Information Retrieval and Extraction
Term Project 1

Team 2
F03944049 原博文
D05922018 程子玲
R05922038 黃郁庭

Division of work

原博文	model, report, ppt, presentation
程子玲	model, report, ppt, presentation
黃郁庭	model, report, ppt, presentation

Outline

1. Language models
    a. SDM
    b. FSDM
    c. FSDM-ELR
2. Bonus
    a. ensemble
    b. FSDM-ELR (more fields)
3. Neural network
    a. doc2vec
    b. ANN
4. Conclusion

Language models

Introduction

- Sequential Dependence Model (SDM)
  - Language Model
  - ranking function

$$P(D|Q) \overset{rank}{=} \lambda_T \sum_{q_i \in Q} f_T(q_i, D) + \lambda_O \sum_{q_i, q_{i+1} \in Q} f_O(q_i, q_{i+1}, D) + \lambda_U \sum_{q_i, q_{i+1} \in Q} f_U(q_i, q_{i+1}, D)$$

$f_T(q_i, D)$ (unigram)	$\log[\frac{tf_{q_i,D} + \mu \frac{cf_{q_i}}{\|C\|}}{\|D\| + \mu}]$
$f_O(q_i, q_{i+1}, D)$ (ordered bigram)	$\log[\frac{tf_{\#1(q_i,q_{i+1}),D} + \mu \frac{cf_{\#1(q_i,q_{i+1})}}{\|C\|}}{\|D\| + \mu}]$
$f_U(q_i, q_{i+1}, D)$ (unordered bigram)	$\log[\frac{tf_{\#uwN(q_i,q_{i+1}),D} + \mu \frac{cf_{\#uwN(q_i,q_{i+1})}}{\|C\|}}{\|D\| + \mu}]$

$\lambda_T + \lambda_O + \lambda_U = 1$

Introduction (Con't)

- Fielded Sequential Dependence Model (FSDM)
  - Mixture of Language Model (MLM)
  - ranking function

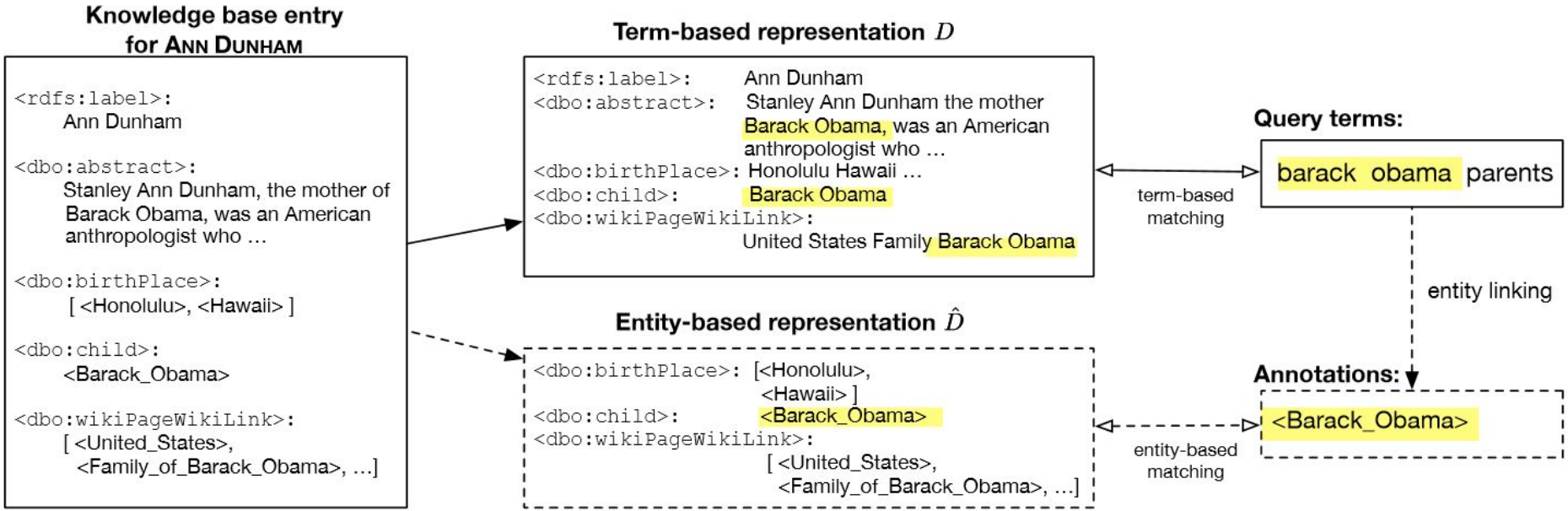
$$P(D|Q) \overset{rank}{=} \lambda_T \sum_{q_i \in Q} f_T(q_i, D) + \lambda_O \sum_{q_i, q_{i+1} \in Q} f_O(q_i, q_{i+1}, D) + \lambda_U \sum_{q_i, q_{i+1} \in Q} f_U(q_i, q_{i+1}, D)$$

