Web Search based on Ranking summary

P. Petsinis, Prof. Ioannis Pitas
Aristotle University of Thessaloniki
pitas@csd.auth.gr
www.aiia.csd.auth.gr
Version 1.1

Artificial Intelligence & Information Analysis Lab

Introduction to: Web Search based on Ranking

- **What is Web, Search engine and Ranking**
- Architecture of Web Search Engine:
  Crawler, Indexer, Query Processor
- Timeline of Ranking at Indexed Pages
- Ranking Algorithms
  - Based on Frequency
    - TF-IDF
  - Based on Graph-Link Analysis
    - PageRank
    - Hits
    - Salsa
    - UsersRank
    - SimRank
  - Neural Networks
    - Learning to rank algorithms Categorization
      - Pointwise(Ranking with Large Margin Principles-SVM)
      - Pairwise(SortNet)
      - Listwise(ListNet)
    - Current Neural Networks for Ranking
      - GEPS
      - MatchPyramid
      - C-DSSM(and its extansion NeuBase)
- Deep Learning Metrics
  - Angular Loss
  - Nearest Neighbors Gaussian Kernels
- References

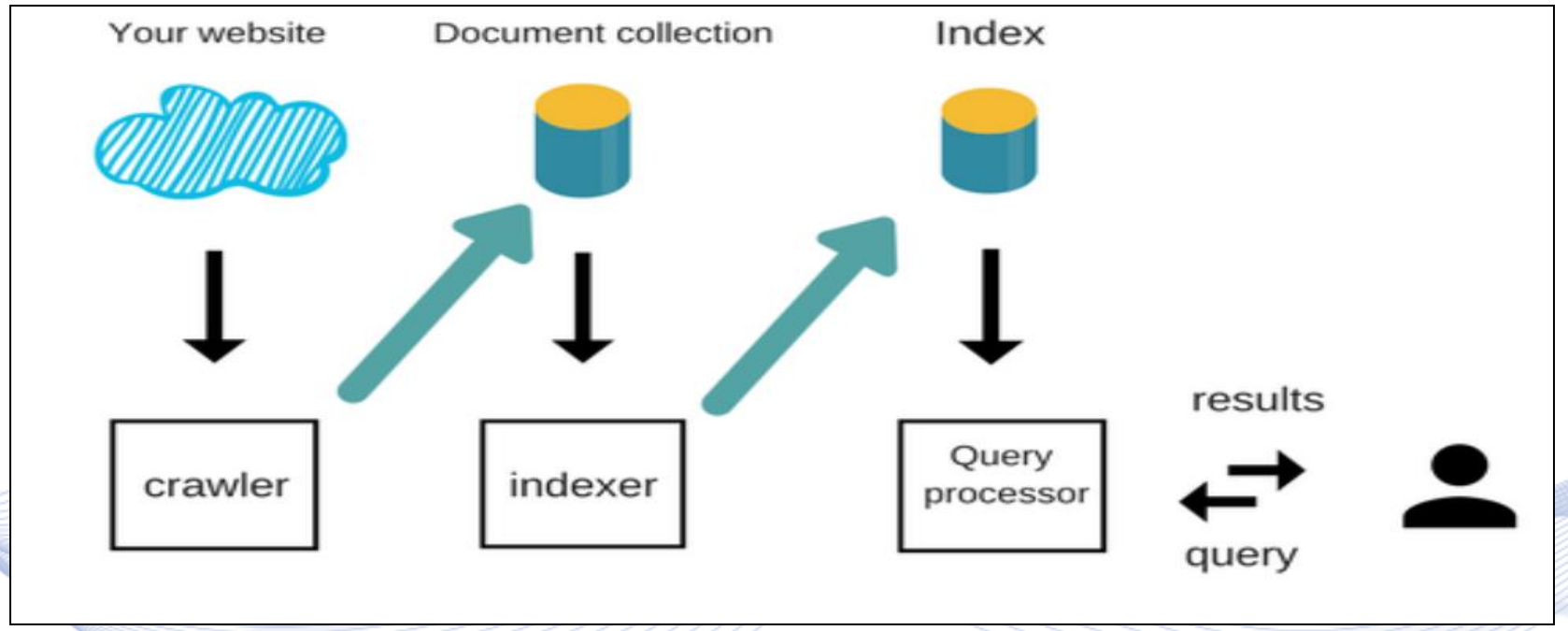
Web Search

As Internet is rapidly gaining popularity these days, web searching has become more important. A collection of pages where all pages are interlinked looks like a Spider's Web, therefore called as **"Web"**. A **search engine** makes use of the combination of textual keywords to display the results. When internet user searches some information using search engine, the result contains all the relevant and irrelevant data. As this data is so vast indexing complete web is impractical, so some filtering should be applied to deliver the quality result. A filter mechanism called as **"Ranking"** is used now days in many popular search engines.

Introduction to: Web Search based on Ranking

- What is Web, Search engine and Ranking
- **Architecture of Web Search Engine: Crawler, Indexer, Query Processor**
- Timeline of Ranking at Indexed Pages
- Ranking Algorithms
  - Based on Frequency
    - TF-IDF
  - Based on Graph-Link Analysis
    - PageRank
    - Hits
    - Salsa
    - UsersRank
    - SimRank
  - Neural Networks
    - Learning to rank algorithms Categorization
      - Pointwise(Ranking with Large Margin Principles-SVM)
      - Pairwise(SortNet)
      - Listwise(ListNet)
    - Current Neural Networks for Ranking
      - GEPS
      - MatchPyramid
      - C-DSSM(and its extansion NeuBase)
- Deep Learning Metrics
  - Angular Loss
  - Nearest Neighbors Gaussian Kernels
- References

Architecture of: Web Search Engine



Architecture of Web Search Engine (from [12])

Crawler-Indexer-Searcher

- **Crawler:**

  A web crawler, also called bot or robot, searches the web to find HTML pages. The web crawler stores these pages unchanged in its search engine's document database.

- **Indexer:**

  Indexer is the software that receives the web pages from the crawler and then organizes the content with such way that allows quick search. The method of organization commonly used is the structure of the inverted index. The indexer is responsible for updating the directory with new contents and changes that need to be made in case of conversion content of a web page. The key used to distribute web pages is usually the URL.

- **Searcher:** Also called as query processor, receives an informative need (a query consisting of keywords), and in conjunction with the inverted index returns the web pages that are considered relevant to the user's query.

Ranking Algorithms

- Based on Frequency:
  - TF-IDF(“Term frequency and inverse document frequency”)

- Based on Graph-Link Analysis**:**
  - PageRank
  - HITS
  - SALSA(Stochastic Approach for Link-Structure Analysis)
  - UsersRank
  - SimRank

- Neural Networks
  - Learning to rank algorithms Categorization
    - Pointwise(Ranking with Large Margin Principles-SVM)
    - Pairwise(SortNet)
    - Listwise(ListNet)
  - Current Neural Networks for Ranking
    - GEPS
    - MatchPyramid
    - C-DSSM(and its extansion NeuBase)

Ranking Algorithms

- **Based on Frequency:**
  - TF-IDF("Term frequency and inverse document frequency")

- Based on Graph-Link Analysis:
  - PageRank
  - HITS
  - SALSA(Stochastic Approach for Link-Structure Analysis)
  - UsersRank
  - SimRank

- Neural Networks
  - Learning to rank algorithms Categorization
    - Pointwise(Ranking with Large Margin Principles-SVM)
    - Pairwise(SortNet)
    - Listwise(ListNet)
  - Current Neural Networks for Ranking
    - GEPS
    - MatchPyramid
    - C-DSSM(and its extansion NeuBase)

TF-IDF

This method is founded on the inspection of the web page text itself and how the words in the query relate within it. **TF-IDF** or **"Term frequency and inverse document frequency"** applies the vector space model with a scoring function where the importance of a term increases if it is repeated often in a document, however, its weight decreases if the same term appears multiple times on numerous documents as it is considered less descriptive:

$$TF\ -\ IDF(\mathbf{t},\boldsymbol{d},D)\ = TF(\mathbf{t},\boldsymbol{d})\times IDF(\mathbf{t},D)\ =\ [1+\log\ ftd\,]\ \times\ [\log(1+\ \frac{N}{dt}\ )]$$

where **t** is the keyword, **d** is the web page, *ftd* the amount of occurrences that keyword t is contained within the web page ***d***, *N* is the entire amount of web pages, and *dt* is the quantity of web pages that include the keyword **t**.

