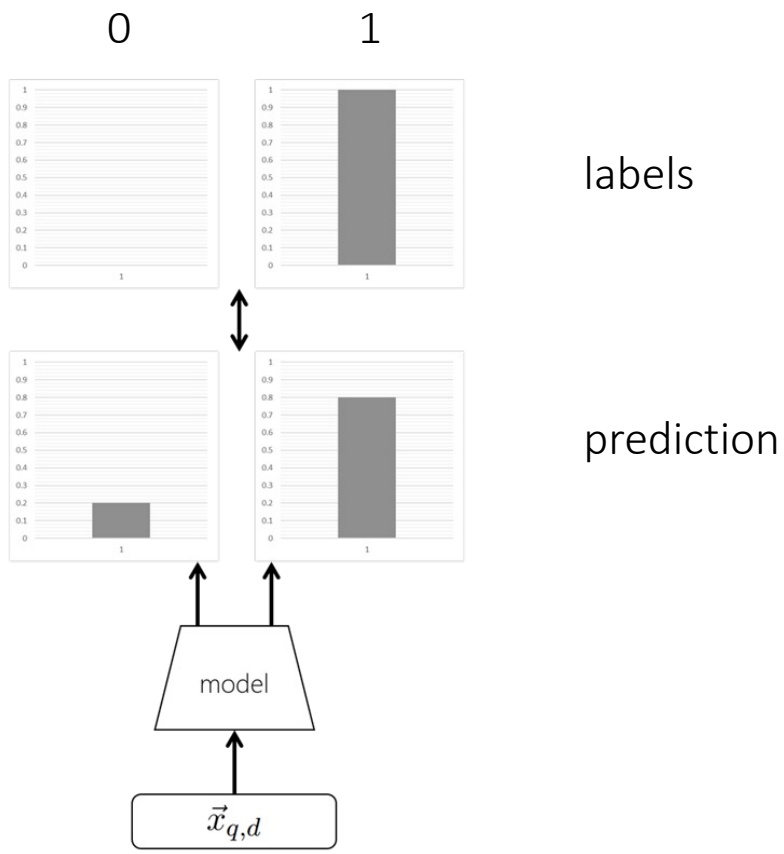
## Classification loss

Given $\langle q, d \rangle$ predict the *class* $y_{q,d}$

e.g., cross-entropy with softmax over categorical labels $Y$ [Li et al., 2008],

$$\mathcal{L}_{\text{CE}}(q, d, y_{q,d}) = -log\Big(p(y_{q,d}|q, d)\Big) = -log\Big(\frac{e^{\gamma \cdot s_{y_{q,d}}}}{\sum_{y \in Y} e^{\gamma \cdot s_y}}\Big)$$

where, $s_{y_{q,d}}$ is the model's score for label $y_{q,d}$




Ping Li, Qiang Wu, and Christopher J Burges. Mcrank: Learning to rank using multiple classification and gradient boosting. In NIPS, 2008.

# Pairwise objectives

Pairwise loss minimizes the average number of inversions in ranking—i.e., $d_i \succ d_j$ w.r.t. $q$ but $d_j$ is ranked higher than $d_i$

Given $\langle q, d_i, d_j \rangle$, predict the more relevant document

For $\langle q, d_i \rangle$ and $\langle q, d_j \rangle$,

Feature vectors: $\vec{x}_i$ and $\vec{x}_j$

Model scores: $s_i = f(\vec{x}_i)$ and $s_j = f(\vec{x}_j)$

Pairwise loss generally has the following form [Chen et al., 2009],

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

where, $\phi$ can be,

- Hinge function $\phi(z) = max(0, 1 - z)$ [Herbrich et al., 2000]
- Exponential function $\phi(z) = e^{-z}$ [Freund et al., 2003]
- Logistic function $\phi(z) = log(1 + e^{-z})$ [Burges et al., 2005]
- Others…


Wei Chen, Tie-Yan Liu, Yanyan Lan, Zhi-Ming Ma, and Hang Li. Ranking measures and loss functions in learning to rank. In NIPS, 2009.

Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Large margin rank boundaries for ordinal regression. 2000.

Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. In JMLR, 2003.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In ICML, 2005.

