Insights into Entity Recommendation in Web Search

Nitish Aggarwal[†⋆], Peter Mika[°], Roi Blanco[°], and Paul Buitelaar[†]

[†]Insight Centre for Data Analytics
National University of Ireland
Galway, Ireland
firstname.lastname@insight-center.org
[°]Yahoo Labs
125 Shaftesbury Ave, WC2H 8HR
London, UK
pmika@yahoo-inc.com, roi@yahoo-inc.com

Abstract. User engagement is a fundamental goal for search engines. Recommendations of entities that are related to the user's original search query can increase engagement by raising interest in these entities and thereby extending the user's search session. Related entity recommendations have thus become a standard feature of the interfaces of modern search engines. These systems typically combine a large number of individual signals (features) extracted from the content and interaction logs of a variety of sources. Such studies, however, do not reveal the contribution of individual features, their importance and interaction, or the quality of the sources. In this work, we measure the performance of entity recommendation features individually and by combining them based on a novel dataset of 4.5K search queries and their related entities, which have been evaluated by human assessors.

1 Introduction

With the advent of large knowledge bases like DBpedia [5], YAGO [13] and Freebase [7], search engines have started recommending entities related to web search queries. Pound et al. [12] reported that around 50% web search queries pivot around a single entity and can be linked to an entity in the knowledge bases. Consequently, the task of entity recommendation in the context of web search can be defined as finding the entities related to the entity appearing in a web search query. It is very intuitive to get the related entities by obtaining all the explicitly linked entities to a given entity in the knowledge bases. However, most of the popular entities have more than 1,000 directly connected entities, and the knowledge bases mainly cover some specific types of relations. For instance, "Tom Cruise" and "Brad Pitt" are not directly connected in DBpedia graph with any relation, however, they can be considered related to each other.

⋆ This work was done while the author was visiting Yahoo! Research Labs, Barcelona.

Therefore, to build a system for entity recommendation, there is a need to find related entities beyond the explicit relations defined in Knowledge bases. Further, these related entities require a ranking method to select the most related ones.

Blanco et al. [6] described the Spark system for related entity recommendation and suggested that such recommendations are successful at extending users' search sessions. Microsoft also published a similar system [14] that performs personalized entity recommendation by analyzing the user click through logs. In this paper, we focus on exploring the different features in an entity recommendation system and investigate their effectiveness. Yahoo's entity recommendation system "Spark" utilizes more than 100 different features providing the evidence of the relevance of an entity. The final relevance scores are calculated by combining the different features using state-of-the-art learning-to-rank approach. Although, Blanco et al. presented some experimentation with the Spark system, in particular by reporting on the importance of the top 10 features, and the evaluation metrics on different types of entities; further experimentation is required to investigate the impact of individual features and their different combinations. The features used in Spark can be divided in five types: co-occurrence based, linear combination of co-occurrence based features, graph-based, popularity-based, and type-based features. Co-occurrence based features make use of four different data sources: query term, user specific query sessions, Flickr tags, and tweets. In this paper, we explore the impact of the features used in the Spark system by combining them based on their types and data sources. In order to investigate the quality of different data sources, we focus extensively on co-occurrence based features. All of the data sources used to calculate co-occurrence based features are not publicly accessible. For instance, only major search engines have the datasets like query terms and query sessions. Therefore, we measure the performance of a system that has only co-occurrence based features extracted from Wikipedia. The data sources like query terms, Flickr tags and tweets can only capture the presence of an entity. However, Wikipedia articles are long enough to obtain the associative weight of an entity with a Wikipedia article, which provides an opportunity to build the distributional semantic model (DSM) [1, 4, 10] over Wikipedia concepts. Therefore, in addition to co-occurrence based features that consider only the presence, we also explore the DSM based feature built over Wikipedia. We evaluate the performance by adding the Wikipedia-base features in the current Spark system, which will be referred as Spark+Wiki in rest of the paper.