PageRank

- It is the ranking based algorithms which employees hyperlinks on the web. **PageRank** algorithm's results are ordered according to pertinence and importance of that page.

- A rank is calculated for every page on the web by counting total number of votes earned by that page. Using the Markov chain matrix from the vast structure of hyperlinks, static ranking of a web page is formed. Hyperlink from page **A** to page **B** interprets one vote from page **A** to page **B**.

- As the **PageRank** algorithm does not rely on search queries, Rank of a web page is calculated offline and updated after certain duration. It is based on the principle that says when a page contains important incoming links then its outgoing links to other pages are also important.

PageRank

To calculate the Overall page rank previous rank is added to text matching score. The original **PageRank** algorithm is shown below:

$$PR(\mathbf{A}) = (1-d) + d\,\frac{PR(\mathbf{T_1})}{C(\mathbf{T_1})} + \ldots + \frac{PR(\mathbf{T_n})}{C(\mathbf{T_n})}$$

*PR (***A***)* is the Rank of page **A**,
PR ($\mathbf{T_i}$*)* is the PageRank of pages $\mathbf{T_i}$ which link to page **A**,
C($\mathbf{T_i}$*)* is the number of outbound links on page $\mathbf{T_i}$ and
d is a damping factor which can be set between 0 and 1.

The **PageRank** theory holds that even an imaginary surfer who is randomly clicking on links will eventually stop clicking. The probability, at any step, that the person will continue is a damping factor *d.*

PageRank can also be calculated as: $$PR(\mathbf{A}) = \sum_{\mathbf{B} \in B_{\mathbf{A}}} \frac{PR(\mathbf{B})}{|\mathbf{B}|}$$

$B_{\mathbf{A}}$: is the numbers of incoming links of page **A** and |**B**| is the number of outgoing links from **B**.

HITS

HITS identifies good **authorities** and **hubs** for a topic by assigning two numbers to a page: an authority and a hub score. These scores are defined recursively.

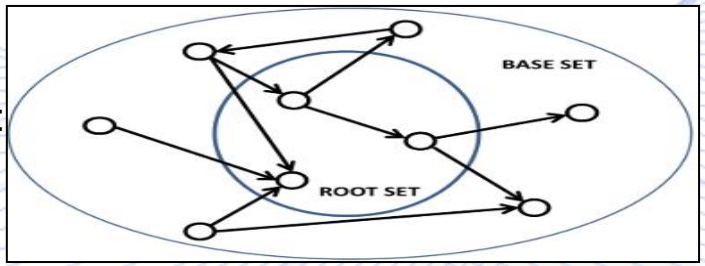
A higher **authority score** occurs if the page is pointed to by pages with high hub scores.

A higher **hub score** occurs if the page points to many pages with high authority scores.

In the HITS algorithm, the **first step** is to retrieve the most relevant pages to the search query.

This set is called the ***root set*** and can be obtained by taking the top pages returned by a text-based search algorithm.

Root and Base Sets (from [12]):



A base *set* is generated by augmenting the root set with all the web pages that are linked from it and some of the pages that link to it.

The web pages in the base set and all hyperlinks among those pages form a focused subgraph.

The HITS computation is performed only on this focused subgraph.

SALSA

The **SALSA** algorithm, or Stochastic Approach for Link-Structure Analysis, combines **Page Rank** and **HITS** by taking a random walk alternating between hubs and authorities. Two random walks are assigned two scores per web page; it is based on the in link and out link that represent the proportion of authority and hub respectively.

The **hub array** is expressed in the below sum:

$$h_{i,j} = \sum_{\{\mathbf{k} \mid (\mathbf{i}_h, \mathbf{k}_a), (\mathbf{j}_h, \mathbf{k}_a) \in G\}} \frac{1}{\deg(\mathbf{i}_h)} \times \frac{1}{\deg(\mathbf{k}_a)}$$

The **authority array** is expressed in the below sum:

$$a_{i,j} = \sum_{\{\mathbf{k} \mid (\mathbf{k}_h, \mathbf{i}_a), (\mathbf{k}_h, \mathbf{j}_a) \in G\}} \frac{1}{\deg(\mathbf{i}_a)} \times \frac{1}{\deg(\mathbf{k}_h)}$$

where web document **h** links to mutual documents **i** and **j** and deg is the amount of hyperlinks connecting to a page.

UsersRank

Bookmark is necessary for some reasons: web searching is dynamic in nature, so even if user finds out the same information after certain amount of time, search engine may produce different results every time. Therefore the link may be lost. Another reason could be it saves the time; instead of searching it again and again user may use it from bookmark's folder.

UsersRank algorithm makes use of these **bookmarks** and produces valuable information for search engines. Here **user** is treated as a core ingredient for making web search more powerful. It believes in the logic that if user is having some links as bookmarked then those links are actually used by someone hence really valuable and gives effective results for web searches.

Main objective of UsersRank Algorithm is to concentrate on the information which is actually referred by number of users thus gives quality search results. Here user is treated as a crawler discovering information using different media and collected information contains a group of URLs visited, gathered and tagged by users.

SimRank

The intuition behind the **SimRank** algorithm is that, in many domains, **similar objects are referenced by similar objects**. More precisely, objects *a* and *b* are considered to be similar if they are pointed from objects *c* and *d*, respectively, and *c* and *d* are themselves similar.

The base case is that objects are maximally similar to themselves .

It is important to note that **SimRank** is a general algorithm that determines only the similarity of structural context. **SimRank** applies to any domain where there are enough relevant relationships between objects to base at least some notion of similarity on relationships.

Obviously, **similarity** of other domain-specific aspects are important as well; these should be combined with relational structural-context similarity for an overall similarity measure.

For example, for **Web Pages, SimRank** can be combined with traditional textual similarity.

Similarities can be combined with the similarity scores that are computed based on preference patterns, in order to produce an overall similarity measure.

Ranking Algorithms

- Based on Frequency:
  - TF-IDF("Term frequency and inverse document frequency")

- Based on Graph-Link Analysis:
  - PageRank
  - HITS
  - SALSA(Stochastic Approach for Link-Structure Analysis)
  - UsersRank
  - SimRank

- **Neural Networks**
  - Learning to rank algorithms Categorization
    - Pointwise(Ranking with Large Margin Principles-SVM)
    - Pairwise(SortNet)
    - Listwise(ListNet)
  - Current Neural Networks for Ranking
    - GERS
    - MatchPyramid
    - NeuBase

Ranking Algorithms

- Based on Frequency:
  - TF-IDF("Term frequency and inverse document frequency")

- Based on Graph-Link Analysis:
  - PageRank
  - HITS
  - SALSA(Stochastic Approach for Link-Structure Analysis)
  - UsersRank
  - SimRank

- **Neural Networks**
  - **Learning to rank algorithms Categorization**
    - Pointwise(Ranking with Large Margin Principles-SVM)
    - Pairwise(SortNet)
    - Listwise(ListNet)
  - Current Neural Networks for Ranking
    - GERS
    - MatchPyramid
    - NeuBase

Pointwise

The **pointwise** method considers every query web page couple within a training set is assigned to a quantitative or ordinal value. It assumes an individual document as its only learning input. **Pointwise** is represented as a **regression** model where, provided a unique query document couple, it predicts its rating.

SVM

Shashua and Levin tried to use **SVM** to learn model parameter **w** and **thresholds** $b_k (k = 0,...,K-1)$, ***for ordinal regression***. The strategy referred as the *fixed margin strategy*.