# Pairwise objectives

## RankNet loss

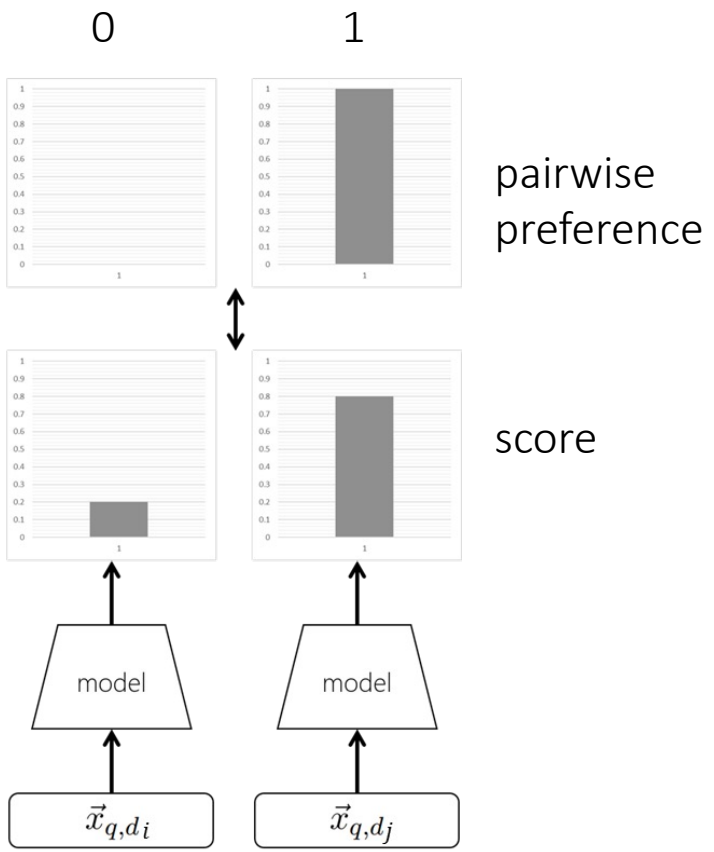
Pairwise loss function proposed by Burges et al. [2005]—an industry favourite [Burges, 2015]

Predicted probabilities: $p_{ij} = p(s_i > s_j) \equiv \frac{e^{\gamma . s_i}}{e^{\gamma . s_i} + e^{\gamma . s_j}} = \frac{1}{1 + e^{-\gamma . (s_i - s_j)}}$

Desired probabilities: $\bar{p}_{ij} = 1$ and $\bar{p}_{ji} = 0$

Computing cross-entropy between $p$ and $\bar{p}$

$$\mathcal{L}_{RankNet} = -\bar{p}_{ij} . log(p_{ij}) - \bar{p}_{ji} . log(p_{ji}) = -log(p_{ij}) = log\left(1 + e^{-\gamma . (s_i - s_j)}\right)$$





Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In ICML, 2005.

Chris Burges. RankNet: A ranking retrospective. https://www.microsoft.com/en-us/research/blog/ranknet-a-ranking-retrospective/. 2015.