2 Entity recommendation system

This section provides a detailed overview of the Spark system. Section 2.1 describes the construction of Yahoo's knowledge graph, part of which is used to obtain the potential entity candidates. Section 2.2 explains different types of features and how they are extracted from different data sources. Spark and

Spark+Wiki combines the values obtained from different features, by using a learning to rank approach, which is explained in Section 2.3.

2.1 Yahoo knowledge graph

In order to retrieve a ranked list of the entities, the system requires a list of potential entity candidates that can be considered related with the given entity. These candidates can be obtained from existing knowledge bases like DBpedia or YAGO. However, such existing knowledge bases may not cover all the relations that can be defined between the related entities. For instance, "Tom Cruise" can be considered highly related to "Brad Pitt", but they are not connected by any relation in DBpedia graph. Therefore, Spark uses an entity graph extracted from different structured and unstructured data sources including public data sources such as DBpedia and Freebase. It also uses a manually constructed ontology that defines the types of an entity extracted from different resources. In order to extend the coverage of the defined relations in entity graph, it performs information extraction over various unstructured data sources in different domains like movies, music, TV shows and sports. The subset of the entity graph used in Spark covers entity-types in media, sports and geography and consisted of over 3.5M entities and 1.4B relations at the time of our experiments (see for more detail [6]).

2.2 Feature extraction

Spark uses more than 100 different features. These features are divided into five different categories: co-occurrence based features, linear combination of co-occurrence based features, graph-based features, popularity-based features, and type-based features.

Co-occurrence features are derived from the hypothesis that the entities, which occur often in the same event or context, are more likely to be related to each other. Spark system uses 11 different types of features which are obtained by using different co-occurrence measures. Let E_1 and E_2 are two entities and S is the set of events, where $S = \{s_1, s_2, ...s_n\}$ and s_n is the n^{th} event. The event is defined as one observation under consideration for measuring the co-occurrence. For instance, every query in query logs is an event and entity occurrence is defined by $\sum_{i=0}^{N} o_i$, where $o_i = 1$ if an event s_i contains the entity E otherwise $o_i = 0$.

1. **Probability** (P_1, P_2) it is calculated by taking the ratio of the number of events that contain the given entity to the total number of events. P is the probability of an entity E.
$$P = \frac{\sum_{i=0}^{N} o_i}{N} \tag{1}$$

where N is the total number of events. The value of P of an entity is independent of the other entities, therefore it gives two values P_1 and P_2 for an entity pair consisting of E_1 and E_2.

2. **Entropy** (Ent_1, Ent_2) This is the standard entropy of an entity that is defined by
$$Ent_1 = -P_1 * log(P_1) \tag{2}$$
and P is the probability defined in feature 1. Similar to the probability feature, it gives two values Ent_1 and Ent_2 for an entity pair.
3. **KL divergence** (KL_1, KL_2) It is KL divergence of an entity E. Similar to the above features, it also gives two values $KL_1 and KL_2$ for an entity pair.
4. **Joint probability (JPSYM)** This score is obtained by taking the ratio of the number of events that contain both the given entities to total number of events.
$$JPSYM = \frac{\sum_{i=0}^{N} co_i}{N} \tag{3}$$
where $co_i = 1$ if an event s_i contains both the entities E_1 and E_2, otherwise $o_i = 0$.
5. **Joint user probability (PUSYM)** This is similar to the feature 4, however, it calculates the co-occurrence over users rather than the events.
$$PUSYM = \frac{\sum_{i=0}^{U} cou_i}{U} \tag{4}$$
where U is the total number of users and $cou_i = 1$ if a user u_i contains both the entities $E_1 and E_2$, otherwise $cou_i = 0$.
6. **PMI (SISYM)** It computes the point wise mutual information (PMI).
$$PMI(E_1, E_2) = \frac{log(P(E_1, E_2))}{P(E_1) * P(E_2))} \tag{5}$$
.
7. **Cosine similarity (CSSYM)** The cosine similarity is calculated as
$$Cosine(E_1, E_2) = \frac{P(E_1, E_2)}{P(E_1) * P(E_2))} \tag{6}$$
.
8. **Conditional probability (CPASYM)** It is calculated as the ratio of the total number of events that contain E_1 and E_2, to the total number of events that contain E_1.
$$CPASYM(E_1, E_2) = \frac{\sum_{i=0}^{N} co_i}{\sum_{i=0}^{N} oe_{1i}} \tag{7}$$
where $oe_{1i} = 1$ if an event s_i contains the entity E_1, otherwise $oe_{1i} = 0$.
9. **Conditional user probability (CUPASYM)** This is similar to the CPASYM except it computes the score over the users.
$$CUPASYM(E_1, E_2) = \frac{\sum_{i=0}^{U} cou_i}{\sum_{i=0}^{U} oue_{1i}} \tag{8}$$
where $oue_{1i} = 1$ if an user u_i contains the entity E_1, otherwise $oue_{1i} = 0$.