$\lambda_T + \lambda_O + \lambda_U = 1$

feature function	SDM	FSDM
$f_T(q_i, D)$	$\log[\frac{tf_{q_i,D} + \mu \frac{cf_{q_i}}{\|C\|}}{\|D\| + \mu}]$	$\log \sum_f w_f^T \frac{tf_{q_i,D_f} + \mu_f \frac{cf_{q_i,f}}{\|C_f\|}}{\|D_f\| + \mu_f}$
$f_O(q_i, q_{i+1}, D)$	$\log[\frac{tf_{\#1(q_i,q_{i+1}),D} + \mu \frac{cf_{\#1(q_i,q_{i+1})}}{\|C\|}}{\|D\| + \mu}]$	$\log \sum_f w_f^O \frac{tf_{\#1(q_i,q_{i+1}),D_f} + \mu_f \frac{cf_{\#1(q_i,q_{i+1}),f}}{\|C_f\|}}{\|D_f\| + \mu_f}$
$f_U(q_i, q_{i+1}, D)$	$\log[\frac{tf_{\#uwN(q_i,q_{i+1}),D} + \mu \frac{cf_{\#uwN(q_i,q_{i+1})}}{\|C\|}}{\|D\| + \mu}]$	$\log \sum_f w_f^U \frac{tf_{\#uwN(q_i,q_{i+1}),D_f} + \mu_f \frac{cf_{\#uwN(q_i,q_{i+1}),f}}{\|C_f\|}}{\|D_f\| + \mu_f}$

Introduction (Con't)

- FSDM-Entity Linking incorporated Retrieval (ELR)
  - entity-based matching



Introduction (Con't)