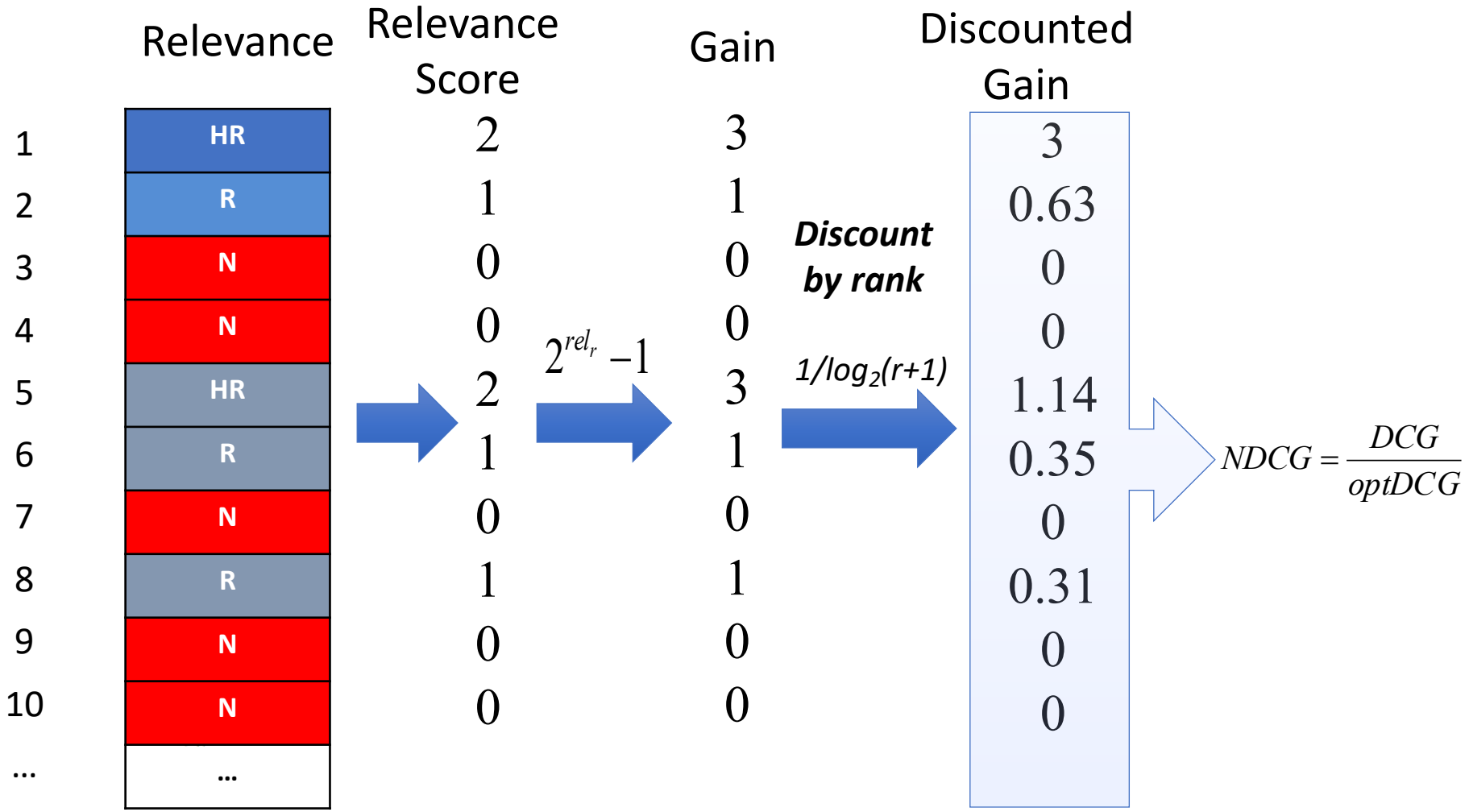
# LISTWISE OBJECTIVES

Optimize for a metric that is assumed to be evaluating user satisfaction from the ranking

[Burges, 2010]

Christopher JC Burges. From ranknet to lambdarank to lambdamart: An overview. Learning, 2010.

# Evaluating Ranking Quality: Normalized Discounted Cumulative Gain (NDCG)



# LISTWISE OBJECTIVES

Blue: relevant  Gray: non-relevant

NDCG higher for left but pairwise errors less for right

Due to strong position-based discounting in IR measures, errors at higher ranks are much more problematic than at lower ranks

But listwise metrics are non-continuous and non-differentiable

[Burges, 2010]

Christopher JC Burges. From ranknet to lambdarank to lambdamart: An overview. Learning, 2010.

# Listwise objectives

Burges et al. [2006] make two observations:

1. To train a model we don't need the costs themselves, only the gradients (of the costs w.r.t model scores)
2. It is desired that the gradient be bigger for pairs of documents that produces a bigger impact in NDCG by swapping positions

## LambdaRank loss

Multiply actual gradients with the change in NDCG by swapping the rank positions of the two documents

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

Christopher JC Burges, Robert Ragno, and Quoc Viet Le. Learning to rank with nonsmooth cost functions. In NIPS, 2006.

# Listwise objectives

According to the Luce model [Luce, 2005], given four items $\{d_1, d_2, d_3, d_4\}$ the probability of observing a particular rank-order, say $[d_2, d_1, d_4, d_3]$, is given by:

$$p(\pi|s) = \frac{\phi(s_2)}{\phi(s_1) + \phi(s_2) + \phi(s_3) + \phi(s_4)} \cdot \frac{\phi(s_1)}{\phi(s_1) + \phi(s_3) + \phi(s_4)} \cdot \frac{\phi(s_4)}{\phi(s_3) + \phi(s_4)}$$

where, $\pi$ is a particular permutation and $\phi$ is a transformation (e.g., linear, exponential, or sigmoid) over the score $s_i$ corresponding to item $d_i$

## ListNet loss

Cao et al. [2007] propose to compute the probability distribution over all possible permutations based on model score and ground-truth labels. The loss is then given by the K-L divergence between these two distributions.

This is computationally very costly, computing permutations of only the top-K items makes it slightly less prohibitive.


R Duncan Luce. Individual choice behavior. 1959.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In ICML, 2007.

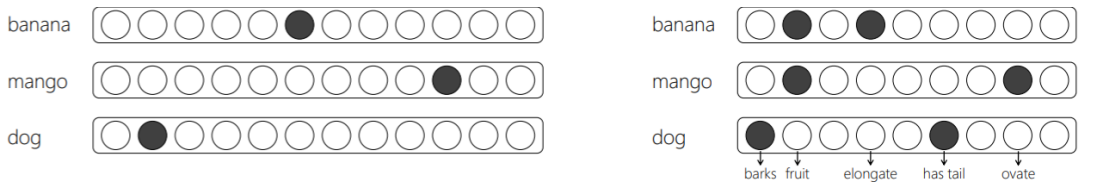
Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In ICML, 2008.