Given *n training queries* $\{\mathbf{q}_i\}, i=1..n$, their associated documents $\mathbf{x}^{(i)} = \{\mathbf{x}_j^{(i)}\},\ j=1\ldots m^{(i)}$, *and the corresponding relevance judgments* $\mathbf{y}^{(i)} = \{\mathbf{y}_j^{(i)}\},\ j=1\ldots m^{(i)}$, the learning process is defined below, where the adoption of a linear scoring function is assumed. The constraints basically require every document to be correctly classified into its target ordered category, i.e., for documents in category *k*, $\mathbf{w}^T \mathbf{x}_j^{(i)}$ *should exceed threshold* b_{k-1} but be smaller than threshold b_k, *with certain soft margins:*

$$1-\xi_{j,k-1}^{(i)*} \ \text{and}\ 1-\xi_{j,k}^{(i)}$$

$$\min \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\sum_{j=1}^{m^{(i)}}\sum_{k=1}^{K-1}\left(\xi_{j,k}^{(i)} + \xi_{j,k+1}^{(i)*}\right)$$

$$s.t. \quad \mathbf{w}^T\mathbf{x}_j^{(i)} - b_k \le -1+\xi_{j,k}^{(i)}, \quad if\ y_j^{(i)} = k,$$

$$\mathbf{w}^T\mathbf{x}_j^{(i)} - b_k \ge 1-\xi_{j,k}^{(i)}, \quad if\ y_j^{(i)} = k+1,$$

$$\xi_{j,k}^{(i)} \ge 0,\ \ \xi_{j,k+1}^{(i)*} \ge 0,$$

$$j = 1,...,m^{(i)}, \quad i = 1,...,n, \quad k = 0,...,K-2$$

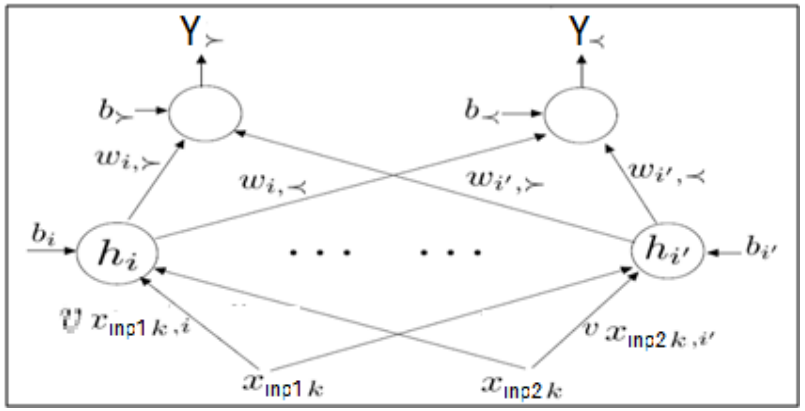
*The margin term 1/2×||**w**||² controls the* complexity of model *w.*

Pairwise

The **pairwise** method evaluates only the relative order between a pair of web pages; it collects documents pairs from the training data on which a label that represents the respective order of the two documents is assigned to each document pair. **Pairwise** is approximated by a **classification problem**. Algorithms take document pairs as instances against a query where the optimization target is the identification of the best document pair preferences.

SortNet:
Learning To Rank by a Neural Preference Function

The CmpNN architecture adopts a weight sharing technique in order to ensure that the reflexivity and the equivalence between $\prec$ and $\succ$ hold.



Example of a CmpNN architecture (from [5])

For each hidden neuron i, a dual neuron i′ exists whose weights are shared with i, according to the following schema:

1. $v\mathbf{x}_{inp1(k),i'} = v\mathbf{x}_{inp2(k),i}$ and $v\mathbf{x}_{inp2(k),i'} = v\mathbf{x}_{inp1(k),i}$ hold, i.e., the weights from $\mathbf{x}_{inp1(k)}$, $\mathbf{x}_{inp2(k)}$ to i are swapped in the connections to i′;
2. $w_{i',\succ} = w_{i,\prec}$ and $w_{i',\prec} = w_{i,\succ}$ hold, i.e., the weights of the connections from the hidden i to the outputs $N_\succ$,$N_\prec$ are swapped in the connections leaving from i′;
3. $b_i = b_{i'}$ and $b_\succ = b_\prec$ hold, i.e., the biases are shared between the dual hiddens i and i′ and between the outputs $\mathbf{Y}_\succ$ and $\mathbf{Y}_\prec$.

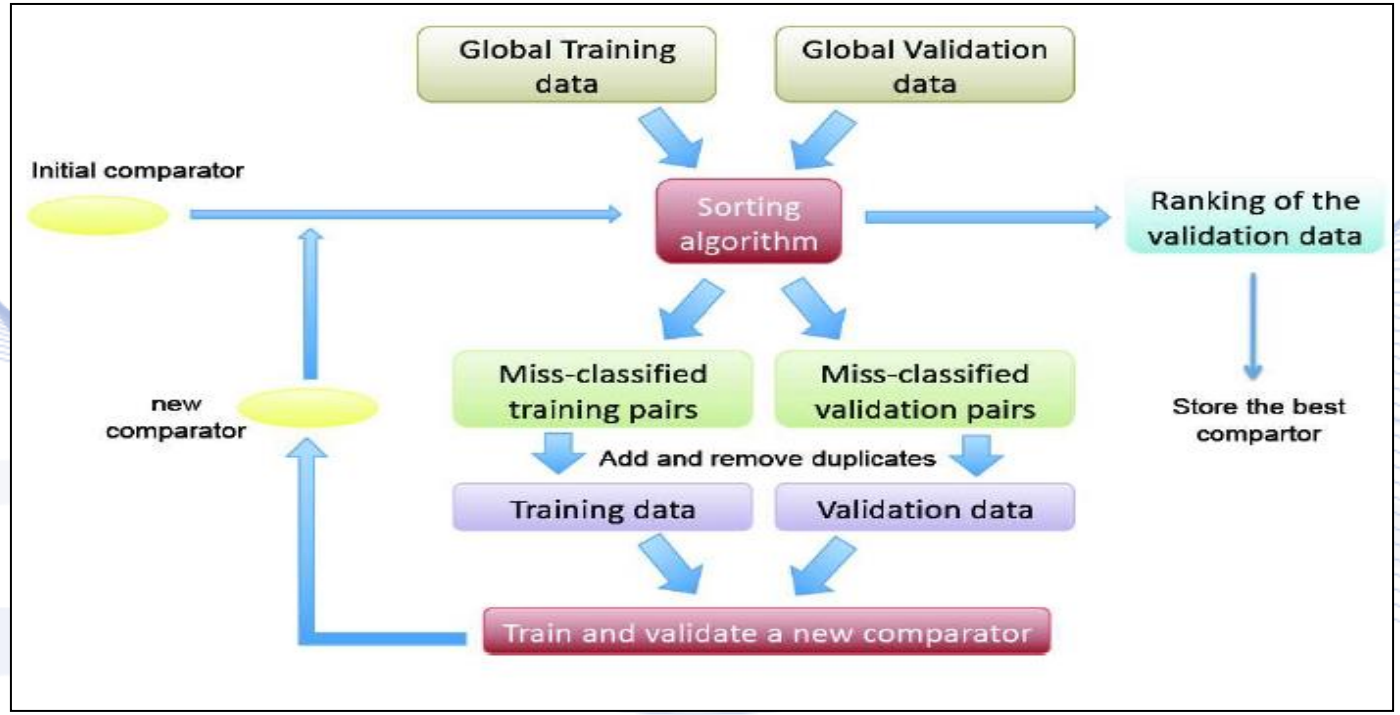
The learning algorithm

CmpNN is trained on a dataset composed of pairs of objects for which the value of the preference function is given. The comparative neural network is trained using the square error function that forces the network outputs to be close to the desired targets on each single pair of objects. When the network produces a perfect classification of any input pair, also the ranking algorithm yields a perfect sorting of the objects, but, in general, the optimization of the square error does not necessarily correspond to a good ranking.

SortNet:
Learning To Rank by a Neural Preference Function

The SortNet Algorithm (from [5])

Once the neural network is trained, SortNet can order a set of objects by O(n log n) operations, which is the lowest computational complexity reachable for a ranking algorithm.