10. **Reverse conditional probability (RCPASYM)** It is reverse of the CPASYM.

$$RCPASYM(E_1, E_2) = \frac{\sum_{i=0}^{N} co_i}{\sum_{i=0}^{N} oe_{2i}} \tag{9}$$

where $oe_{2i} = 1$ if an event s_i contains the entity E_2, otherwise $oe_{1i} = 0$.
11. **Reverse conditional user probability (RCUPASYM)** It is reverse of the CUPASYM.

$$RCUPASYM(E_1, E_2) = \frac{\sum_{i=0}^{U} cou_i}{\sum_{i=0}^{U} oue_{1i}} \tag{10}$$

where $oue_{2i} = 1$ if an user u_i contains the entity E_2, otherwise $oue_{1i} = 0$.

Combined features are the combination of co-occurrence features. The Spark system uses 8 different types of combined features from every data source. Therefore it generates a total of 32 different features. These are the following 8 features:

1. **CF1** is the combination of conditional user probability and prior probability of target entity defined by:

$$CF1 = CUPASYM * P_2 \tag{11}$$

2. **CF2** is the combination of conditional user probability and prior probability of target entity defined by:

$$CF2 = \frac{CUPASYM}{P_2} \tag{12}$$

3. **CF3** is the combination of reverse conditional probability and prior probability of target entity defined by:

$$CF3 = RCPASYM * P_2 \tag{13}$$

4. **CF4** is the combination of reverse conditional probability and entropy of target entity defined by:

$$CF4 = RCPASYM * Ent_2 \tag{14}$$

5. **CF5** is the combination of joint user probability and prior probability of target entity defined by:

$$CF5 = JPUSYM * P_2 \tag{15}$$

6. **CF6** is the combination of joint user probability and prior probability of target entity defined by:

$$CF6 = \frac{JPUSYM}{P_2} \tag{16}$$

7. **CF7** is the combination of joint user probability and entropy of target entity defined by:
$$CF7 = JPUSYM * E_2 \tag{17}$$
8. **CF8** is the combination of joint user probability and entropy of target entity defined by:
$$CF8 = \frac{JPUSYM}{E_2} \tag{18}$$

Graph-based features use the knowledge graphs like DBpedia and Freebase. Spark computes 5 different features by using knowledge graphs.

1. **Graph similarity (GSCEG)** This feature computes the total shared connections between two given entities in Yahoo! knowledge graph.
2. **Entity popularity in movies (EPOPUMOVIE)** This feature counts the total number of directly connected nodes in *movie specific* knowledge graph, to compute the entity popularity rank.
3. **Facet popularity in movies (FPOPUMOVIE)** This is *facet* popularity rank in *movie specific* knowledge graph.
4. **Entity popularity in all (EPOPUALL)** Similar to *EPOPUMOVIE* it counts the total number of directly connected nodes in *complete* Yahoo! knowledge graph.
5. **Facet popularity in all (FPOPUALL)** This is *facet* popularity rank in the *complete* knowledge graph.