# Term embeddings for IR

# Types of vector representations

**Local (or one-hot) representation**
Every term in vocabulary T is represented by a binary vector of length |T|, where one position in the vector is set to one and the rest to zero

**Distributed representation**
Every term in vocabulary T is represented by a real-valued vector of length k. The vector dimensions may be *observed* (e.g., hand-crafted features) or *latent* (e.g., embedding dimensions).




(a) Local representation (b) Distributed representation

**Figure 3.1:** Under local representations the terms "banana", "mango", and "dog" are distinct items. But distributed vector representations may recognize that "banana" and "mango" are both fruits, but "dog" is different.

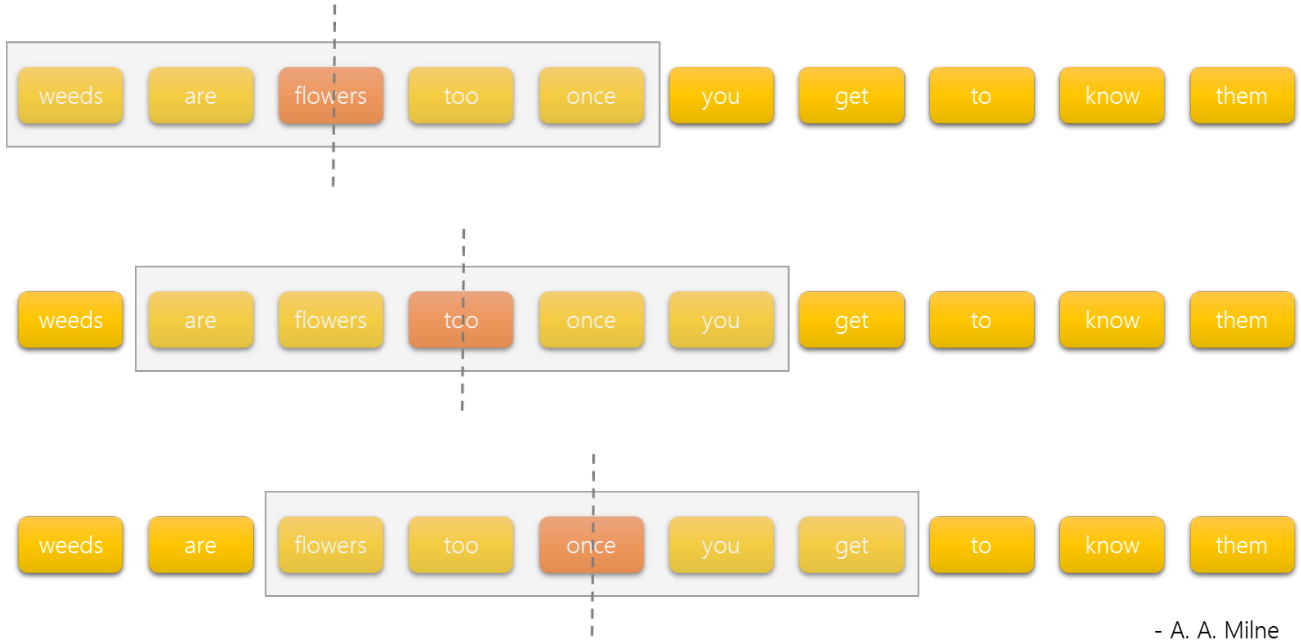
# Word2vec

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

Predict neighbor (context word) $t_{i+j}$ given term $t_i$

Trained over very large corpus

One of the simplest and earliest models

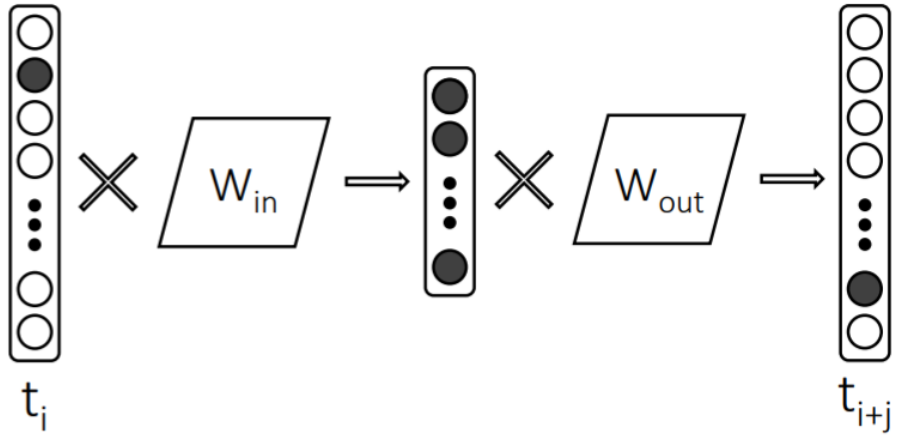



- A. A. Milne

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In NIPS, 2013.

