A latent Semantic Model with Convolutional-pooling structure for information retrieval

Shen, Y., He, X., Gao, J., Deng, L., & Mesnil, G.

What is Convolution?

0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

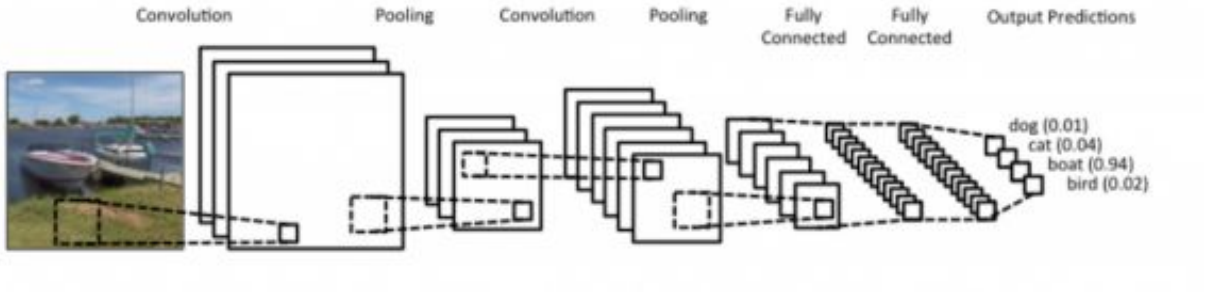
Blurring a picture

	0	1	0	
	1	-4	1	
	0	1	0	

Detecting edges

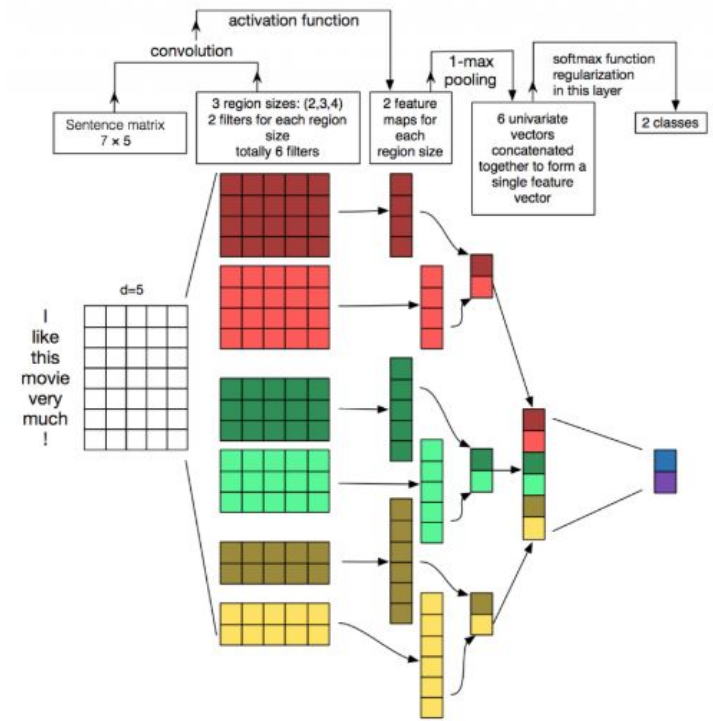
Convolutional Neural Networks

- Several layers of convolutions with ***nonlinear activation*** function like *ReLU* or *tanh* applied to the results
- Location Invariance: pooling layers (subsampling)
- Compositionality : each filter composes a local patch of lower-level features into higher-level representation.



CNN applied to NLP

- Inputs are sentences or documents represented as a matrix
- Each row corresponds to one token (word)
  - Word embeddings, like word2vec, GloVe
  - one -hot vectors
- Use filters that slide over full rows of the matrix
  - “Width” of filters the same as the width of the input matrix
  - Height, or region size vary from 2-5 words



Supervised Text Matching

Traditional IR data consists of *search queries* and *document collection*

Ground truth can be based on explicit human judgements or implicit user behaviour data (e.g. clickthrough rate)

Lexical vs. Semantic matching

Traditional IR models estimate relevance based on **lexical matches** of query terms in document