- FSDM-ELR
  - query length normalization
  - ranking function

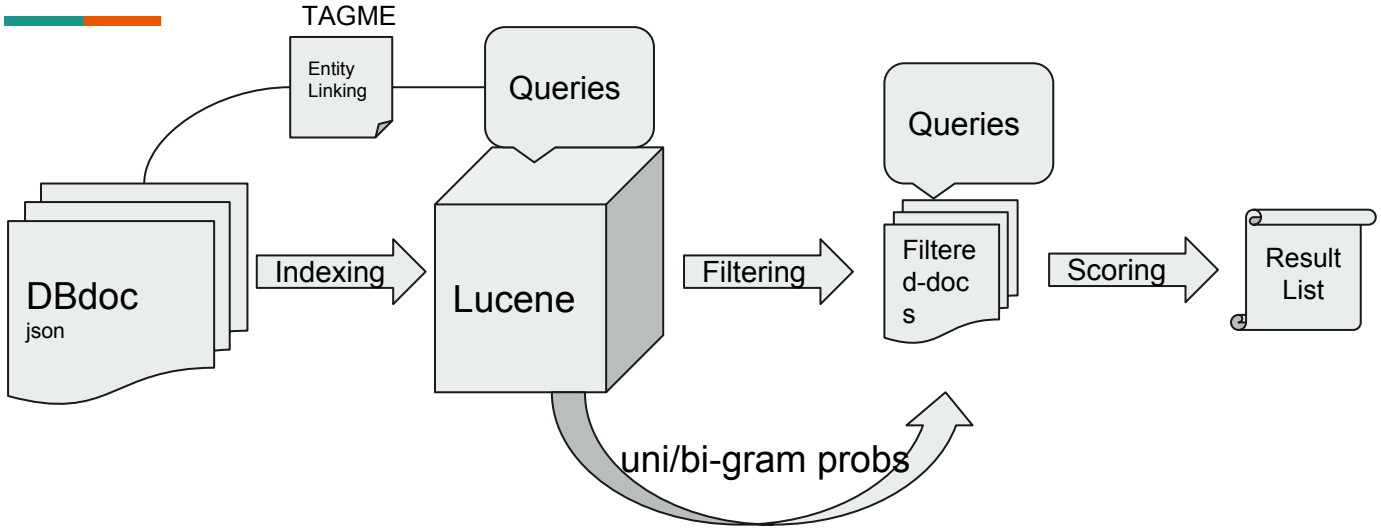
$$P(D|Q) \overset{rank}{=} \lambda_T \sum_{q_i \in Q} \frac{1}{|Q|} f_T(q_i, D) + \lambda_O \sum_{q_i, q_{i+1} \in Q} \frac{1}{|Q|-1} f_O(q_i, q_{i+1}, D) + \lambda_U \sum_{q_i, q_{i+1} \in Q} \frac{1}{|Q|-1} f_U(q_i, q_{i+1}, D) + \lambda_E \sum_{e \in E(Q)} s(e) f_E(e, D)$$

$$f_E(e, D) = \log \sum_{f \in \mathcal{F}} w_f^E \left[(1-\alpha) tf_{\{0,1\}(e, \hat{D}_f)} + \alpha \frac{df_{e,f}}{df_f} \right]$$

feature function	FSDM
$f_T(q_i, D)$	$\log \sum_f w_f^T \frac{tf_{q_i, D_f} + \mu_f \frac{cf_{q_i, f}}{\lvert C_f \rvert}}{\lvert D_f \rvert + \mu_f}$
$f_O(q_i, q_{i+1}, D)$	$\log \sum_f w_f^O \frac{tf_{\#1(q_i, q_{i+1}), D_f} + \mu_f \frac{cf_{\#1(q_i, q_{i+1}), f}}{\lvert C_f \rvert}}{\lvert D_f \rvert + \mu_f}$
$f_U(q_i, q_{i+1}, D)$	$\log \sum_f w_f^U \frac{tf_{\#uwN(q_i, q_{i+1}), D_f} + \mu_f \frac{cf_{\#uwN(q_i, q_{i+1}), f}}{\lvert C_f \rvert}}{\lvert D_f \rvert + \mu_f}$

Methodology

We follow the procedure in [Hasibi, Faegheh, 2016]



parameter	N	λ_T	λ_O	λ_U	λ_E	α
FSDM-ELR	8	0.8	0.05	0.05	0.1	0.1
FSDM	8	0.8	0.1	0.1	-	-
SDM	8	0.8	0.1	0.1	-	-

Evaluation

	MAP	nDCG@10
FSDM	0.3076	0.3773
FSDM-ELR	**0.3225**	**0.3963**
SDM	0.3102	0.3558

Discussion

- FSDM-ELR vs. FSDM
  - FSDM-ELR > FSDM
  - 在subdataset下，FSDM-ELR與FSDM的performance差異較(full dataset)大
- FSDM vs. SDM
  - 由於subdataset中fields比較少，因此FSDM與SDM的performance差異不顯著
  - MAP : SDM > FSDM
  - nDCG@10 : FSDM > SDM