Popularity-based features

1. **Web search citation (WCTHWEB)** It counts the total hits in web search results of Yahoo!.
2. **Web deep citation (WCDHWEB)** It counts the total number of user clicks in web search results of Yahoo!.
3. **Entity Volume in query(COVQ)** It counts the total number of occurrence of given entity in query logs.
4. **Entity Volume in facet (COVF)** Facet volume in query logs.
5. **Entity view volume in query ($WPOP_1$, $WPOP_2$)** It compute the total number user clicks for given entity while the entity occur in query.

Entity type features reflect the entity types and relation types present in the knowledge bases. Spark uses two different entity type features:

1. **Entity class type (ET_1, ET_2)** This is the type of an entity defined in the knowledge base. It provides two different feature values ET_1 and ET_2 for an entity pair of the entities E_1 and E_2.
2. **Relation type (RT)** This feature defines the relation type between two given entities. For instance, “Brad Pitt” and “Angelina Jolie” are defined by relation type “Partner” in DBpedia.

Wikipedia-based features The Spark system does not use Wikipedia to extract the features. However, in addition to the features reported by Blanco et al. [6], we experiment with additional Wikipedia-based features that we refer as Spark+Wiki. Aggarwal et al. [2,3] presented an entity recommendations system "EnRG" that shows the effectiveness of using only Wikipedia-based features. Therefore, in this section we explain the additional features.

In order to obtain the Wikipedia-based features, we use Wikipedia as two types of data sources: collection of textual content and the collection of Wikipedia hyperlinks. We use 7 types of co-occurrence features from Wikipedia, where 6 out these 7 features types are already defined above: Probability (P_1, P_2), Joint Probability (JPSYM), Conditional Probability (CPASYM), Cosine Similarity (CSSYM), PMI (SISYM) and Reverse Conditional Probability (RCPASYM). The above described co-occurrence features only consider presence of an entity, as the events (search queries or tweets) used in Spark are very short in length. However, Wikipedia articles have long enough content to measure the importance of an entity to a given article (or an event in this case). Therefore, Wikipedia can provide the occurrence information of the entities with their importance weights that can be used to build the distributional vector of the entities. Spark+Wiki uses Wikipedia-based distributional semantic model (DSM) [4, 9] as an additional co-occurrence feature. DSM score is calculated by computing the cosine score between two distributional vectors. The DSM vector is defined by v, where $v = \sum_{i=0}^{N_w} a_i * c_i$ and c_i is i^{th} concept in the Wikipedia concept space, and a_i is the tf-idf weight of the entity e with the concept c_i. Here, N_w represents the total number of Wikipedia concepts. As mentioned above, we use Wikipedia as a collection of textual content and the collection of Wikipedia hyperlinks, there are 16 features that compute the values by using Wikipedia.

2.3 Ranking

In order to predict the ranking by combining all the features, Spark uses learning to rank approach [8] considering all the scores obtained from different features. As all the learning algorithm requires a training data, Blanco et al. [6] built the dataset that contains more than four thousand web search queries. Every query refers to an entity defined in knowledge graph, and contain a list of entity candidates. Finally, the dataset consists of 47,623 entity-pairs, which are tagged by professional experts. The ranking can be defined by learning a ranking function f(.) that generates a score for an input query entity q_i and an entity candidate e_j. Spark makes use of Stochastic Gradient Boosted Decision Trees (GBDT) to obtain the ranking score to decide the appropriate label for given pairs.