The learning algorithm pseudocode (from [5])

$T \leftarrow$ *Set of training objects*
$V \leftarrow$ *Set of validation objects*
$P_T^0 \leftarrow$ *A small random subset of* $T \times T$;
$P_V^0 \leftarrow$ *A small random subset of* $V \times V$;
for $i = 0$ to max_iter **do**
  $C^i \leftarrow TrainAndValidate(P_T^i, P_V^i)$;
  $[E_T^i\ ,\ R_T^i] \leftarrow Sort(C^i, T)$;
  $[E_V^i\ ,\ R_V^i] \leftarrow Sort(C^i, V)$;
  $score \leftarrow RankQuality(R_V^i)$;
  if $score > best\_score$ **then**
    $best\_score \leftarrow score$;
    $C^* \leftarrow C^i$;
  end if
  $P_T^{i+1} \leftarrow P_T^i \cup E_T^i$;
  $P_V^{i+1} \leftarrow P_V^i \cup E_V^i$;
  if $P_T^{i+1} = P_T^i$ and $P_V^{i+1} = P_V^i$ **then**
    return C^*;
  end if
end for
return C^*;

Listwise

The **Listwise** method takes ranked web page lists as instances to train ranking models by minimizing a cost function defined on a predicted list and a ground truth list; the objective of learning is to provide the best ranked list. **Listwise** directly learns document lists by treating ranked lists as learning instances instead of reducing ranking to regression or classification.

ListNet

Algorithm 1 Learning Algorithm of ListNet

Input: training data $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), ..., (x^{(n)}, y^{(n)})\}$
Parameter: number of iterations T and learning rate η
Initialize parameter ω
for $t = 1$ **to** T **do**
  for $i = 1$ **to** m **do**
    Input $x^{(i)}$ of query $q^{(i)}$ to Neural Network and compute score list $z^{(i)}(f_\omega)$ with current ω
    Compute gradient $\Delta\omega$ using Eq. (3)
    Update $\omega = \omega - \eta \times \Delta\omega$
  end for
end for
Output Neural Network model ω

Algorithm of ListNet (from [6])

With Cross Entropy as metric, the loss for query $q^{(i)}$ *becomes:*

$$L(\mathbf{y}^{(i)}, \mathbf{z}^{(i)}(f_\omega)) = -\sum_{j=1}^{n^{(i)}} p_{y^{(i)}}(\mathbf{x}_j^{(i)}) \log(P_{z^{(i)}(f_\omega)}(\mathbf{x}_j^{(i)}))$$

The gradient of $L(\mathbf{y}^{(i)}, \mathbf{z}^{(i)}(f_\omega))$ *with respect to the parameter* ω*:*

$$\Delta\omega = \frac{\partial L(\mathbf{y}^{(i)}, \mathbf{z}^{(i)}(f_\omega))}{\partial\omega} = -\sum_{j=1}^{n^{(i)}} p_{\mathbf{y}^{(\omega)}}(\mathbf{x}_j^{(i)}) \frac{\partial f_\omega(\mathbf{x}_j^{(i)})}{\partial\omega}$$

$$+ \frac{1}{\sum_{j=1}^{n^{(i)}} \exp(f_\omega(\mathbf{x}_j^{(i)}))} \sum_{j=1}^{n^{(i)}} \exp(f_\omega(\mathbf{x}_j^{(i)})) \frac{\partial f_\omega(\mathbf{x}_j^{(i)})}{\partial\omega}$$

Ranking Algorithms

- Based on Frequency:
  - TF-IDF("Term frequency and inverse document frequency")

- Based on Graph-Link Analysis:
  - PageRank
  - HITS
  - SALSA(Stochastic Approach for Link-Structure Analysis)
  - UsersRank
  - SimRank

- **Neural Networks**
  - Learning to rank algorithms Categorization
    - Pointwise(Ranking with Large Margin Principles-SVM)
    - Pairwise(SortNet)
    - Listwise(ListNet)
  - Current Neural Networks for Ranking
    - GEPS
    - MatchPyramid
    - C-DSSM(and its extansion NeuBase)

Ranking Algorithms

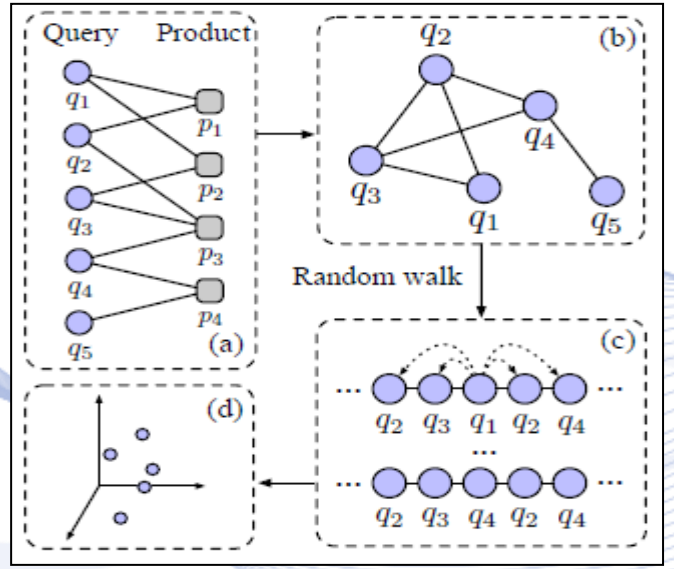
- Based on Frequency:
  - TF-IDF("Term frequency and inverse document frequency")
- Based on Graph-Link Analysis:
  - PageRank
  - HITS
  - SALSA(Stochastic Approach for Link-Structure Analysis)
  - UsersRank
  - SimRank
- Neural Networks
  - Learning to rank algorithms Categorization
    - Pointwise(Ranking with Large Margin Principles-SVM)
    - Pairwise(SortNet)
    - Listwise(ListNet)
  - **Current Neural Networks for Ranking**
    - GEPS
    - MatchPyramid
    - C-DSSM(and its extansion NeuBase)

GEPS

It is a **Graph Embedding**-based ranking model for Product Search (GEPS), which is, to the best of our knowledge, the first to integrate **click-graph** features into a unified neural ranking framework.

What is Graph Embedding?

Graph embedding is kind of like fixing vertices onto a surface and drawing edges to represent say a network. So example be like planar graph can be embedded on to a 2D surface without edge crossing.



Example of Graph Embedding (from [7])

State of the art graph embedding methods: Deepwalk, Line.

What is click-graph?

Search engines can record which documents were clicked for which and use these query-document pairs as ‘soft’ relevance judgments.

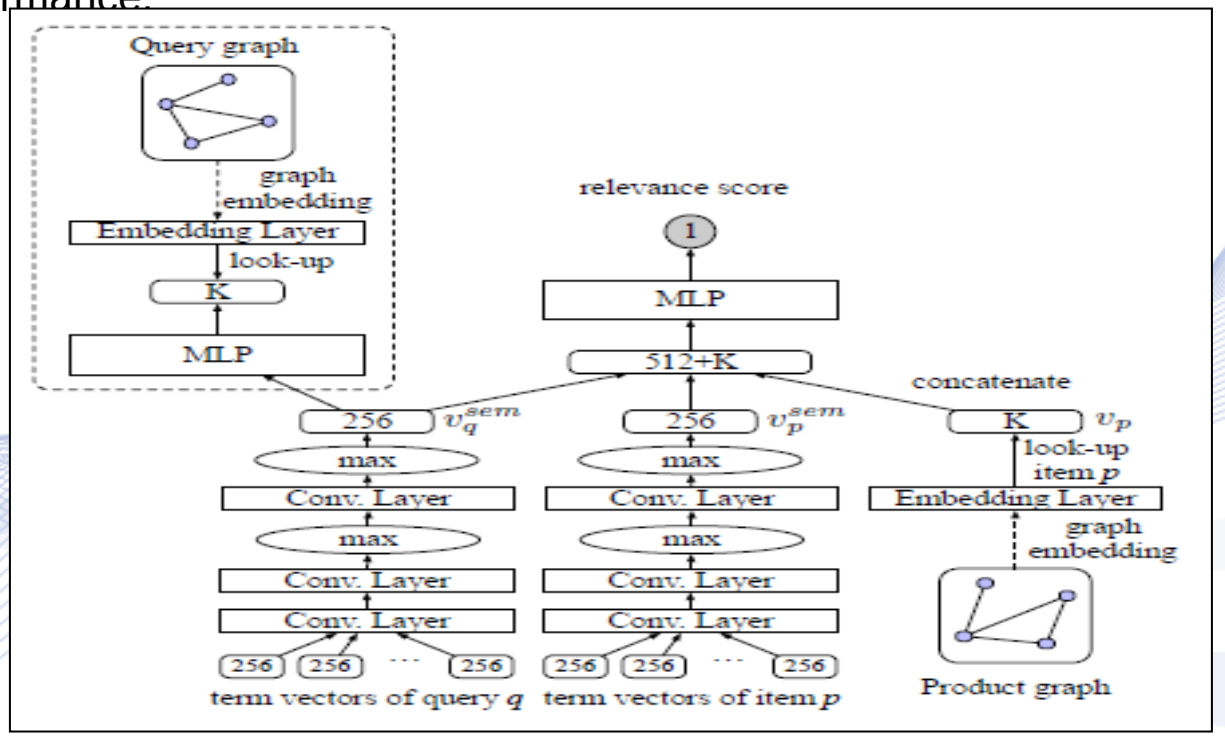
GEPS

Goal is to combine the advantages of both **graph-based** methods and **neural** approaches.