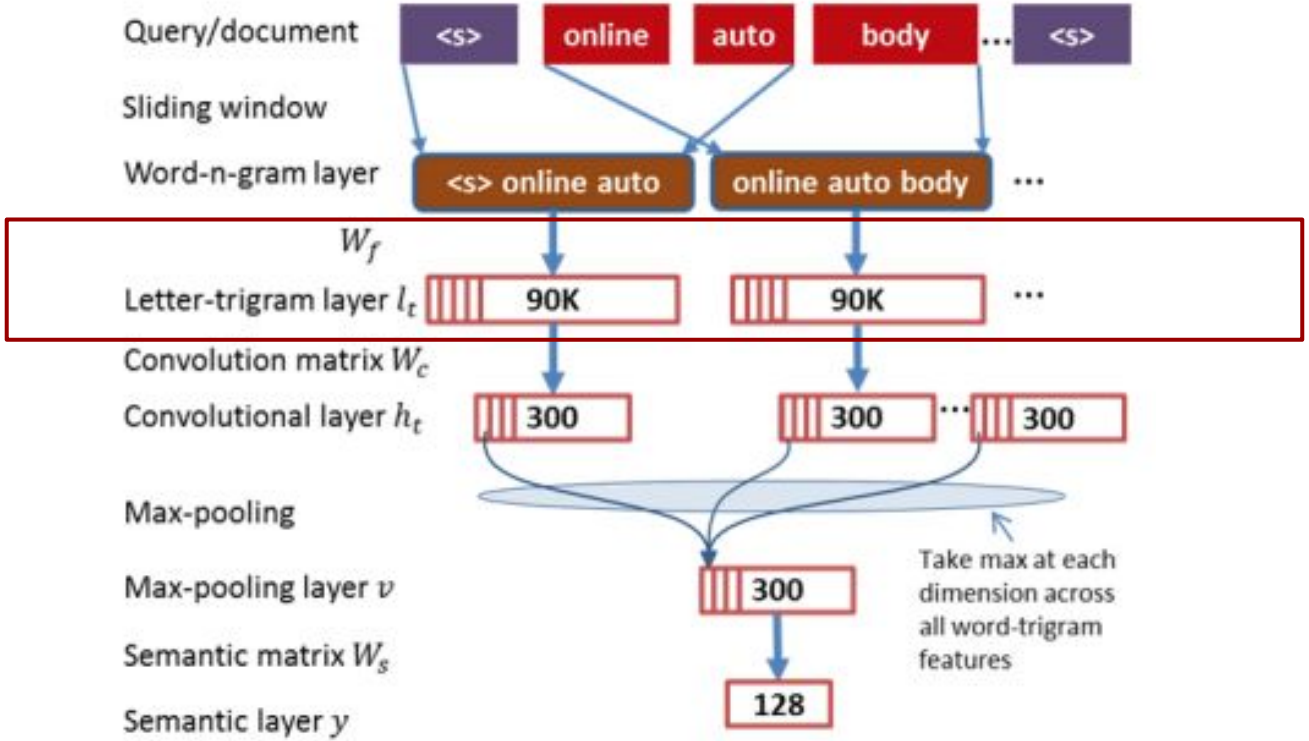
Representation learning based models garner evidence of relevance from all document terms based on **semantic matches** with query

Both **lexical** and **semantic matching** are important and can be modelled with **neural networks**

Semantic Matching

- ➔ Pros
  - Ability to match synonyms and related words
  - Robustness to spelling variations (10% of search queries contain spelling errors)
  - Helps in cases where **lexical matching** fails
- ➔ Cons
  - More computationally expensive than **lexical matching**

CLSM Architecture

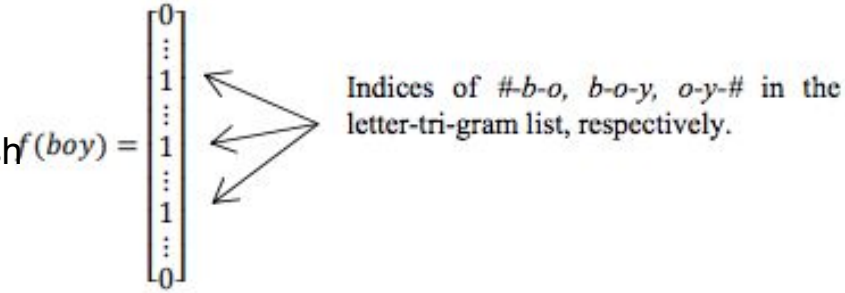


CLSM - Word Hashing

How to represent text ?