# Skip-gram Model

Predict neighbor (context word) $t_{i+j}$ given term $t_i$




(a) Skip-gram

# The Skip-gram Objective

**Negative sampling**: Randomly sample negative context words from the corpus

Maximize the + label for the pairs from the positive training data, and the – label for the pairs sample from the negative data

$$\sum_{(t,c)\in+} logP(+|t,c) + \sum_{(t,c)\in-} logP(-|t,c)$$

c is the positive/negative context term we would like to predict, given $t$

# Retrieval Using Vector Representations

# Retrieval using vector representations

Generate vector representation of query

Generate vector representation of document

Estimate relevance from q-d vectors

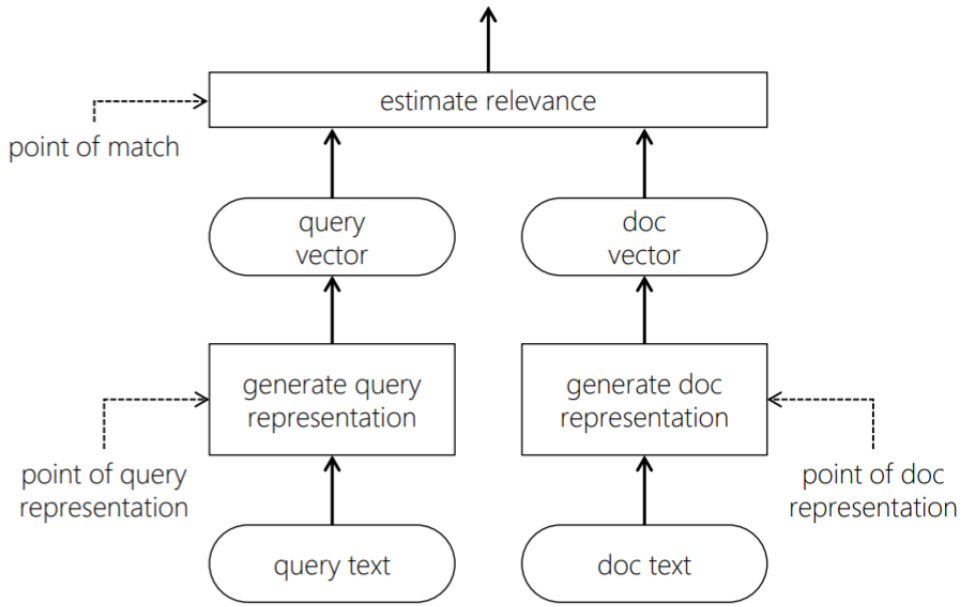
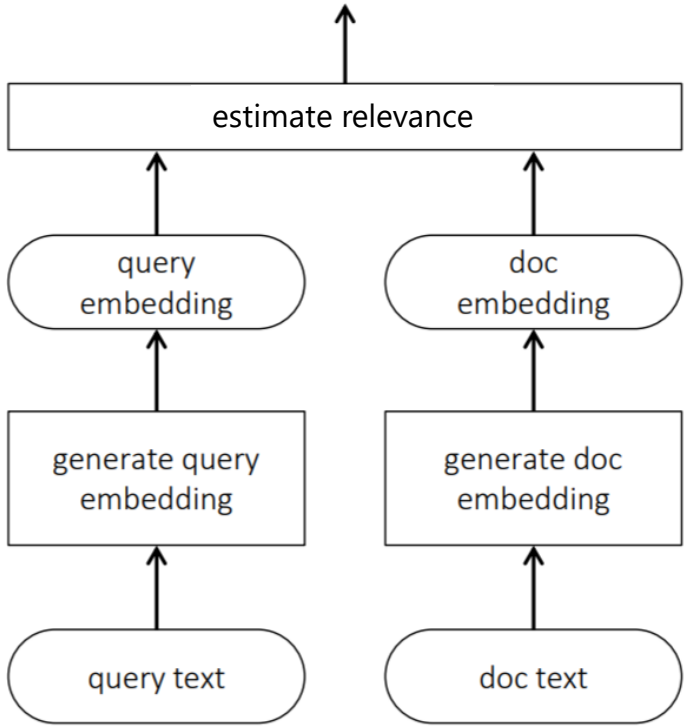
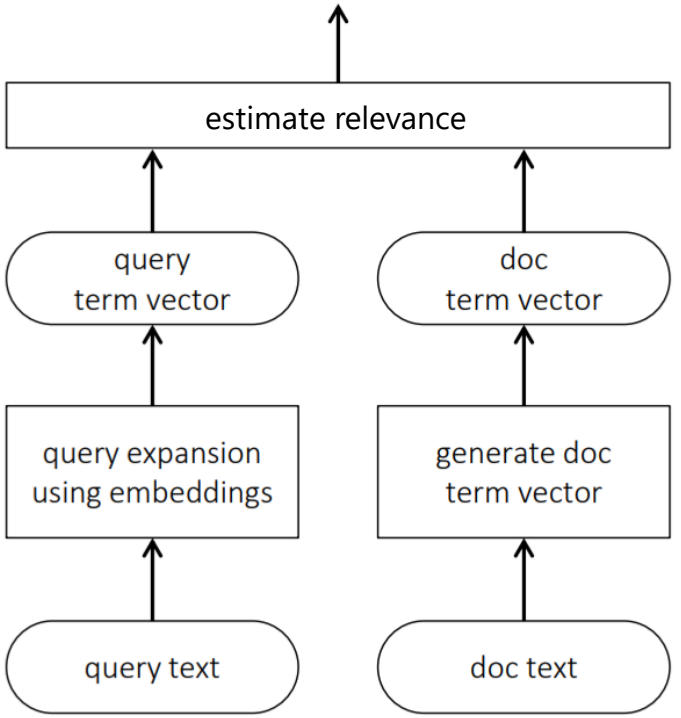