Neural network architecture for product search as the base model,

Graph embedding techniques can be plugged into such neural network models to incorporate graph structured information for better retrieval performance.

Architecture of the proposed model (from [7]):

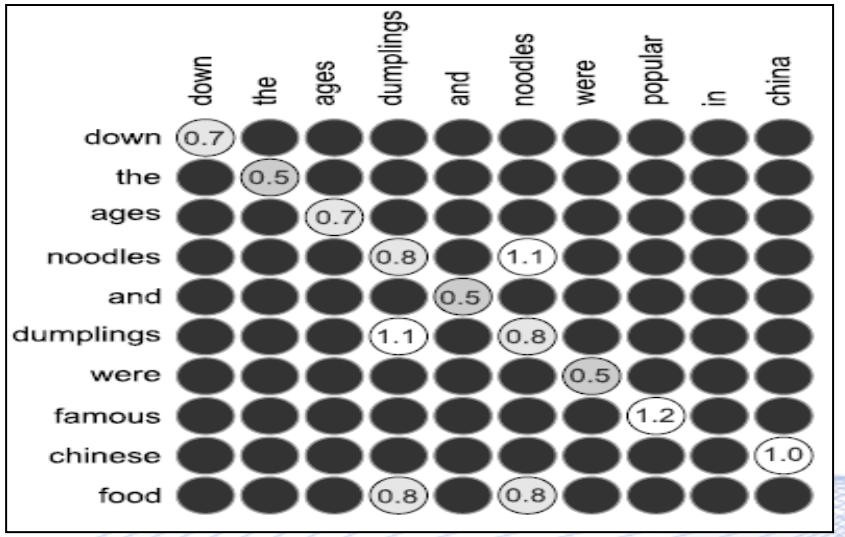


This proposed graph embedding based approach can be used as a "plug-in" to boost the performance of almost any other neural retrieval model.

MatchPyramid

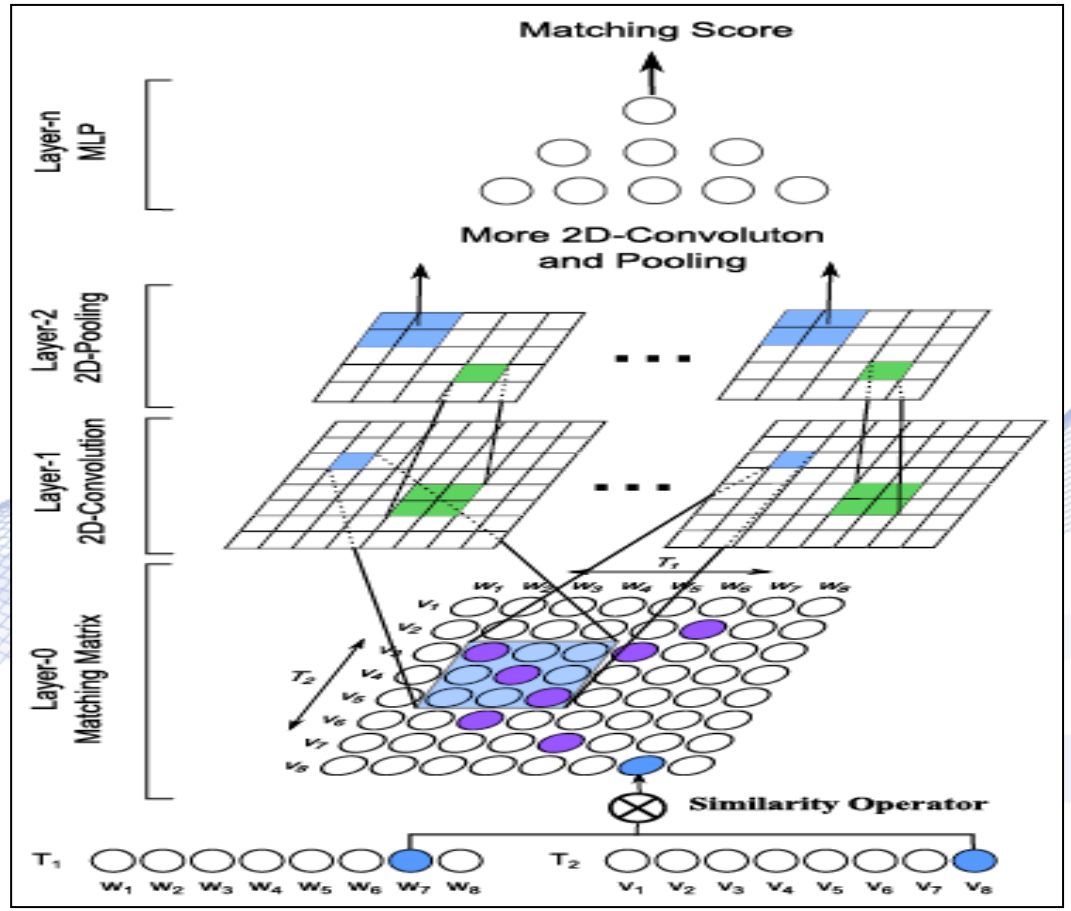
Dot Product Function further considers the norm of word vectors.

$$M_{ij} = \boldsymbol{\alpha}_i^T \boldsymbol{\beta}_j$$

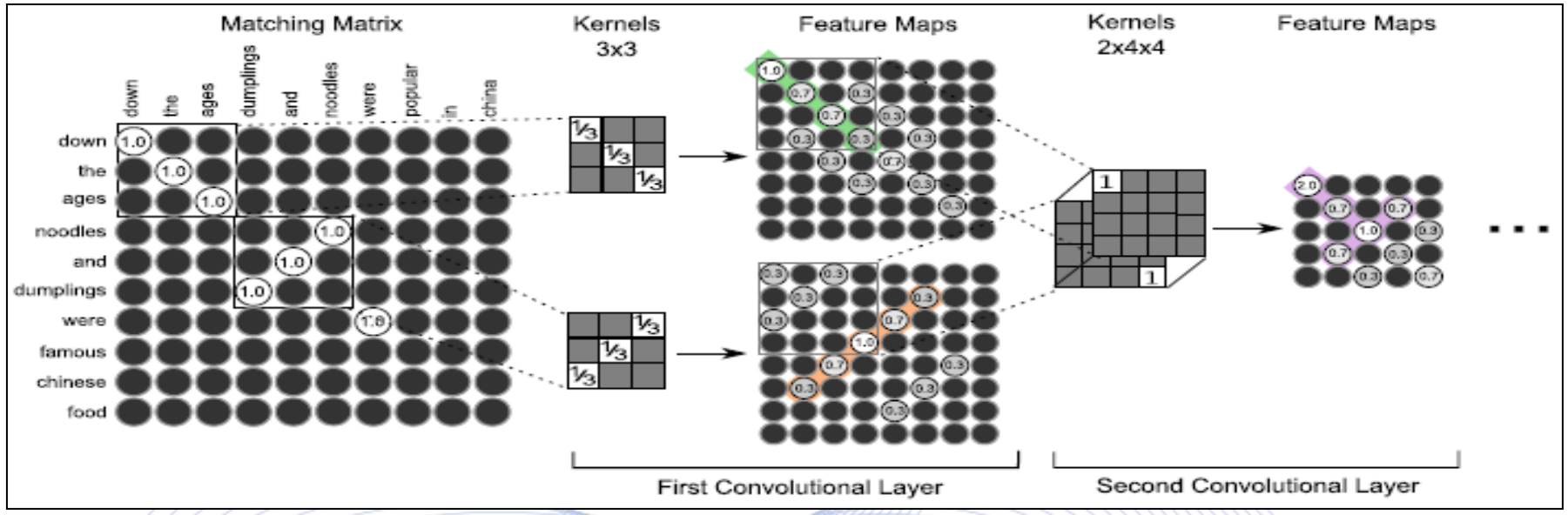


Matching Pyramid Dot Product (from [8])

