Information Retrieval and Extraction - Term Project 1

Team 2

D05922018 程子玲　R05922038 黃郁庭　F03944049 原博文

1.Division of work

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

2. LM Models

我們這次實做了4種models。這4種模型可以分成兩大類，其中第一種是基於語言模型：SDM、FSDM以及使用ELR方法改進的FSDM-ELR，這3種會在section.2介紹。第二類模型是基於類神經網路模型，這種方法會在section.3 介紹。

2.1.Intoiduction

a.**Sequential Dependence Model (SDM)**

SDM是一種Markov Random Field model，SDM結合了query terms彼此之間的相依關係(bigram)計算query Q與document D的joint probability P(D|Q)，依P(D|Q)大小為collection中的documents做排序。SDM的ranking function如下所示：

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

其中$\lambda_T + \lambda_O + \lambda_U = 1$，feature finction f 如下所示：

$f_T(q_i, D)$ (unigram)	$\log[\frac{tf_{q_i,D} + \mu \frac{cf_{q_i}}{\|C\|}}{\|D\| + \mu}]$	$tf_{qi,D}$：query term qi在document D中出現的次數(頻率) cf_{qi}：qi在collection中出現的次數 μ：Dirichlet prior，documents平均長度 |D|：document D長度(D中所有tokens個數) |C|：collection所有tokens個數 #1(qi,qi+1)：兩兩相鄰的term phrase #uwN(qi,qi+1)：在window size N中coocurrence 的 terms (我們在這裡設N=8)
$f_O(q_i, q_{i+1}, D)$ (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)	$\log[\frac{tf_{\#uwN(q_i,q_{i+1}),D} + \mu \frac{cf_{\#uwN(q_i,q_{i+1})}}{\|C\|}}{\|D\| + \mu}]$	

b.**Fielded Sequential Dependence Model (FSDM)**

FSDM將SDM中每個document訓練成Language Model (LM)的概念延伸，FSDM可以將structured document (這次project所使用的DBpedia-Entity v2的test collection中documents為structured)中不同的field訓練成不同的Language Model再線性結合成Mixture of Language Model (MLM)，最後再依query與documents的joint probability作排序。下表為SDM與FSDM的feature function f 的比較：

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

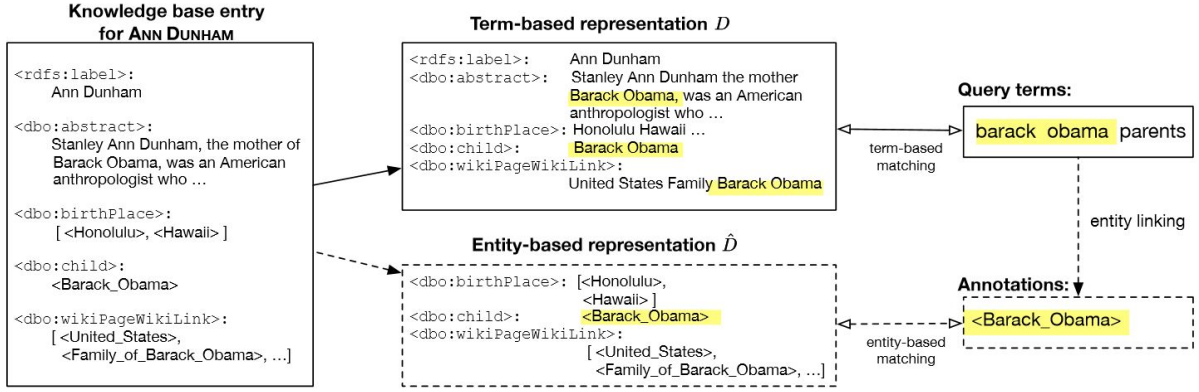
$f_U(q_i, q_{i+1}, D)$	$\log[\frac{tf_{\#uwN(q_i,q_{i+1}),D} + \mu\frac{cf_{\#uwN(q_i,q_{i+1})}}{\lvert C\rvert}}{\lvert D\rvert + \mu}]$	$\log\sum_f w_f^U \frac{tf_{\#uwN(q_i,q_{i+1}),D_f} + \mu_f\frac{cf_{\#uwN(q_i,q_{i+1}),f}}{\lvert C_f\rvert}}{\lvert D_f\rvert + \mu_f}$