Bag of Letter Trigrams(BoLT)

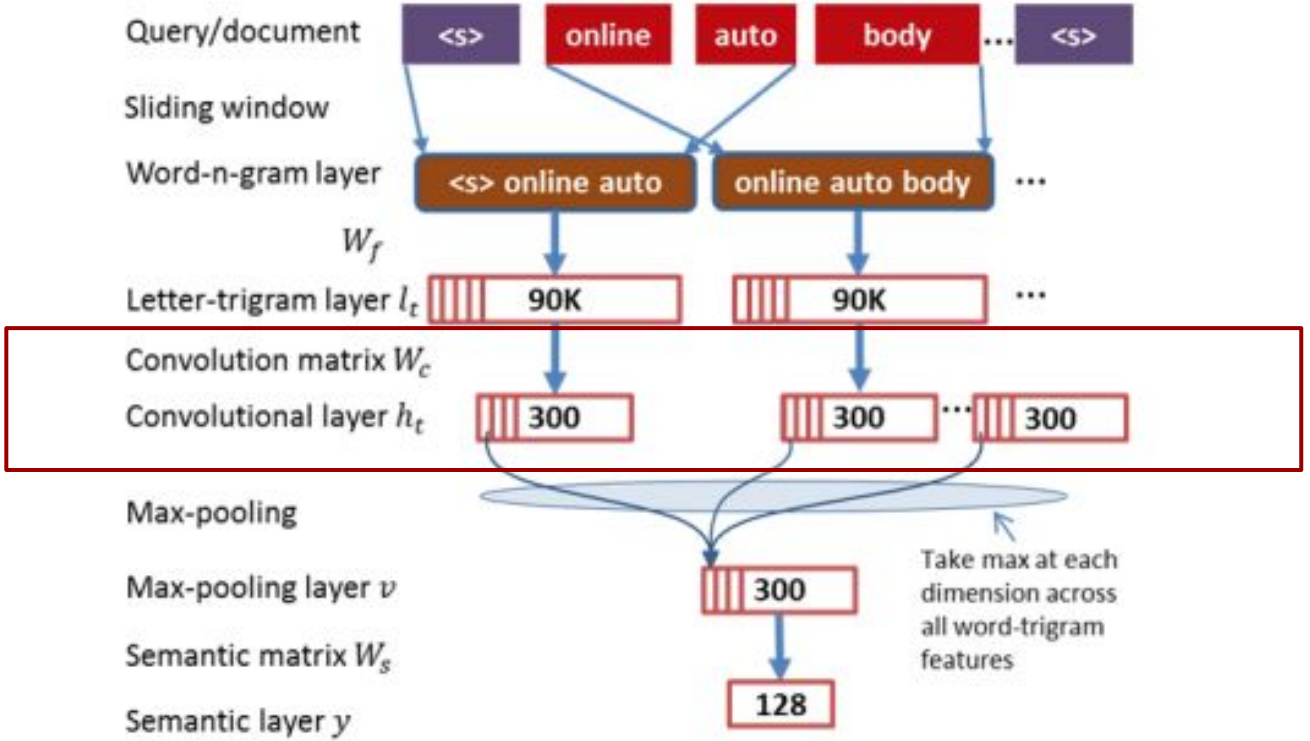
- Requires no learning
- The number of distinct letter-trigrams in English is limited
- In this experiment, about 30K unique letter-trigrams
- **Letter-trigram layer** has a dimensionality of n*30K



n =2d+1 is the size of the contextual window

$$l_t = [f_{t-d}^T, \ldots, f_t^T, \ldots, f_{t+d}^T]^T, \qquad t = 1, \ldots, T$$

CLSM Architecture



Convolutional Layer

- The convolutional operation can be viewed as **sliding window based** feature extraction
- A sequence of local contextual feature vectors, one for each word-n-gram
- Projecting **letter-trigram representation vector** l_t to a **contextual feature vector** h_t

$$h_t = tanh(W_c \cdot l_t), \qquad t = 1, \ldots, T$$

$$tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

- Output is a **variable** length sequence which is proportional to the length of the input word vector