An overview of MatchPyramid on Text Matching (from [8])



MatchPyramid

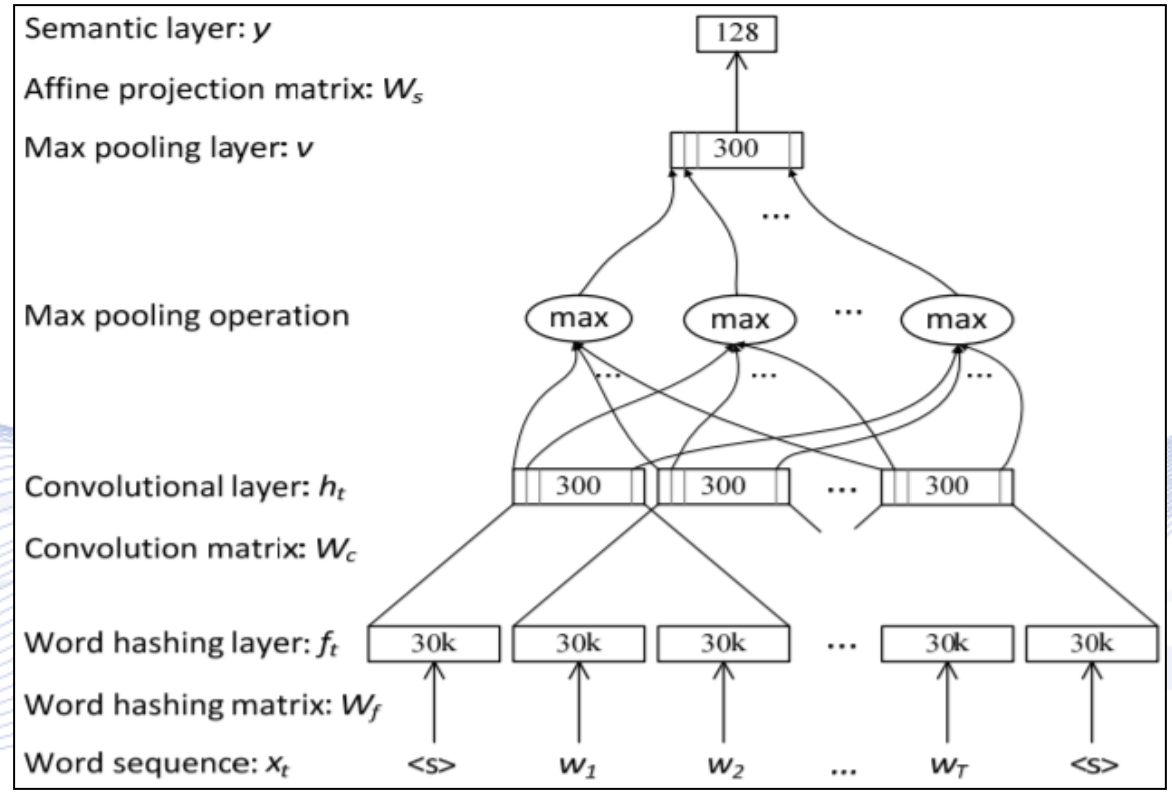


Example of Convolutional Layers(Layer1,Layer2. . . ..Layer(n-1)) using Hierarchical Convolution (from [8])

From the above analysis we can see that MatchPyramid can abstract complicated matching patterns, from phrase to sentence level, by hierarchical convolution.

Convolutional Deep Structured Semantic Models (C-DSSM)

The architecture of the **C-DSSM**, is illustrated in the picture below. The **CDSSM** contains a word hashing layer that transforms each word into a letter-tri-gram input representation, a convolutional layer to extract local contextual features, a max-pooling layer to form a global feature vector, and a final semantic layer to represent the high-level semantic feature vector of the input word sequence.



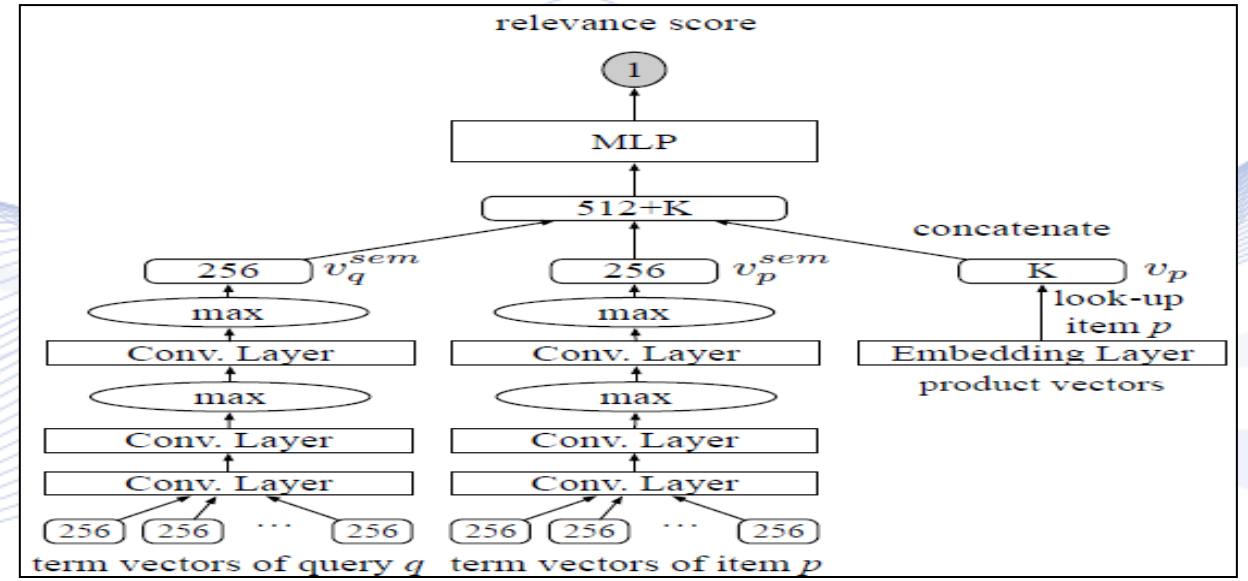
C-DSSM Architecture (from [9])

Convolutional Deep Structured Semantic Models (C-DSSM)

In this model, both the convolutional layer and the semantic layer use the **tanh function** as the non-linear activation function. We further compute the relevance score between the query and each document by measuring the cosine similarity between their semantic vectors. Formally, the semantic relevance score between a query **q** and a document **d** is measured as:

$$R(\mathbf{q},\mathbf{d}) = \cos ine(\mathbf{y}_q, \mathbf{y}_d) = \frac{\mathbf{y}_q^T \mathbf{y}_d}{\|\mathbf{y}_q\| \|\mathbf{y}_d\|}$$

where $\mathbf{y}_Q$ and $\mathbf{y}_D$ are the semantic vectors of the query and the document, respectively. **In Web search, given the query, the documents are ranked by their semantic relevance scores.** The parameters of the C-DSSM to be learned include convolution matrix W_c and semantic projection matrix W_s. Note that the word hashing matrix W_s is fixed without need of learning.