	MAP	nDCG@10
FSDM	0.3076	0.3773
FSDM-ELR	**0.3225**	**0.3963**
SDM	0.3102	0.3558

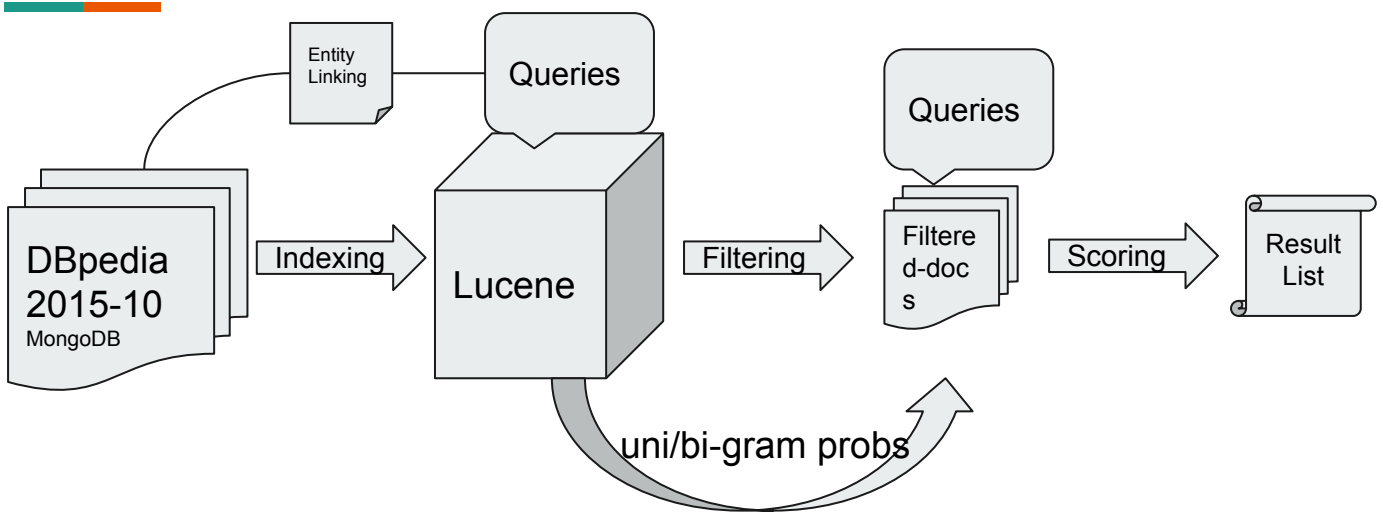
Bonus

Introduction

- dataset
  - DBPedia-2015-10
  - more fields, more entities
  - Fields: "<rdfs:label>", "<rdfs:comment>", "<foaf:name>", "!<dbo:wikiPageRedirects>", "<rdf:type>", "<dcterms:subject>", "<dbo:abstract>"
- model :
  - ensemble : 不同model在不同類別query下有不同的表現
  - 在FSDM/FSDM-ELR 中使用更多 fields

Methodology

Recall the flow



Evaluation

	SemSearch		INEX-LD		ListSearch		QALD-2		All	
	MAP	nDCG	MAP	nDCG	MAP	nDCG	MAP	nDCG	MAP	nDCG
SDM	0.4349	0.5174	0.2109	0.3390	0.2476	0.3729	0.1941	0.3039	0.2863	0.3933
FSDM	0.4680	0.5904	0.2377	0.3753	0.2664	0.3998	0.2160	0.3235	0.3125	0.4354
FSDM-ELR	0.4754	0.5759	0.2460	0.3900	0.2404	0.3443	0.2371	0.3316	0.3191	0.4276
ensemble (FSDM+FSDM-ELR)	0.4680	0.5904	0.2460	0.3900	0.2664	0.3998	0.2371	0.3316	0.3206	0.4410
FSDM-ELR (more fields)	**0.4827**	**0.5994**	**0.2614**	**0.4119**	**0.2973**	**0.4268**	**0.2371**	**0.3316**	**0.3380**	**0.4618**

Evaluation (Con't)