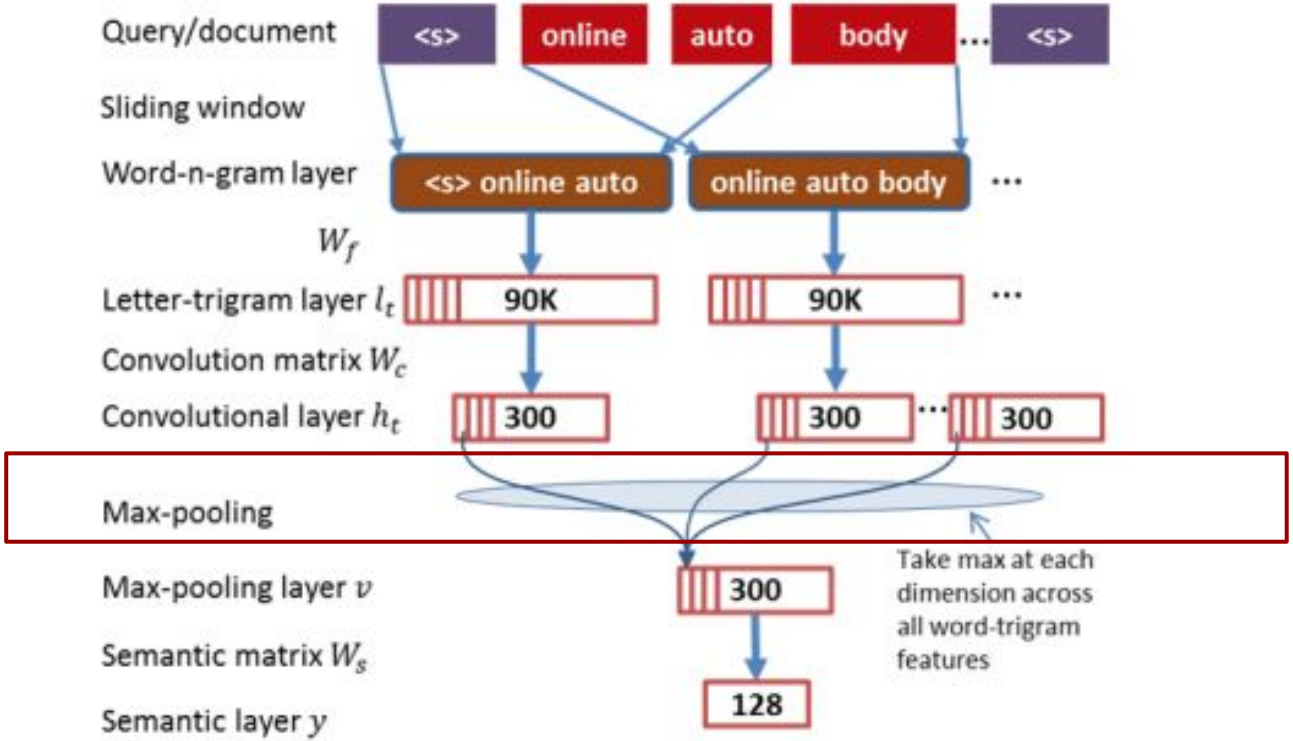
Convolutional Layer

- Words within their contexts are projected to closed vectors if they are semantically similar

microsoft *office* software		car *body* shop	
Free *office* 2000	0.550	car *body* kits	0.698
download *office* excel	0.541	auto *body* repair	0.578
word *office* online	0.502	auto *body* parts	0.555
apartment *office* hours	0.331	wave *body* language	0.301
massachusetts *office* location	0.293	calculate *body* fat	0.220
international *office* berkeley	0.274	forcefield *body* armour	0.165

Table 2: Sample word n-grams and the cosine similarities between the learned word-n-gram feature vectors of "*office*" and "*body*" in different contexts after the CLSM is trained.

CLSM Architecture



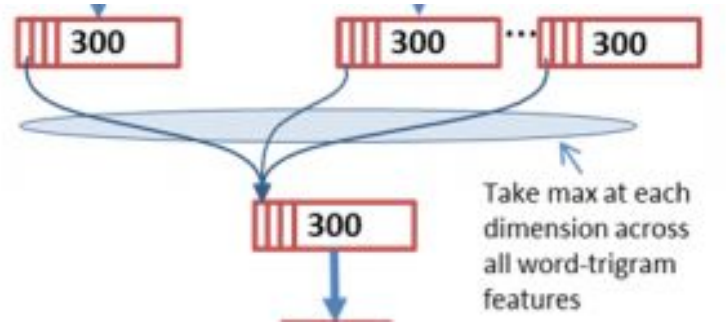
Max Pooling Layer

- **Local features** extracted by convolutional layer need to be aggregated to obtain a **sentence-level feature vector** with a ***fixed*** size
- Max pooling force the network to retain only the ***most useful*** local features
- Selecting the ***highest*** neuron activation value across all local word n-gram feature vectors ***at each dimension***