**Figure 2.3:** Document ranking typically involves a query and a document representation steps, followed by a matching stage. Neural models can be useful either for generating good representations or in estimating relevance, or both.

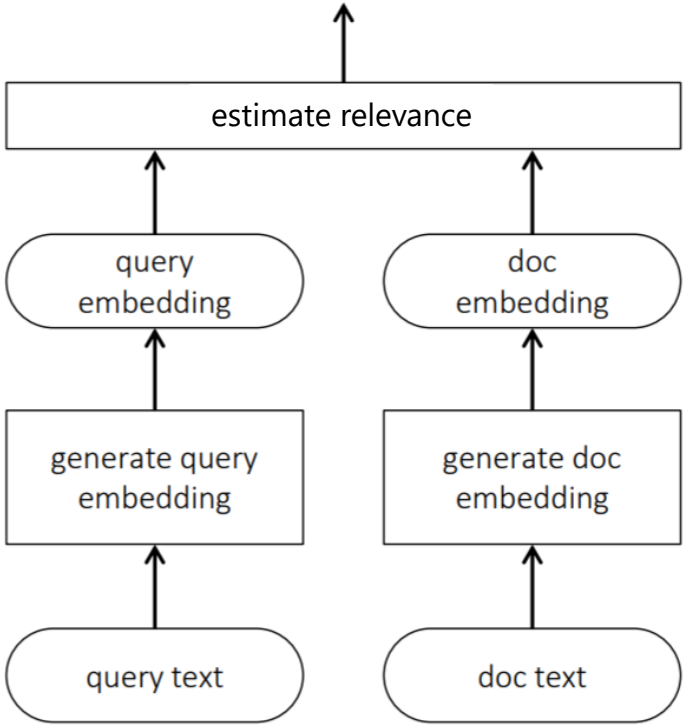
# Popular approaches to incorporating term embeddings for matching




Compare query and document directly in the embedding space




Use embeddings to generate suitable query expansions




Compare query and document directly in the embedding space

E.g.,

Generalized Language Model [Ganguly et al., 2015]

Neural Translation Language Model [Zuccon et al., 2015]

Average term embeddings [Le and Mikolov, 2014, Nalisnick et al., 2016, Zamani and Croft, 2016, and others]

Word mover's distance [Kusner et al., 2015, Guo et al., 2016]

# Average Term Embeddings

Q-D relevance estimated by computing cosine similarity between centroid of q and d term embeddings

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

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

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

Eric Nalisnick, Bhaskar Mitra, Nick Craswell, and Rich Caruana. Improving document ranking with dual word embeddings. In WWW, 2016.
Bhaskar Mitra, Eric Nalisnick, Nick Craswell, and Rich Caruana. A dual embedding space model for document ranking. arXiv preprint arXiv:1602.01137, 2016.