	MAP	nDCG@10
ensemble (FSDM+FSDM-ELR)	0.3206	0.4410
FSDM-ELR (more fields)	**0.3380**	**0.4618**
BM25F-CA (best in paper)	0.3361	0.4378

Discussion

- FSDM/FSDM-ELR vs. SDM
  - 不論是哪種 query，FSDM與FSDM-ELR均贏過SDM
- FSDM vs. FSDM-ELR
  - SemSearch、ListSearch：FSDM > FSDM-ELR
  - INEX-LD、QALD-2：FSDM-ELR > FSDM
- FSDM-ELR vs. FSDM-ELR with more fields
  - 不論是哪種 query，FSDM-ELR(more fields)均贏過FSDM-ELR
- Compared with SIGIR2017_paper
  - Our ensemble model is better than this paper in terms of nDCG@10.
  - FSDM-ELR with more fields is better than this paper in terms of MAP and nDCG@10.

Neural network

Doc2Vec Similarity

Introduction

- Doc2vec is an unsupervised algorithm to generate vectors for sentence/paragraphs/documents
- The algorithm is an adaptation of word2vec which can generate vectors for words
- Unlike sequence models like RNN, where word sequence is captured in generated sentence vectors, doc2vec sentence vectors are word order independent.
- For sentence similarity tasks, doc2vec vectors may perform reasonably well.
- We use Gensim Library to implement Doc2Vec and calculate the similarity between queries and abstract of entities.

Methodology

- We use TaggedDocument method in Gensim to prepare our vectors of training data
- We'll instantiate a Doc2Vec model with a vector size with 150 words and iterating over the training corpus 50000 times.
- We set the minimum word count to 2 in order to give higher frequency words more weighting.
- Model accuracy can be improved by increasing the number of iterations but this generally increases the training time.
- Now, we can infer a vector of queries to be compared with other vectors via cosine similarity.
- Finally, we print out the top 100 most similar entities for eachquery.

Evaluation and Discussion

	MAP	nDCG@10
Doc2Vec Similarity iteration = 55, word = 150	0.0075	0.0179
Doc2Vec Similarity iteration = 100, word = 150	0.0042	0.0116
Doc2Vec Similarity iteration = 500 , word = 150	0.0060	0.0152
Doc2Vec Similarity iteration = 100, word = 100	0.0052	0.0124
Doc2Vec Similarity iteration = 500, word = 100	0.0041	0.0120

Neural network

ANN

Introduction

- Artificial neural networks (ANN) have become a hot topic of interest and chat-bots often use them in text classification.
- The weight of synapse connecting a series of neurons and an iterative process to achieve training data.
- In each iteration there is additional fine tuning (back-propagation) to adjust to the desired pitch.
- Eventually neural network can be used for prediction, and have acceptably low error rates.
- In our experiment, we use DBdoc.json as our training data, we label each abstract as entity name.
- Afterwards, we use queries in queries-v2.txt as our testing set, and try to predict which entity should each query belong with.

Methodology

- We'll use 2 layers of neurons (1 hidden layer) and a doc2Vec approach to organizing our training data.
- We use a sigmoid function to normalize values and its derivative to measure the error rate. Iterating and adjusting until our error rate is acceptably low.
- After creating synaptic weights, we will save this as a json structure to represent our synaptic weights.
- synapse_0 += alpha * synapse_0_weight_update
- We use 20 neurons in our hidden layer, these parameters will vary depending on the dimensions and shape of training data, tune them down to ~10^-3 as a reasonable error rate.
- We can now generate the probability of a sentence belonging to one (or more) of our classes.

Evaluation and Discussion

	MAP	nDCG@10
epoch = 500	KDD3	

Conclusion

Conclusion

- 不同model在不同類別query下有不同的表現，因此使用 ensemble能提高retrieval performance。
- 考慮fields (FSDM/FSDM-ELR)能改善retrieval performance，且考慮越多fields(不超過10個)有機會提高越多performance。
- Entity Linking incorporated Retrieval (ELR) 能改善retrieval performance。

Thanks!