Max Pooling Layer

- v(i) is the i-th element of the max pooling layer v
- $h_t(i)$ is the i-th element of the t-th local feature vector h_t
- K is the dimensionality of the max pooling layer which is the same as the $\{h_t\}$

$$v(i) = \max_{t=1,\dots,T} \{h_t(i)\}, \qquad i = 1, \dots, K$$



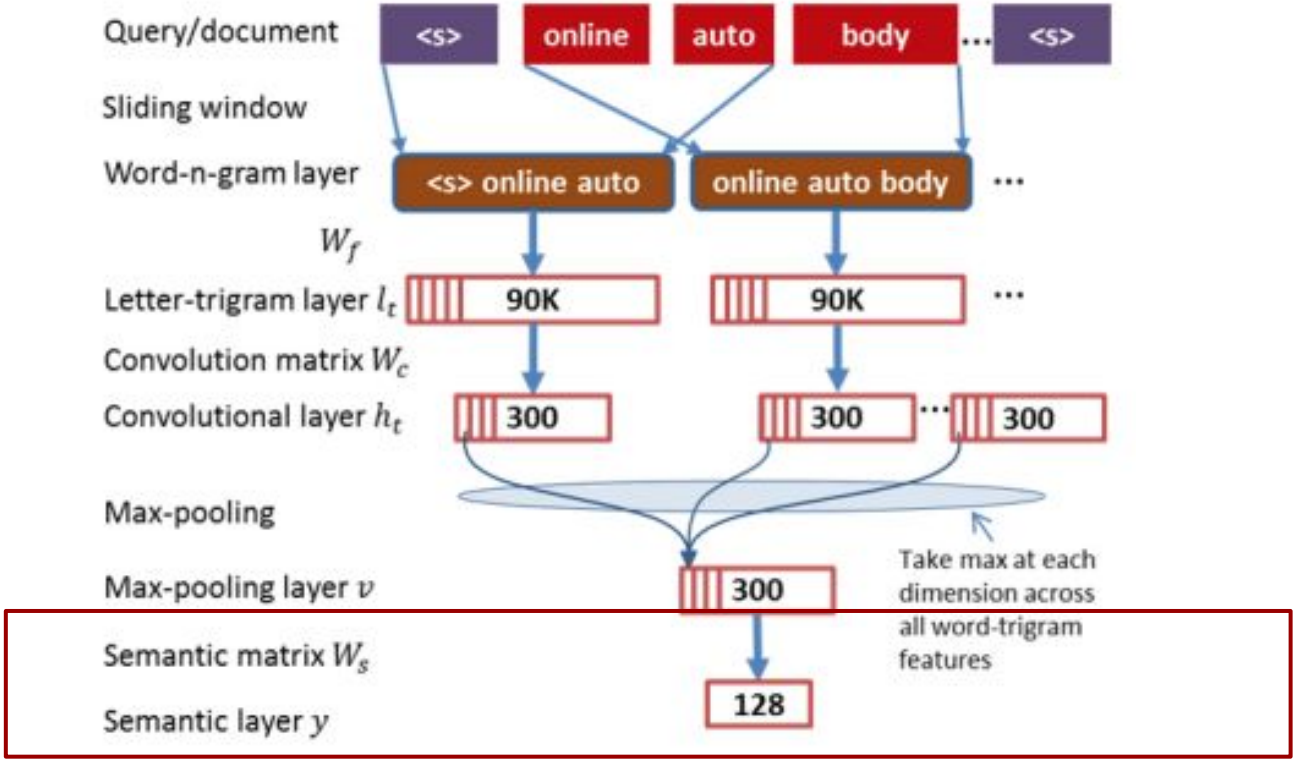
Max Pooling Layer

- Whose local features give these five highest neuron activation values
- THe important concepts make the most significant contribution to the overall semantic meaning of the sentence

microsoft **office excel** could allow remote **code execution**
welcome to the **apartment office**
online **body fat** percentage **calculator**
online **auto body** repair **estimates**
vitamin a the **health** benefits given by **carrots**
calcium supplements and **vitamin d** discussion stop **sarcoidosis**

Table 3: Sample document titles. We examine the five most active neurons at the max-pooling layer and highlight the words in **bold** who win at these five neurons in the *max* operation. Note that, the feature of a word is extracted from that word together with the context words around it, but only the center word is highlighted in bold.