3 Evaluation

This section describes the evaluations of Spark and Spark+Wiki. As explained above, Spark+Wiki is actually the Spark with additional Wikipedia-based features. We evaluate the performance on a dataset that consists of 47,623 query-entity pairs. As Spark uses GBDT ranking method, we tune the GBDT parameters by splitting the dataset in 10 folds. The final parameters are obtained by performing cross validation. Due to variations in the number of retrieved related entities for a query, we use Normalized Discounted Cumulative Gain (nDCG) [11] for the performance metric. $nDCG_p$ is defined by the ratio of DCG_p to maximum or ideal DCG_p.

$$\mathrm{nDCG_p} = \frac{DCG_p}{IDCG_p}. \tag{19}$$

DCG_p is defined by:

$$\mathrm{DCG_p} = \sum_{i=1}^{p} \frac{2^{g(l_i)} - 1}{\log_2(g(l_i)) + 1} \tag{20}$$

$g(l_i)$ is the gain for the label l_i. nDCG gives different scores on different values of p, therefore, we reported the nDCG scores for 1, 5, and 10.

3.1 Datasets

Blanco et al. [6] reported the Spark performance on a dataset that consists of 4,797 search queries obtained from commercial search engines. Every query refers to an entity in DBpedia, and contains a list of entity candidates. The entity candidates are tagged by professional editors on 5 label scales: Excellent, Prefer, Good, Fair, and Bad. The dataset contains different types of entity candidates such as person, location, movie, and TV show. Table 1 provides the details about different types of instances in the dataset. It shows that most of the entities are of type "location" or "person". Section 3.3 reports the performance for these specific types in addition to the overall dataset.

Type	Total instance	Percentage
Locations	22,062	46.32
People	21,626	45.41
Movies	3,031	6.36
TV shows	280	0.58
Album	563	1.18
Total	47,623	100.00

Table 1. Dataset details

3.2 Experiment

We evaluate the performance of Spark system, and compare it with the model that was built only over Wikipedia. In order to inspect whether the additional features generated using Wikipedia can complement Spark performance, we perform the experiments with Spark+Wiki. We calculate nDCG@10, nDCG@5, and nDCG@1 as the evaluation metrics. In addition to perform experiments on the dataset with all the entity types, we also evaluated the systems for the datasets including only person type entities or location type entities. Spark combines the scores that are obtained from different types of features by using GBDT. It contains 112 features in total where 56 features are co-occurrence based, 32 features are the linear combination of co-occurrence based features, 5 features are graph-based, 6 features are popularity-based, 3 features are type-based, and the remaining 10 features are of types such as string length and Wikipedia clicks. These 56 co-occurrence based features are built over 4 different data sources: query term (QT), query session (QS), Flickr tags (FL), and tweets (TW). It means that there are 14 co-occurrence based features generated from each data source. Spark+Wiki has additional co-occurrence based features built over Wikipedia. Spark+Wiki uses the Wikipedia as two types of data sources: collection of documents with textual content and collection of documents with hyperlinks only. However, it does not generate 14 co-occurrence based features for both the data sources. Spark+Wiki uses 8 co-occurrence based features: Probability (P_1, P_2), Joint probability (JPSYM), PMI (SYSYM), Cosine similarity (CSSYM), Conditional probability (CPASYM), Reverse conditional probability (RCPASYM), and Distributional semantic model (DSM) vector. The DSM feature was not available in Spark as the data sources used in Spark have small documents (query or tweet). However, Wikipedia characteristics allow us to build the DSM vector over Wikipedia concepts [4, 9]. As a result, Spark+Wiki consists of 128 features where 16 features are additional to Spark system presented by Blanco et al. [6].

In order to investigate the importance of the features, we build the ranking model by taking the features from one category at a time. Therefore, we examine the performance of all five models: co-occurrence based, linear combination of co-occurrence based features, graph-based, popularity-based, and type-based. Further, we perform the experiments with only co-occurrence based features as they turn out to be most significant features of the system. We calculate the scores by taking co-occurrence based features and compare the importance of each data source separately.