其中field f ∈ F，F為universe of fields；w_f為每個field的weight，$\Sigma_f w_f = 1$；μ_f為field f 平均長度。將(1)式的SDM的三個feature function(unigram, ordered bigram, unodered bogram)分別改為上表中的FSDM的feature function便是FSDM的rank function了。

c.**FSDM based on Entity Linking incorporated Retrieval (ELR)**

(1)entity-based matching

entity-based matching的概念可以簡單用下列例子說明：



以collection中entity "Ann Dunham"為例，一般的document retrieval使用的representation單位為document (entry)中的terms，例如上圖D中的”Stanley”、"Ann”、...、"Barack"、"Obama"、...等，因此在計算unigram或bigram時都是考慮term的tf或idf；而entity linking的概念為將document的representation單位升為entity，例如上圖D̂中的"Honolulu"、"Hawaii"、"Barack_Obama"、...等，而entity對應的tf或idf的算法與term相同。當document與query的representation都可以用entity單位表示後，便可以用一般document retrieval會用到的衡量mathing方法來判斷query與document的相關度，例如在上圖中，當query Q=”barack obama parents”時，Q的entity annotation "Barack_Obama"會與D̂中的"Barack_Obama" match。

entity-based matching的feature function與SDM或FSDM的unigram類似：

$$f_E(e, D) = \log\sum_{f\in\mathcal{F}} w_f^E\left[(1-\alpha)tf_{\{0,1\}(e,\hat{D}_f)} + \alpha\frac{df_{e,f}}{df_f}\right] \quad (2)$$

其中符號意義為：

parameter	$tf_{\{0,1\}(e,\hat{D}_f)}$	$df_{e,f}$	df_f
meaning	entity e是否有出現在 $\hat{D}_f$ 中	$\lvert\{\hat{D}\lvert e \in \hat{D}_f\}\rvert$	$\lvert\{\hat{D}\lvert\hat{D}_f \neq \emptyset\}\rvert$

(2)rank function

我們將entity-based matching概念結合到上述FSDM中(直接將(2)式加到(1)式)，可以得到下列公式：

$$P(D|Q) \overset{rank}{=} \lambda_T\sum_{q_i\in Q} f_T(q_i, D) + \lambda_O\sum_{q_i,q_{i+1}\in Q} f_O(q_i, q_{i+1}, D) + \lambda_U\sum_{q_i,q_{i+1}\in Q} f_U(q_i, q_{i+1}, D) + \sum_{e\in E(Q)}\lambda_E f_E(e, D) \quad (3)$$

再經過normalize (3)式後，得到最後實作用到的rank function：

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

$$+ \lambda_U \sum_{q_i, q_{i+1} \in Q} \frac{1}{|Q|-1} f_U(q_i, q_{i+1}, D) + \lambda_E \sum_{e \in E(Q)} s(e) f_E(e, D) \quad (4)$$

其中E(Q)為query Q的linked entities的集合；|Q|為query Q的長度；s(e)為entity e的confidence score，$e \in E(Q)$，$\Sigma_{e \in E(Q)} s(e) = 1$。

2.2.Methodology

我們使用DBDoc.json 作為knowledge base， queries-v2.txt作為測試集, 以及qrels-v2.txt作為ground truth。這些資料是由助教提供的。DBDoc.json 一共有45668個entities以及對應的abstract。為了使用index.py，我們將DBDoc.json 存入MongoDB，這個code在json2mongo.py中。我們使用開源軟體Lucene作為建立index與計算term frequency的工具。Index分成URI與Term兩類，URI index是用來統計與計算entity probability， Term index用來計算abstract中的unigram和bigram的 term probablity。 同時，我們使用Lucene的內置函數合併Term index中的少大小寫與去除停用詞。以上功能實作與index.py。

queries-v2.txt一共有467個query。同上，使用Lucene去除停用詞與大小寫混淆。為了使用ELR中的Entity Linking，我們使用開源API TAGME標示出queries中的entity與對應的DBPedia的ID。我們參考API文檔，使用0.1作為門檻值來篩選entity的標示可信度。標示后的query存在tagme_annotations.json中。由於每一個query都有有45668可能的候選集，因此我們需要計算45668*467次score。即使我們使用Lucene這種可以高效率indexing的工具，依然會是一個非常大的時間開銷。因此，為了減少計算時間，我們使用Lucene提供的簡單relevance 計算，選取relevance最高的10000個entities作為篩選后的候選集。在篩選的基礎上我們使用上述models再次計算score，然後做第二次排序。