# A tale of two queries

"pekarovic land company"

Hard to learn good representation for the rare term *pekarovic*

But easy to estimate relevance based on count of exact term matches of *pekarovic* in the document

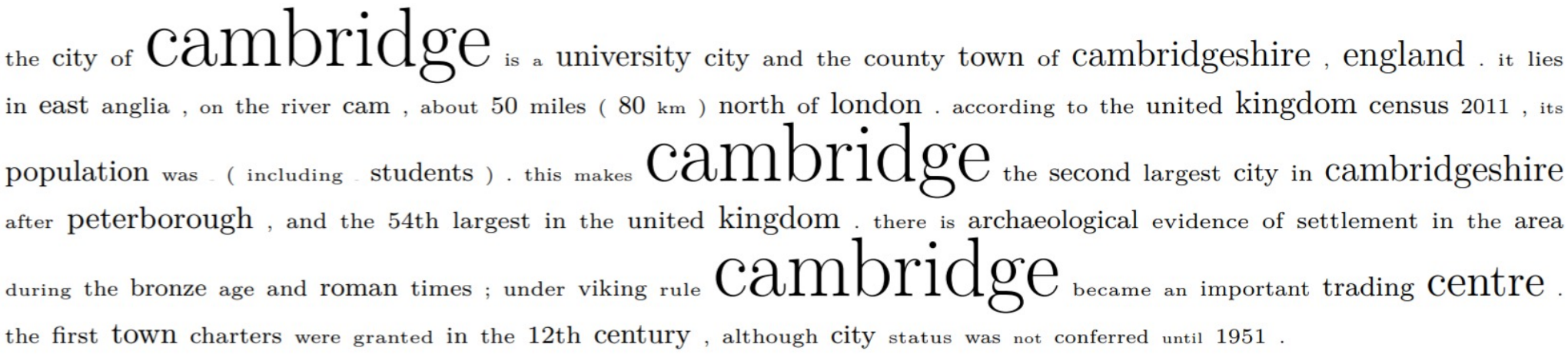
"what channel are the seahawks on today"

Target document likely contains *ESPN* or *sky sports* instead of *channel*

The terms *ESPN* and *channel* can be compared in a term embedding space

Matching in the term space is necessary to handle rare terms. Matching in the latent embedding space can provide additional evidence of relevance. Best performance is often achieved by combining matching in both vector spaces.

# Query: Cambridge (Font size is a function of term-term cosine similarity)




**(a)** Passage about the city of Cambridge

Besides the term “Cambridge”, other related terms (e.g., “university”, “town”, “population”, and “England”) contribute to the relevance of the passage

Bhaskar Mitra, Eric Nalisnick, Nick Craswell, and Rich Caruana. A dual embedding space model for document ranking. arXiv preprint arXiv:1602.01137, 2016.

# Query: Cambridge (Font size is a function of term-term cosine similarity)

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 .

**(b)** Passage about the city of Oxford

However, the same terms may also make a passage about Oxford look somewhat relevant to the query “Cambridge”

Bhaskar Mitra, Eric Nalisnick, Nick Craswell, and Rich Caruana. A dual embedding space model for document ranking. arXiv preprint arXiv:1602.01137, 2016.

# Query: Cambridge (Font size is a function of term-term cosine similarity)

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 .

**(c)** Passage about giraffes

A passage about giraffes, however, obviously looks non-relevant in the embedding space…

Bhaskar Mitra, Eric Nalisnick, Nick Craswell, and Rich Caruana. A dual embedding space model for document ranking. arXiv preprint arXiv:1602.01137, 2016.

# Query: Cambridge (Font size is a function of term-term cosine similarity)

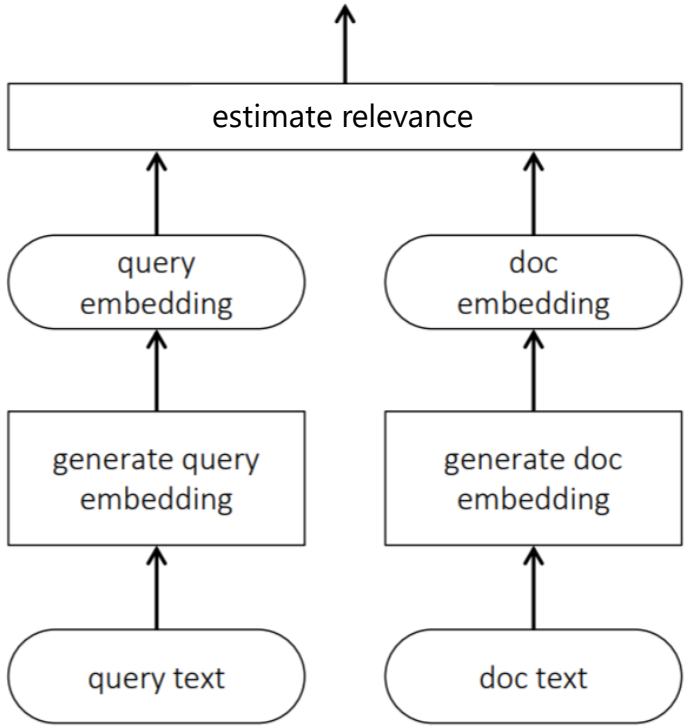
the cambridge ( giraffa camelopardalis ) is an african _ ungulate mammal , the tallest living terrestrial animal and the largest ruminant . its species name refers to its _ shape and its _ colouring . its chief distinguishing characteristics are its extremely long neck and legs , its _ _ , and its distinctive 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 .