Features	All			Person			Location		
	ndcg@10	ndcg@5	ndcg@1	ndcg@10	ndcg@5	ndcg@1	ndcg@10	ndcg@5	ndcg@1
Spark	0.9276	0.9038	0.8698	0.9479	0.9337	0.8990	0.8882	0.8507	0.8120
Wiki	0.9173	0.8878	0.8415	0.9432	0.9271	0.8857	0.8795	0.8359	0.7773
Spark+Wiki	**0.9325**	**0.9089**	**0.8747**	**0.9505**	**0.9361**	**0.9032**	**0.8987**	**0.8620**	**0.8253**

Table 2. Retrieval performance on labeled data

3.3 Result and Discussion

This section presents the results obtained from the above described experiments. Table 2 shows the retrieval performance of Spark, and compare it with

	All		Person		Location	
Rank	Feature	Importance	Feature	Importance	Feature	Importance
1	RT$	100	CUPASYMQS	100	P2WT	100
2	CSSYMFL	63.3224	DSMWL	89.6374	P2WL	57.8837
3	P2WT	55.9451	DSMWT	88.5268	DSMWL	56.6901
4	CF7FL	54.7444	RT$	87.6937	P1WL	56.3085
5	DSMWL	54.0078	CPASYMWL	83.381	P2FL	55.8144
6	CUPASYMQT	45.7274	CPASYMQT	76.5171	CSSYMFL	51.6135
7	DSMWT	42.2918	CUPASYMFL	64.0326	CPASYMFL	51.3306
8	P1WL	39.9875	CF7FL	60.437	CUPASYMFL	48.8751
9	P2FL	38.6405	CPASYMQS	53.7163	EPOPUALL	44.535
10	P2WL	36.2818	E2FL	53.6132	KL2FL	44.3557
11	CUPASYMQS	34.3559	FPOPUALL	53.4569	CF7FL	42.329
12	KL2FL	33.945	MRC2	52.9191	CF4FL	41.5688
13	CPASYMFL	33.062	JPSYMWL	52.5841	E2QT	39.6188
14	FPOPUALL	30.1997	P2WL	49.6414	FPOPUALL	38.6026
15	CF6FL	29.4447	CF8FL	48.7009	E2FL	38.0442
16	CF1FL	28.6009	EPOPUALL	48.1824	CF1QS	35.3637
17	E2FL	27.679	WPDC2	47.3189	CPASYMWT	34.6719
18	CUPASYMFL	27.2086	WPOP1	47.1626	EL2	34.3997
19	CPASYMQS	27.1851	GSCEG	46.3486	CF8FL	34.3575
20	CF8FL	26.9402	CF5QS	45.8869	CF1FL	34.0841

Table 3. Top 20 features sorted by rank according to their importance in Spark+Wiki

Spark+Wiki and the Wikipedia only model. It shows that Wikipedia-based model achieved comparable results on full dataset and person type entities. However, it could not cope well for location type entities. The possible reason behind it could be that most of the locations are too specific which do not have enough information on Wikipedia. Although, Wikipedia-based model could not outperform Spark, the combination of both i.e. Spark+Wiki achieved higher scores for all the test cases. Wikipedia-based model obtained relatively lower scores for location type entities, however, it is able to compliment the Spark performance.

In order to inspect the effectiveness of different features, we compute the feature importance in our learning algorithm. We calculate the reduction in the loss function for every split of feature variable and then compute the total reduction in loss function. It provides that how many times the given features was used in making the final decision by the learning algorithm. Table 3 shows the importance of top 20 features used in Spark+Wiki. The names of the features listed in the table correspond to their acronyms explained in section 3.2. The co-occurrence features have additional suffixes QT, QS, FL, TW, WT, and WL for query term, query sessions, Flickr tags, tweets, Wikipedia text, and Wikipedia links, respectively. For instance, the feature CSSYMFL refers to cosine similarity generated over Flickr tags. Table 3 shows that relation type (RT$) is the most important feature in Spark+Wiki which is same as reported by Blanco et al. [6]. Further, this table reports the effectiveness of the Wikipedia-based features as there are 5 Wikipedia based features in the top 10 most effective ones for the full dataset. It also shows the advantage of using additional DSM features. In particular, for person type entities, Wikipedia-based DSM feature shows a remarkable importance. Moreover, Wikipedia turned out to be a useful data source to obtain the background information about location type entities. The Wikipedia document collection created by keeping only hyperlinks, shows more effectiveness than taking all the textual content for building the DSM model. It shows the constancy of the results with the ones reported by Aggarwal and Buitelaar [4] that hyperlink-based DSM outperforms the text-based DSM model for entity relatedness and ranking. As we performed experiments by categorizing the features