CLSM Architecture



Latent semantic vector representations

- Non-linear transformation layer: to extract the high-level semantic representation $y = tanh(W_s \cdot v)$
- V is the global feature vector after max pooling
- W_s is the semantic project matrix
- Y is the vector representation of the input query in the latent semantic space, with a dimensionality of L

CLSM for IR

- Compute the semantic vector representation for the query and all the documents
- Compute the relevance score between the query and each document using cosine similarity

$$R(Q,D) = \text{cosine}(y_Q, y_D) = \frac{y_Q{}^T y_D}{\|y_Q\| \|y_D\|}$$

Where y_Q and y_D are the semantic vectors of query and the documents

- In web search, given the query, the documents are ranked by the R(Q,D)

CLSM for IR

- Training data: clickthrough data
- Train the CLSM in such a way that the ***semantic relevance scores*** between the clickthrough data given the queries are **maximized**.
- Posterior probability of a **positive** document given the query

$$P(D^+|Q) = \frac{\exp\left(\gamma R(Q, D^+)\right)}{\sum_{D' \in \boldsymbol{D}} \exp\left(\gamma R(Q, D')\right)}$$

γ is a smoothing factor in the softmax function, is set empirically. D denotes the set of candidate documents to be ranked, including the clicked document D^+, and randomly selected unclicked documents $\{D_j^-; j=1....J\}$

CLSM for IR

- Loss function: Maximizing the likelihood of the clicked documents given the queries across the training set. That is to minimize the loss function:

$$L(\Lambda) = -\log \prod_{(Q,D^+)} P(D^+|Q)$$

Λ denotes the parameter set of the CLSM

- Model trained on training data and optimized on the validation data
- Weights are randomly initialized as suggested in [1]
- Trained using mini-batch based stochastic gradient descent

[1] Montavon, G., Orr, G., Müller, K., 2012. Neural Networks: Tricks of the Trade (Second edition). Springer.

Dataset and Evaluation

- Evaluation data set henceforth:
  - 12,071 English queries labeled by human judgers
  - Each query is associated with 74 Web documents(URL)
  - 5 relevance scale: bad, fair, good, excellent, and perfect, 0-4
- Using only the title field of a Web document for ranking
  - Effective for document retrieval

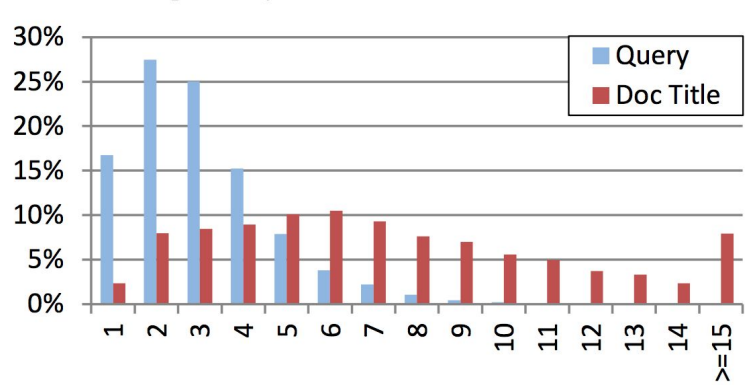


Figure 2: The distribution of query length and document title length in the evaluation data set. The evaluation set consists of 12,071 queries and 897,770 documents. Query length is 3.01 on average. Document title length is 7.78 on average.

Field	NDCG@1	NDCG@3	NDCG@10
Body	0.275	0.310	0.375
Title	0.305 $^{\alpha}$	0.328 $^{\alpha}$	0.388 $^{\alpha}$

