Entity oriented Task Extraction from Query Logs

Manisha Verma
University College London
m.verma@cs.ucl.ac.uk

Emine Yilmaz
University College London
emine.yilmaz@ucl.ac.uk

ABSTRACT

Identifying user tasks from query logs has garnered considerable interest from the research community lately. Several approaches have been proposed to extract tasks from search sessions. Current approaches segment a user session into disjoint tasks using features extracted from query, session or clicked document text. However, user tasks most often than not are entity centric and text based features will not exploit entities directly for task extraction. In this work, we explore entity specific task extraction from search logs. We evaluate the quality of extracted tasks with Session track data. Empirical evaluation shows that terms associated with entity oriented tasks can not only be used to predict terms in user sessions but also improve retrieval when used for query expansion.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information Search and Retrieval

Keywords

search tasks, query log analysis, task discovery

1. INTRODUCTION

Users constantly interact with search engines to accomplish some tasks such as '*buy a car*', '*plan a wedding*' etc. Such broad requirements prompts the use of multiple queries, sometimes spanning multiple sessions. Approximately 75% of user search sessions involve multi-tasking [5], which makes, task identification an important step towards understanding user goals. Recent approaches [3, 4, 5, 6] use either search query or clicked documents to identify tasks. Most of these approaches cluster queries from current or neighboring sessions into tasks based on lexical or semantic similarity.

Often, tasks are associated with some entity. Extracting and mining entity information from queries across users can provide insight into different tasks that can be accomplished with a search engine. However, current approaches do not directly leverage entities for task extraction. They use entity or entity type information as features. They semantically represent a query using features extracted, either from Wikipedia [5] or from some knowledge base [3]. These representations do not concretely capture entity specific intents in user queries. For instance, two queries that share same or similar type of entities, may still have diverse concept representation. Such queries will not be classified as part of the same task. For example, above approaches may not consider '*buy wedding flowers*' and '*book wedding destinations*' to be part of the same task due to the topic drift that '*destination*' and '*flower*' induce in concept vectors, even though both queries represent the same task - '*wedding planning*'. Whereas the entities in these two queries - '*flower*', '*destination*' and '*wedding*' shall have high co-occurrence in search logs. Queries, such as these, can be easily mapped to the same task by leveraging their entities. By considering the entities and their associations one can extract better semantics from user queries. Existing work extracts tasks from independent sessions, thus providing information only about a single user. Such tasks have limited applications as they do not give a complete picture about tasks that exist globally. However, entity oriented tasks can be extracted from search sessions accross several users. Such a global set of tasks can benefit related search applications too. For instance, it can be used to find similar users by mining their task histories or for query suggestions.

With this motivation, in this work we explore entity based task extraction from search logs. Our system finds entity oriented tasks for each category by populating words that co-occur with entities from that category. We evaluate the quality of extracted tasks in two ways: 1) Query term prediction and 2) Query expansion. Given a session, we use task terms related to entities in the first query to predict terms of subsequent queries in the session. We further show that the proposed method can improve retrieval performance when used for query expansion. Experiments on Session track[1] data indicate that terms associated with entity oriented tasks can not only predict query terms in a session but can also improve retrieval when used for query expansion.

2. RELATED WORK

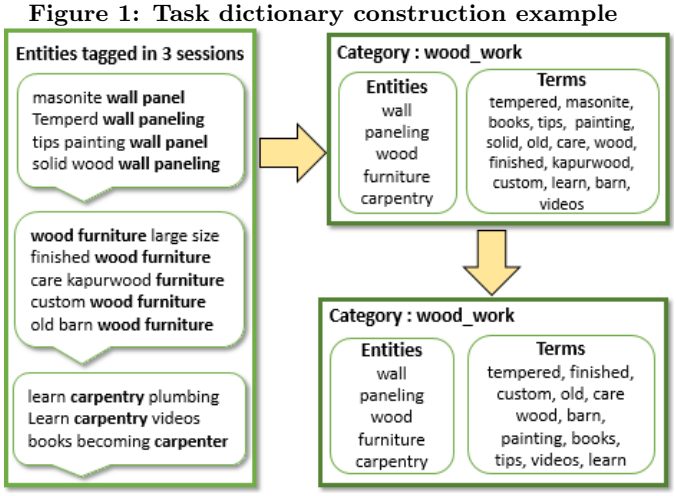
Recent work mainly explores task extraction from search sessions. The objective in [3, 4, 5] is to segment a search session into disjoint sets of queries where each set represents

[1] http://ir.cis.udel.edu/sessions/

CIKM'14, November 3–7, 2014, Shanghai, China.

http://dx.doi.org/10.1145/2661829.2662076.