Features types	All			Person			Location		
	ndcg@10	ndcg@5	ndcg@1	ndcg@10	ndcg@5	ndcg@1	ndcg@10	ndcg@5	ndcg@1
Co-occurrence	**0.9305**	**0.9054**	**0.8710**	**0.9493**	**0.9353**	**0.8983**	**0.8964**	**0.8591**	**0.81848**
Combined	0.9185	0.8908	0.8552	0.9420	0.9261	0.8844	0.8757	0.8330	0.7837
Graph	0.8953	0.8609	0.8051	0.9284	0.9088	0.8524	0.8407	0.7883	0.7157
Popularity	0.8892	0.8505	0.7886	0.9269	0.9068	0.8487	0.8355	0.7780	0.6987
Type	0.8918	0.8571	0.7965	0.9229	0.9027	0.8424	0.8318	0.7760	0.7028

Table 4. Retrieval performance per feature type

Co-occurrence Features	All			Person			Location		
	ndcg@10	ndcg@5	ndcg@1	ndcg@10	ndcg@5	ndcg@1	ndcg@10	ndcg@5	ndcg@1
QT	0.9065	0.8760	0.8312	0.9358	0.9187	0.8768	0.8592	0.8127	0.7655
QS	0.9027	0.8697	0.8215	0.9378	0.9210	0.8740	0.8421	0.7913	0.7308
FL	0.9137	0.8848	0.8477	0.9386	0.9220	0.8753	**0.8807**	**0.8424**	**0.8047**
TW	0.8910	0.8530	0.7949	0.9266	0.9070	0.8503	0.8306	0.7732	0.6992
Wiki	**0.9174**	**0.8878**	**0.8415**	**0.9439**	**0.9280**	**0.8862**	0.8795	0.8359	0.7772

Table 5. Retrieval performance per data source

based on their types, we also evaluate models which are built over the subset of the features coming from the same category. Table 4 shows the scores obtained from five different models based on the feature categories: co-occurrence features, linear combination of co-occurrence features, graph-based features, popularity-based features, and type-based features. It shows that co-occurrence based features are very effective. Although, relation-type feature turned out to be the most important feature (see table 3), the type-based features are not very effective without other features. The co-occurrence based features are built by using 5 data sources: query terms, query sessions, Flickr tags, tweets, and Wikipedia. Therefore, we reported the scores generated by co-occurrence based features over different data sources in table 5. It shows that Wikipedia is the most effective resource for all types of entities. However, for location type entities, Flickr tags perform better than Wikipedia. This shows the usefulness of the Flickr data to capture the specific and non-popular place names. Table 5 shows that Wikipedia-

	All		Person		Location	
Rank	Feature	Importance	Feature	Importance	Feature	Importance
1	P2WT	100	CPASYMWL	100	P2WT	100
2	DSMWL	87.4249	DSMWL	88.5308	P2WL	69.9444
3	DSMWT	79.5389	DSMWT	82.6514	DSMWT	63.4468
4	P2WL	74.1241	P2WL	67.7069	P1WL	53.1977
5	CPASYMWL	56.9432	P2WT	52.8873	CSSYMWT	49.4132
6	P1WL	55.0238	P1WL	52.6258	RCPASYMWT	43.3197
7	CSSYMWT	52.0919	CSSYMWT	52.2352	CPASYMWT	43.0521
8	CPASYMWT	50.898	JPSYMWL	51.948	JPSYMWT	39.9403
9	RCPASYMWL	48.3321	RCPASYMWL	49.7912	DSMWL	38.4436
10	JPSYMWT	43.949	RCPASYMWT	48.1982	P1WT	36.5881