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

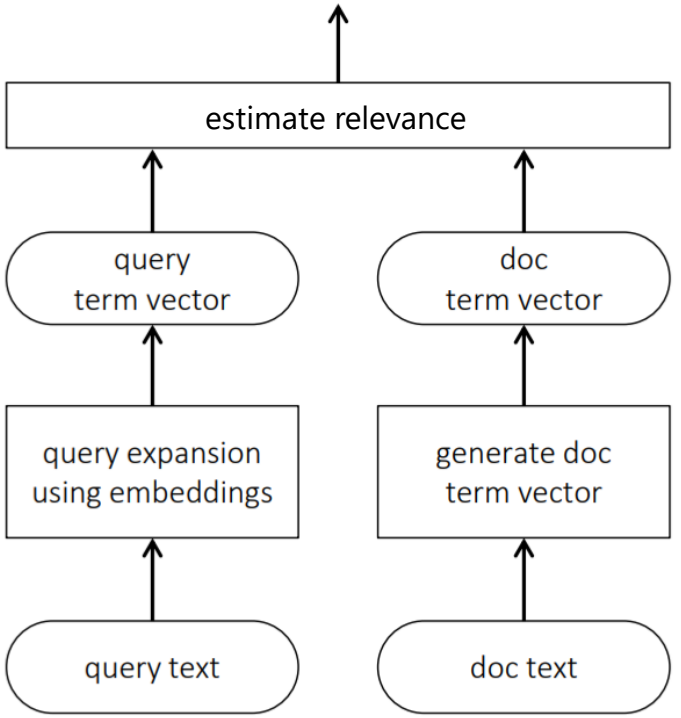
But the embedding based matching model is more robust to the same passage when “giraffe” is replaced by “Cambridge”—a trick that would fool exact term based IR models. In a sense, the embedding based model ranks this passage low because Cambridge is not "an African even-toed ungulate mammal“.

Bhaskar Mitra, Eric Nalisnick, Nick Craswell, and Rich Caruana. A dual embedding space model for document ranking. arXiv preprint arXiv:1602.01137, 2016.

# Popular approaches to incorporating term embeddings for matching




Compare query and document directly in the embedding space



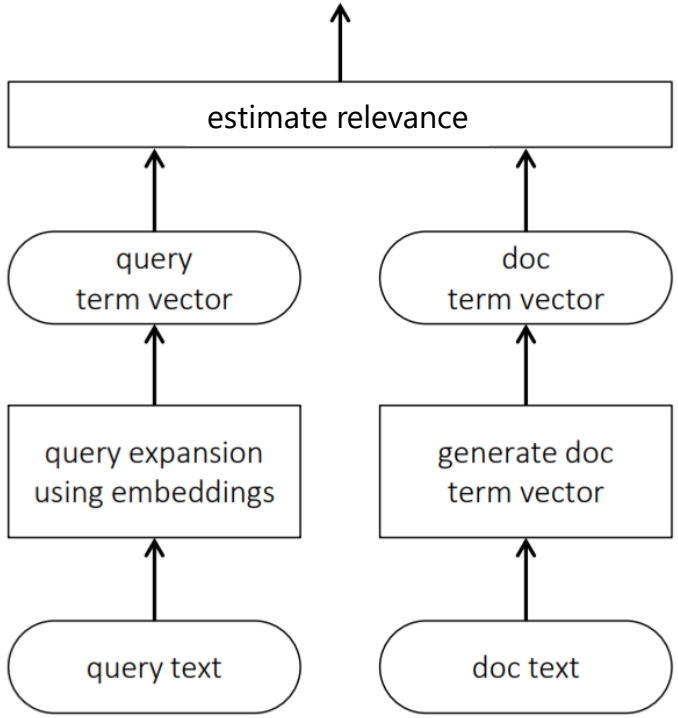

Use embeddings to generate suitable query expansions

# Query expansion using term embeddings

Find good expansion terms based on nearness in the embedding space

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

Better retrieval performance when combined with pseudo-relevance feedback (PRF) [Zamani and Croft, 2016] and if we learn query specific term embeddings [Diaz et al., 2016]




Use embeddings to generate suitable query expansions


Fernando Diaz, Bhaskar Mitra, and Nick Craswell. Query expansion with locally-trained word embeddings. In ACL, 2016.
Dwaipayan Roy, Debjyoti Paul, Mandar Mitra, and Utpal Garain. Using word embeddings for automatic query expansion. arXiv preprint arXiv:1606.07608, 2016.
Hamed Zamani and W Bruce Croft. Embedding-based query language models. In ICTIR, 2016.