計算score的部分我們使用了https://github.com/hasibi/EntityLinkingRetrieval-ELR 中所提供的功能，這個開源軟體包含多種model的scoring，在這裡我們詳細的分析其計算SDM與FSDM與FSDM-ELR的程式部分，並依據助教所提供的資料格式做了一些更改，最後得到了實驗結果。

使用方法如同github所示，使用指令：

python -m nordlys.elr.retrieval_elr sdm

python -m nordlys.elr.retrieval_elr fsdm

python -m nordlys.elr.retrieval_elr fsdm_elr

我們使用的參數是參考SIGIR的論文，設定如下所示：

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	-	-

2.3.Evaluation

這次作業我們使用兩種指標作為實驗的評估方式，分別是MAP與NDCG@10。每一個query我們用以上指令完成二次排序後的結果選取top100作為結果用TREC格式儲存。使用trec_eval作為工具評估實驗結果：

./trec_eval -m map ../term_project/qrels-v2.txt ../term_runs/fsdm.1.treceval

./trec_eval -m ndcg_cut ../term_project/qrels-v2.txt ../term_runs/sdm10_th0.1.treceval

3種方法跑subdataset的結果如下所示：

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

2.4.Discussion

MAP與nDCG最大的差別在於：MAP只考慮document與query相關與否，而nDCG會考慮不同document與query不同的相關程度，因此在比較不同model的performance時可能會有不同的結果(不一定有一致性)。如上表所示，以MAP來看的話3種model的排序(由好到壞)為：FSDM-ELR > SDM > FSDM，而以nDCG@10來看的話其排序為：FSDM-ELR > FSDM > SDM。

不論是用MAP或nDCG@10來衡量，FSDM-ELR效果最好(幾乎不出我們預料)，因為它比SDM或FSDM多考慮了strutured documents的entities特性—利用entity-based matching更精確地判斷query與documents之間的關聯程度。然而從MAP來看SDM比FSDM好而nDCG@10卻呈現相反的結果，我們覺得很有可能是因為MAP只考慮document與query相關與否，雖然FSDM計算出來的documents的排序對關聯度比SDM敏感，但是若只考慮前幾個結果的話SDM的MAP會比FSDM好。另外其實SDM可以看成是一種single field的方法，而在subdataset中因為field不完整，所以FSDM與SDM的performance可能不會差很多。

另外在sigir2017 paper中，他們實做的FSDM-ELR的MAP與nDCG@10均比我們的版本好，應該是因為他們使用了較完整的data、具有較多的fields，所以他們能獲得更精確的fields之間的關係，而因為FSDM-ELR的performance對不同fields的線性關係很敏感，所以才會影響我們的performance。

3.Additional Model: artificial neural network (ANN)

3.1.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.

3.2.Methodology

We'll use 2 layers of neurons (1 hidden layer) and a "bag of words" approach to organizing our training data. We begin by importing our natural language toolkit. We need a way to reliably tokenize sentences into words and a way to stem words. After stemming, we organize our data structures for documents, classes and words, and let each word stemmed and lower-cased.

Next, we have our core functions for our 2-layer neural network. We use numpy because we want our matrix multiplication to be fast. 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.

Also we implement our bag-of-words function, transforming an input sentence into an array of 0's and 1's. This matches precisely with our transform for training data, always crucial to get this right.
Furthermore, we code our neural network training function to create synaptic weights.
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. The synapse.json file contains all of our synaptic weights, this is our model.
Classify() function is all that's needed for the classification once synapse weights have been calculated.
If there's a change to the training data our model will need to be re-calculated.
For a very large dataset this could take a non-insignificant amount of time.
We can now generate the probability of a sentence belonging to one (or more) of our classes. This is super fast because it's dot-product calculation in our previously defined think() function.
Experiment with other sentences and different probabilities, you can then add training data and improve/expand the model. Notice the solid predictions with scant training data.
Some sentences will produce multiple predictions (above a threshold).