Table 6. Top 10 features sorted by rank according to their importance in OnlyWiki

based features are the most effective ones for building the co-occurrence based model. Consequently, we further investigate the importance of Wikipedia-based features. Table 6 shows that the probability obtained from textual content is the most significant feature. However, the DSM based vectors over textual content (WT) and hyperlinks (WL) show a good relevance for the model. In all the experiments, DSM over hyperlinks shows more importance than the DSM built over textual content. The possible reason behind this could be that the DSM vector over textual content may not capture the appropriate semantics of an ambiguous entity. On the contrary, the hyperlink-based DSM vector can differentiate between ambiguous surface forms. For instance, Aggarwal and Buitelaar [4] showed that the text-based DSM vector of an entity "NeXT"[1] may not obtain the relevant dimensions while the hyperlink-based DSM vector obtained all the relevant Wikipedia articles.

4 Conclusion

In this paper, we presented an extensive evaluation of entity recommendation system called "Spark". Spark uses more than 100 features, and produces the final scores by combining these features using learning to rank algorithm. These features are built over varying data sources: query term, query session, Flickr tags, and tweets. Therefore, we investigated the performance of these features individually and by combining them based on their data source. Most of the data

[1] http://en.wikipedia.org/wiki/NeXT

sources used in Spark such as users' query logs, are not publicly available. However, Wikipedia is a continuously growing encyclopedia that is publicly available. Therefore, we showed that the model built only over Wikipedia achieved a comparable accuracy to the Spark. Moreover, Spark does not utilize the Wikipedia to build its features, thus, we also analyzed the effect of using Wikipedia as an additional resource. We showed that Wikipedia-based features complement the overall performance of Spark.

5 Acknowledgement

This work is supported by a research grant from Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289 (INSIGHT) and Yahoo! Labs.

References

1. N. Aggarwal, K. Asooja, G. Bordea, and P. Buitelaar. Non-orthogonal explicit semantic analysis. *Lexical and Computational Semantics (* SEM 2015)*, pages 92–100, 2015.
2. N. Aggarwal, K. Asooja, P. Buitelaar, and G. Vulcu. Is brad pitt related to backstreet boys? exploring related entities. In *Semantic Web Challenge ISWC (2014)*, 2014.
3. N. Aggarwal, K. Asooja, H. Ziad, and P. Buitelaar. Who are the american vegans related to brad pitt?: Exploring related entities. In *Proceedings of the 24th International Conference on World Wide Web Companion*, pages 151–154. International World Wide Web Conferences Steering Committee, 2015.
4. N. Aggarwal and P. Buitelaar. Wikipedia-based distributional semantics for entity relatedness. In *2014 AAAI Fall Symposium Series*, 2014.
5. S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.
6. R. Blanco, B. B. Cambazoglu, P. Mika, and N. Torzec. Entity recommendations in web search. In *International Semantic Web Conference (ISWC)*, 2013.
7. K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM, 2008.
8. J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.
9. E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th international joint conference on Artifical intelligence*, IJCAI'07, pages 1606–1611, 2007.
10. Z. Harris. Distributional structure. In *Word 10 (23)*, pages 146–162, 1954.
11. K. Järvelin and J. Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 41–48. ACM, 2000.
12. J. Pound, P. Mika, and H. Zaragoza. Ad-hoc object retrieval in the web of data. In *Proceedings of the 19th international conference on World wide web*, pages 771–780. ACM, 2010.

13. F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM, 2007.
14. X. Yu, H. Ma, B.-J. P. Hsu, and J. Han. On building entity recommender systems using user click log and freebase knowledge. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 263–272. ACM, 2014.