Model Settings and Baseline Performance

#	Models	NDCG@1	NDCG@3	NDCG@10
1	BM25	0.305	0.328	0.388
2	ULM	0.304	0.327	0.385
3	PLSA (T=100)	0.305	0.335 $^{\alpha}$	0.402 $^{\alpha}$
4	PLSA (T=500)	0.308	0.337 $^{\alpha}$	0.402 $^{\alpha}$
5	LDA (T=100)	0.308	0.339 $^{\alpha}$	0.403 $^{\alpha}$
6	LDA (T=500)	0.310 $^{\alpha}$	0.339 $^{\alpha}$	0.405 $^{\alpha}$
7	BLTM	0.316 $^{\alpha}$	0.344 $^{\alpha}$	0.410 $^{\alpha}$
8	MRF	0.315 $^{\alpha}$	0.341 $^{\alpha}$	0.409 $^{\alpha}$
9	LCE	0.312 $^{\alpha}$	0.337 $^{\alpha}$	0.407 $^{\alpha}$
10	WTM	0.315 $^{\alpha}$	0.342 $^{\alpha}$	0.411 $^{\alpha}$
11	PTM (maxlen = 3)	0.319 $^{\alpha}$	0.347 $^{\alpha}$	0.413 $^{\alpha}$
12	DSSM ($J = 4$)	0.320 $^{\alpha}$	0.355 $^{\alpha\beta}$	0.431 $^{\alpha\beta}$
13	DSSM ($J = 50$)	0.327 $^{\alpha\beta}$	0.363 $^{\alpha\beta}$	0.438 $^{\alpha\beta}$
14	CLSM ($J = 4$)	0.342 $^{\alpha\beta\gamma}$	0.374 $^{\alpha\beta\gamma}$	0.447 $^{\alpha\beta\gamma}$
15	**CLSM ($J = 50$)**	**0.348** $^{\alpha\beta\gamma}$	**0.379** $^{\alpha\beta\gamma}$	**0.449** $^{\alpha\beta\gamma}$

Table 5: Comparative results with the previous state of the art approaches. BLTM, WTM, PTM, DSSM, and CLSM use the same clickthrough data described in section 5.1 for learning. Superscripts $\alpha, \beta,$ and γ indicate statistically significant improvements ($p < 0.05$) over **BM25**, **PTM**, and **DSSM ($J = 50$)**, respectively.

#	Models	NDCG @1	NDCG @3	NDCG @10
1	DSSM ($J = 50$)	0.327	0.363	0.438
2	CLSM ($J = 50$) win =1	0.340 $^{\alpha}$	0.374 $^{\alpha}$	0.443 $^{\alpha}$
3	CLSM ($J = 50$) win =3	0.348 $^{\alpha\beta}$	0.379 $^{\alpha\beta}$	0.449 $^{\alpha\beta}$
4	CLSM ($J = 50$) win =5	0.344 $^{\alpha}$	0.376 $^{\alpha}$	0.448 $^{\alpha\beta}$

Table 6: Comparative results of the CLSMs using different convolution window sizes. The setting of the DSSM is 300/300/128 and the setting of the CLSM is K=300, L=128. Superscripts α and β indicate statistically significant improvements ($p < 0.05$) over DSSM and CLSM (win=1), respectively.

Analysis

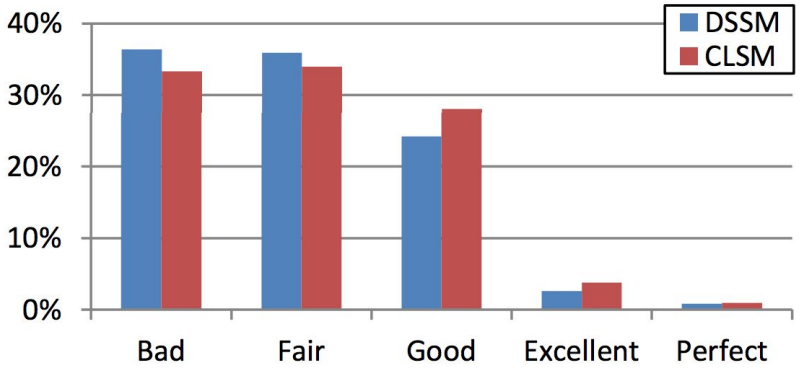


Figure 3: Percentage of top-1 ranked documents in each of the five relevance categories, retrieved by the CLSM and the DSSM, respectively.

Models		DSSM		
	Label	*bad*	*fair*	*good+*
CLSM	*bad*	3052	626	342
	fair	738	2730	631
	good+	601	981	2370

Table 7: CLSM vs DSSM on Top-1 search results. The three relevance categories, *good*, *excellent*, and *perfect*, are merged into one *good+* category.

Reference

[1]http://nn4ir.com/sigir2017/slides/NN4IR.pdf

[2]http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/