3.3.Evaluation

N/A

3.4.Discussion

在ANN的model中，由於我們的training data是採用DBdoc.json中四萬多筆的資料，vocabulary set 非常大，所以在傳統使用 bag of words 建立向量時發生了out of memory的error，同時我們也考量到由於DBdoc.json每個entity的資料都只有一筆，用這樣的資料當作training data可能會效果不太好，後續我們打算使用doc2Vec取代bag of words的方法，希望能跑出ANN以及單純比較vector similarity的vector model，最後的evaluation結果會在presentation file中呈現。

4.Conclusion

從實驗可以得知，FSDM-ELR的performance比FSDM好，當documents具有structured性質時，FSDM-ELR的entity linking in queries確實能改善retrieval performance；而FSDM利用MLM的概念將不同field的model線性結合在一起，這種做法也比可以視為單一field model的SDM好。

另外一個有趣的觀察是，若我們只使用FSDM的rank funtion的第一項feature function，也就是unigram，其實這幾乎就是一般的standard language model，除了它有考慮field之間的關係。我們猜測也許因為subdataset的fields不多，這種較為簡單的方法或許反而能提高performance。

5.Bonus

5.1.intriduction

在bonus中，因為資料量變大，我們沒有更多資源與時間針對全量資料重新調整參數，因此在bonus中我們的方法和設定都沿用之前的，同時由於我們使用Lucene進行預篩選，所以並不會因為候選集的增加而明顯增加開銷，因此可以順利的跑出結果。

5.2.Methodology

我們從SIGIR 2017作者提供的url下載最新的DBPedia-2015-10 MongoDB格式。DBPedia-2015-10有4百萬個entities。與助教提供的DBdoc.json 不同， MongoDB格式的資料多出很多fields。為了更好得到結果，我們參考作者在他們paper中處理field的方式，我們在<rdfs:label> (entity) 的基礎上增加使用幾個fields，包括<rdfs:comment>，<dbo:wikiPageRedirects>，<rdf:type>，<dcterms:subject>，<foaf:name>。這些field可以提供比abstract更為完整和明顯相關信息。(FSDM-ELR*)

同時我們發現，不同的model在不同類別的query下有不同的表現，因此我們有如下疑問，是否可以有在分析每一種model在不同類別的query下的優缺點，然後得到一種ensemble的方式，使得這種ensemble后的model可以各取所長而有最好的效果。

5.3.Evaluation

為了可以更好地展現不同model在各種類別query下的表現，我們對query做了細分，做了如下表格：

	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**

我們ensemble的方法在SemSearch與ListSearch類型的query使用FSDM，在INEX-LD與QALD-2使用FSDM-ELR，這兩種模型只使用兩個fields，也就是前面subdataset中使用的entity name與abstract。另外我們還嘗試使用更多的fields在FSDM-ELR*上。

與paper的比較如下：

	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

5.4 Discussion

簡單的ESB之後，我們的NDCG@10的結果超過了Paper中最好的model，可見不同類型的query確實需要不同model來做比較好，或許我們在試過其他參數後還有機會讓performnace上升。

另外在跑subdataset時SDM雖然有時performance會比FSDM好，但是當我們使用完整dataset後，得到的結果均一致呈現FSDM比SDM好。不過ListSearch這種類型的query下FSDM-ELR卻輸給FSDM，很有可能是因為這種query retrieve回來的documents需要比較高的diversity而不是relevence，因此FSDM-ELR這種針對優化relevence的model可能比較不適合。

使用較多fields的FSDM-ELR*在MAP與nDCG@10贏過paper中最好的model，且不論是在哪種類型的query下均贏過只使用兩個fields的FSDM與FSDM-ELR，可見我們fields的種類與數量選得很洽當，這些fields能體現這些document的重要性(或代表性)，因此使用更多的fields能有更好的performance。另外因為不同query有不同的sentence性質(例如：QALD-2的query具有較完整的sentence性質，而INEX-LD的sentence性質較少)，我們猜測像INEX-LD這種較不具sentence性質的query的entity linking不一定能提供更精確的資訊，若有機會我們可以嘗試比較看看使用更多fields在FSDM上有哪幾種query有機會贏過FSDM-ELR*。