An extension of C-DSSM the NeuBase (from [7])

Introduction to: Web Search based on Ranking

- What is Web, Search engine and Ranking
- Architecture of Web Search Engine:
  Crawler, Indexer, Query Processor
- Timeline of Ranking at Indexed Pages
- Ranking Algorithms
  - Based on Frequency
    - TF-IDF
  - Based on Graph-Link Analysis
    - PageRank
    - Hits
    - Salsa
    - UsersRank
    - SimRank
  - Neural Networks
    - Learning to rank algorithms Categorization
      - Pointwise(Ranking with Large Margin Principles-SVM)
      - Pairwise(SortNet)
      - Listwise(ListNet)
    - Current Neural Networks for Ranking
      - GEPS
      - MatchPyramid
      - C-DSSM(and its extansion NeuBase)
- **Deep Learning Metrics**
  - Angular Loss
  - Nearest Neighbors Gaussian Kernels
- References

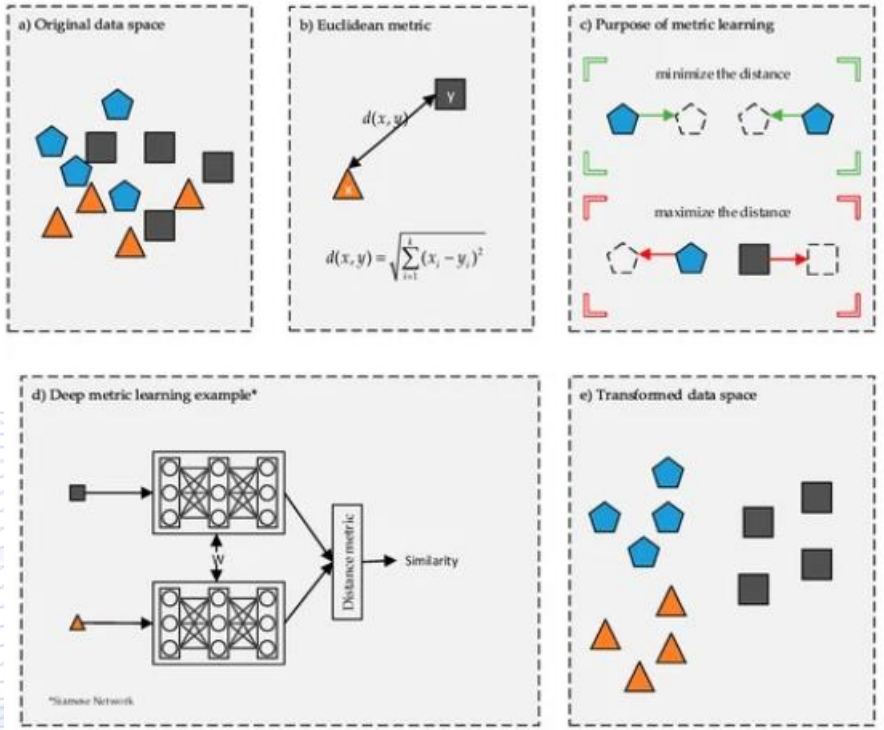
Deep Metric Learning

Metric learning is an approach based directly on a distance metric that aims to establish similarity or dissimilarity between objects. While metric learning aims to reduce the distance between similar objects, it also aims to increase the distance between dissimilar objects. For this reason, there are approaches, such as k-nearest neighbors, which calculate distance information, and approaches where the data is transformed into a new representation.

Deep learning, which provides a new representation of the data over raw data, obtains the automatic extraction of features where the goal is to achieve higher abstraction levels when transforming data. Deep learning offers a compact structure on its own and it includes the classification in the architecture. It has also a nonlinear structure that can provide us more realistic approaches to detect real-world problems with the power of activation functions.

In the last few years, deep learning and metric learning have been brought together to introduce the concept of deep metric learning. **Deep metric learning** is based on the principle of similarity between samples.

Deep Metric Learning



Example of Deep Learning Metric (from [12])

Angular Loss

This approach focus on the two main streams in **deep metric learning**, contrastive embedding and triplet embedding, and their recent variants used in computer vision.

This method pushes the negative point away from the center of positive cluster, and drags the positive points closer to each other. A novel angular loss to augment conventional distance metric learning is proposed and the main idea is to encode the third-order relation inside **triplet** in terms of the angle at the negative point.

A **triplet** is made up of three samples from two different classes, that jointly constitute a positive pair and a negative pair. The positive pair distance is encouraged to be smaller than the negative pair distance, and a soft nearest neighbor classification margin is maximized by optimizing a hinge loss.

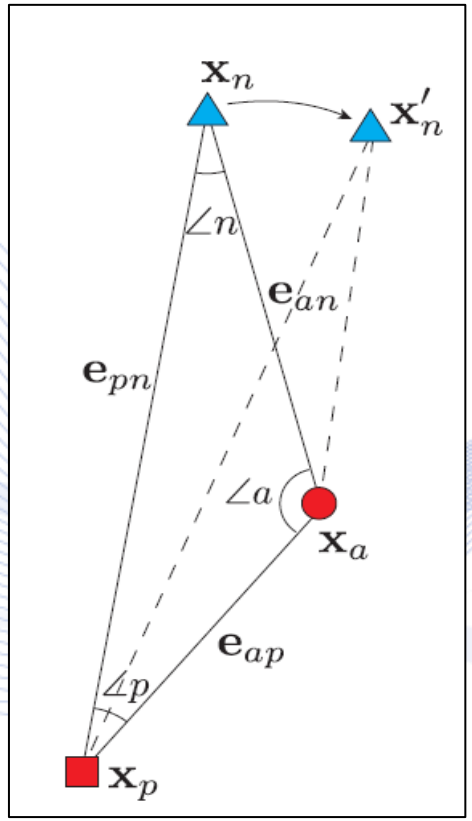
The **goal** is to push the negative point away from the center of positive cluster, and drag the positive points closer to each other.

Angular Loss

Based on these auxiliary structures, we define the new triangle Δmcn by shifting the anchor $\mathbf{x}_a$ and positive $\mathbf{x}_p$ to $\mathbf{x}_c$ and $\mathbf{x}_{am}$ respectively. Given the new triangle, we re-formulate $\angle n \leq a$ to constrain the angle $\angle n' \leq a$ closed by the edge of $\mathbf{e}_{nc}$ and $\mathbf{e}_{nm}$ to be less than a.

$$\tan \angle n' = \frac{\|\mathbf{x}_m - \mathbf{x}_c\|}{\|\mathbf{x}_n - \mathbf{x}_c\|} = \frac{\|\mathbf{x}_a - \mathbf{x}_p\|}{2\|\mathbf{x}_n - \mathbf{x}_c\|} \leq \tan a,$$

where $\|\mathbf{x}_m - \mathbf{x}_c\|$ is the radius of the circumcircle C, which equals to $\frac{\|\mathbf{x}_a - \mathbf{x}_p\|}{2}$



Example of the new Triangle (from [10])

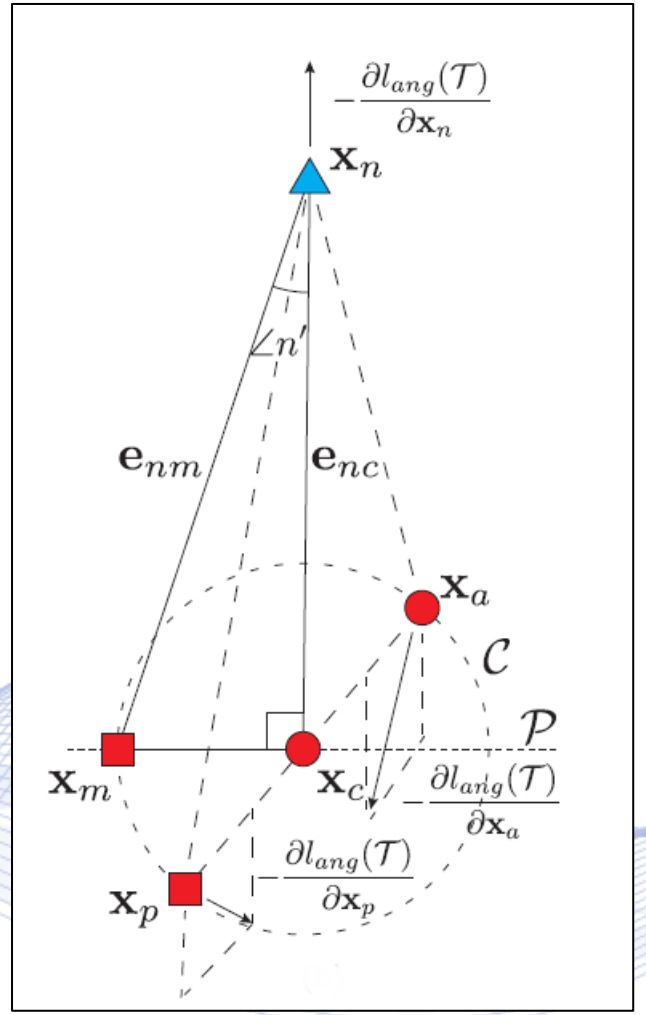
Angular Loss

The gradients:

$$\frac{\partial l_{ang}(T)}{\partial \mathbf{x}_a} = 2(\mathbf{x}_a - \mathbf{x}_p) - 2\tan^2 a(\mathbf{x}_a + \mathbf{x}_p - 2\mathbf{x}_n)$$

$$\frac{\partial l_{ang}(T)}{\partial \mathbf{x}_p} = 2(\mathbf{x}_p - \mathbf{x}_a) - 2\tan^2 a(\mathbf{x}_a + \mathbf{x}_p - 2\mathbf{x}_n)$$

$$\frac{\partial l_{ang}(T)}{\partial \mathbf{x}_n} = 4\tan^2 a(\mathbf{x}_a + \mathbf{x}_p - 2\mathbf{x}_n)$$



Example of how gradients influence the new Triangle (from [10])

Introduction to: Web Search based on Ranking

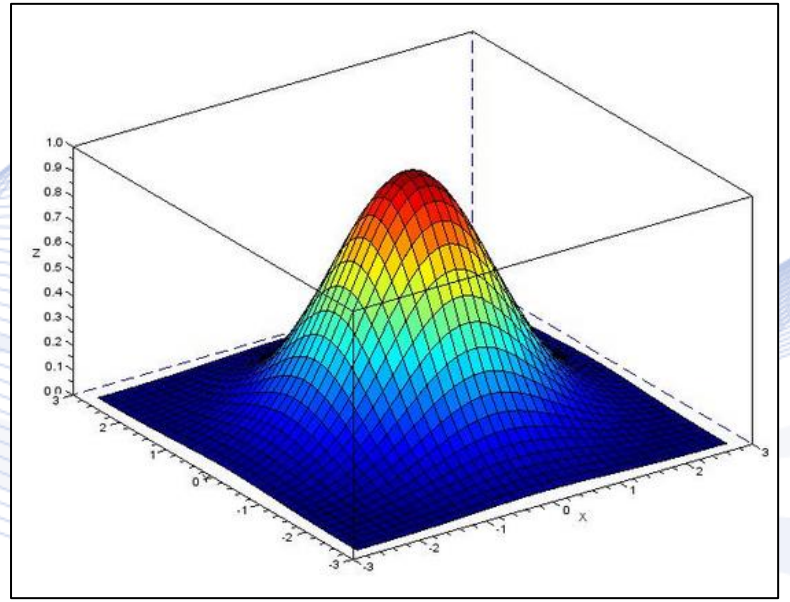
- What is Web, Search engine and Ranking
- Architecture of Web Search Engine:
  Crawler, Indexer, Query Processor
- Timeline of Ranking at Indexed Pages
- Ranking Algorithms
  - Based on Frequency
    - TF-IDF
  - Based on Graph-Link Analysis
    - PageRank
    - Hits
    - Salsa
    - UsersRank
    - SimRank
  - Neural Networks
    - Learning to rank algorithms Categorization
      - Pointwise(Ranking with Large Margin Principles-SVM)
      - Pairwise(SortNet)
      - Listwise(ListNet)
    - Current Neural Networks for Ranking
      - GEPS
      - MatchPyramid
      - C-DSSM(and its extansion NeuBase)
- Deep Learning Metrics
  - Angular Loss
  - **Nearest Neighbors Gaussian Kernels**
- References

Nearest Neighbors Gaussian Kernels

A Gaussian kernel returns a value that depends only on the distance between a point **x** and the Gaussian centre **c**. The Gaussian kernel *f* is calculated as:

$$f(\mathbf{x}, \mathbf{c}) = \exp\left(\frac{-\|\mathbf{x} - \mathbf{c}\|^2}{2\sigma^2}\right)$$

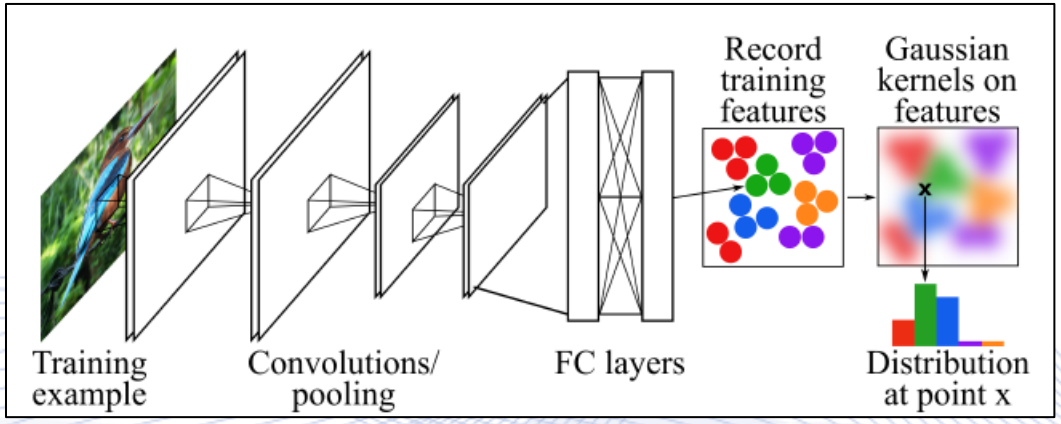
where sets the kernel width.



Gaussian Distribution (from [12])

Nearest Neighbors Gaussian Kernels

In a deep neural network the layer immediately before the loss function or classifier used as the embedding layer. The below image represents an overview of this metric approach:



Metric Approach (from [11])

A classifier is formed by the weighted sum of the kernel distance calculations between a feature embedding and the centres.

Bibliography

[1] Will Serrano: *Neural Networks in Big Data and Web Search,* (2018)

[2] Akshata D. Deore, Prof. R. L. Paikrao: *Ranking Based Web Search Algorithms,* (2012)

[3] Glen Jeh, Jennifer Widom: SimRank: *A Measure of Structural-Context Similarity,* (2011)

[4] Tie-Yan Liu: *Learning to Rank for Information Retrieval*, (2009)

[5] Tiziano Papini: SortNet: Learning To Rank By A Neural Preference Function, (2011)

[6] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, Hang Li*: Learning to Rank: From Pairwise Approach to Listwise Approach*, (2007)

[7] Yuan Zhang, Dong Wang, Yan Zhang: *Neural IR Meets Graph Embedding: A Ranking Model for Product Search*, (2019)

[8] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, Xueqi Cheng: *Text Matching as Image Recognition*, (2016)

[9] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, Gregoire Mensil: *Learning Semantic Representations Using Convolutional Neural Networks for Web Search*, (2014)

[10] Jian Wang, Feng Zhou, Shilei Wen, Xiao Liu, Yuanqing Lin: *Deep Metric Learning with Angular Loss*, (2017)

[11] Benjamin J. Meyer: *Deep Metric Learning And Image Classification With Nearest Neighbor Gaussian Kernels*, (2018)

[12] www.wikipedia.org

Bibliography

[PIT2016] I. Pitas (editor), “Graph analysis in social media”, CRC Press, 2016.
[PIT2021] I. Pitas, “Computer vision”, Createspace/Amazon, in press.
[PIT2017] I. Pitas, “Digital video processing and analysis” , China Machine Press, 2017 (in Chinese).
[PIT2013] I. Pitas, “Digital Video and Television” , Createspace/Amazon, 2013.
[NIK2000] N. Nikolaidis and I. Pitas, “3D Image Processing Algorithms”, J. Wiley, 2000.
[PIT2000] I. Pitas, “Digital Image Processing Algorithms and Applications”, J. Wiley, 2000.

Q & A

Thank you very much for your attention!

More material in
http://icarus.csd.auth.gr/cvml-web-lecture-series/

Contact: Prof. I. Pitas
pitas@csd.auth.gr