Figure 1: Task dictionary construction example



a different task. These approaches do not aggregate tasks across users, thus cannot combine or differentiate between tasks extracted from different sessions. Although, Lucchese *et al.* [6] attempt to cluster tasks across users to create a global representation, their approach uses only category information in Wikipedia to calculate semantic features. However, their approach cannot distinguish between tasks associated with different types of entities. Entity recognition with a knowledge base is used in [3], again, only to enrich concept vectors. Ji *et al.* [4] find global task representations using a similar approach, but they manually tag phrases with tasks. Our work differs from previous work as we identify tasks not by calculating semantic similarity, but by aggregating queries with its entities and their category information.

To find entities in text one can use entity linking techniques such as [2, 7]. These systems can link short texts such as queries to a knowledge base such as DBPedia[2]. Another work [8], closely related to ours, finds query reformulations using entity linking and Wikipedia. They extract expansion terms from Wikipedia and score them using anchor text information. However, we extract expansion terms from search logs and use only category information of an entity to score terms.

3. ENTITY BASED TASK EXTRACTION

Our goal is to use search queries to create a comprehensive list of tasks associated with an entity. Several factors have to be considered before building such a list. First consideration is the representation of task itself. What shall be used to represent a task? In earlier works, it was a sequence of queries. For simplicity, we use a dictionary of terms to represent a list of tasks. Alternatively, one can use a list of phrases, queries or more complicated representations. We chose terms as they can succinctly capture the task (e.g. buy, sell, design) associated with an entity (e.g. ticket) while providing the flexibility to build more complex representations (vectors, networks etc.) of tasks.

The second factor is the granularity of tasks associated with an entity. How specific or general will this list be? It depends on the source of entities. For instance, Freebase has more entities and entity types than DBPedia. Thus, tasks for more categories will be extracted from Freebase than DBPedia. However, since DBPedia is extracted from Wikipedia it is less prone to noise. In this work, we use DBPedia entities and categories to extract tasks.

Final consideration is the method of creating this task list. One can use several approaches like K-Means clustering or algorithms like Random Walk to build task list for an entity. We choose to group terms based on the entities and queries they appear with to build a list of task terms.

[2]http://www.dbpedia.org

Table 2: Summary of Category Task Dictionaries

No of Categories	15510
Avg #entities per cat	20
Max #entities per cat	9356 (living people)
Avg #terms per cat	118
Max #terms per cat	11538 (states of US)

3.1 Entity Linking

Naturally, a system relying on entities needs a method to link query text with entities. We use Dexter [1] to link queries with entities. Dexter, in turn relies on DBPedia for entities and their type information. An entity may belong to different categories. For each query, Dexter returns the phrases that map to an entity (entity mentions), the entity and its categories.

3.2 Task Dictionary Construction

We represent tasks as a collection of diverse but conceptually related terms. We refer to these lists as task dictionaries. Constructing a dictionary for every entity will yield too many entities with only handful of words. The Zipfian distribution of queries will yield a skewed list of terms, since popular entities will contain more words than rare entities. However, aggregating these terms under entity category will yield a comprehensive list as terms from similar entities (popular or rare) will get grouped together. The process of creating these task dictionaries is as follows.

- We begin by tagging entities in a query. As mentioned before, we shall aggregate query terms on category level. For each entity in the query, we associate non entity terms with its categories. Non-entity terms are query terms other than the entity mentions. For example, in query '*flights between paris and London*', *'London'* is an entity, *'City'* its category and *'flights'* and *'paris'* are non-entity terms.

- The previous step results in each category (or entity type) containing several terms. This list, even though exhaustive, will contain some noise. Filtering and scoring this list is important since we want a clean (even if small) task terms list. We rank these terms on the basis of category tf-Idf given by

$$tf\text{-}Idf(t_i, c_j) = tf(t_i, c_j) \times log(N_c/N_{t_i}) \qquad (1)$$

  where $tf(t_i, c_j)$ is the frequency of term t_i in category c_j, N_c is the total number of categories and N_{t_i} is the number of categories that contain the term t_i. We retain those words in the dictionary whose tf-Idf exceeds a certain threshold.

An example of aggregating and filtering query terms for the task dictionary of category '*wood_work*' is depicted in

Table 1: Comparison of Term Prediction Results [3]

	2011 Session Track					2012 Session Track				
		50 Tasks		100 Tasks			50 Tasks		100 Tasks	
Prec	**Ent**	**HTC**	**QCC**	**HTC**	**QCC**	**Ent**	**HTC**	**QCC**	**HTC**	**QCC**
1	**0.014**	0.012	0.004	0.005	0	0.001	0.012	0.012	0.004	**0.019**
5	0.04	0.021	0.019	0.02	**0.046**	0.022	0.019	0.031	**0.034**	0.026
10	0.046	0.035	0.037	0.038	**0.050**	0.041	0.045	0.049	**0.051**	0.049
15	0.051	0.052	0.059	0.061*	**0.064***	0.034	0.045	0.055	0.059	**0.062**
20	**0.072**	0.057	0.062	0.064	0.067	**0.073**	0.049	0.055	0.066	0.072
40	0.086*	0.062*	0.083*	0.084*	**0.089***	**0.096***	0.062	0.069	0.090*	0.092*
50	0.093*	0.066*	0.096*	0.089*	**0.106***	**0.113***	0.070	0.075	0.102*	0.092*

Figure 1. During Step 1, the system aggregates terms on a category node and in Step 2 it cleans this term set.

4. TASK DICTIONARY EVALUATION

Manual evaluation of dictionaries constructed above is infeasible. Since there is no labeled dataset for tasks evaluation, in this work, we evaluate them indirectly with query term prediction and query expansion. Query reformulations in a session are users' indication of possible terms that can be added or removed from the query to accomplish a certain task. The tasks associated with entities in current query can be used to predict terms of future queries. Similarly, the terms from these tasks can be used to enrich this query to improve retrieval. This is the underlying intuition of using both query term prediction and query expansion to evaluate the quality of generated task dictionaries.

Since a limited number of quality terms are required for evaluation, we need to score task dictionary terms with respect to the query. For instance, the query '*lake murray resort*' has one entity '*lake murray*' whose category is '*Reservoirs in South Carolina*'. Task dictionary of this category contains over 50 terms, and since all its terms may not be equally relevant (either for query term prediction or query expansion), its necessary to rank them with respect to the query. Thus, we propose the following scoring mechanism to find most suitable terms for a query given its entities and their types respectively.

- We begin by finding entities in the query. An entity may belong to several categories, for instance, entity '*apple*' may belong to two categories- '*company*' and '*fruit*'. Since all categories of an entity are not equally important, we need to find the one that aligns best with input query. For a given query q_k with entity mention e_i that maps to category c_j, we score categories with Eq 2.

$$mc_i = \underset{c_j}{\text{argmax}} \frac{cos(Q_k, C_j) + jac(Q_k, C_j)}{2} \quad (2)$$

  where $cos(Q_k, C_j)$ is Cosine similarity between query vector Q_k, built from terms in the query and category vector C_j, built from its task dictionary terms. $jac(Q_k, C_j)$ is the Jaccard similarity between C_j and Q_k. For this work, we only consider query terms for calculating the best category for an entity. Since, Jaccard and Cosine are computationally quick to compute, we use the average of both metrics to find most likely category for an entity.

- For queries with multiple entities, we shall get terms from different categories. We aggregate terms from the best category of each entity in a single set. We represent this set as T, where $T = \cup_{mc_i} S_i$, and S_i is the term dictionary of category mc_i. Since we need only a few terms, we score terms $t_m \in T$ with respect to the query using Eq 3.

$$score(t_m, q_k) = tf(t_m, mc_i) \frac{\sum_{w_k \in Q_k} PMI(t_m, w_k)}{|Q_k|} \quad (3)$$

  where $PMI(t_m, w_k)$ represents the Pointwise Mutual Information between query word w_k, and task term t_m. $tf(t_m, mc_i)$ is the term frequency of term t_m in category mc_i. The objective of Eq 3 is to capture both the importance of a term given a certain category ($tf(t_m, mc_i)$) and the average likelihood (PMI) of that term occurring with the query terms. That is, a term should not only be important in the category but should also frequently co-occur with query terms.

We use the above method to rank terms both for term prediction and query expansion. To summarize, we adopt the following mechanism for evaluation:

Query Term Prediction: For each session, we use the first query to predict terms in subsequent queries of the session. We remove the overlapping terms between the base query and subsequent queries in the session to avoid scoring query terms twice during prediction. We also remove stop words from this list. We ignore those sessions where subsequent queries do not contain additional terms.

Query Expansion: For this work, we use the category task dictionaries to expand a user query. An input query is first tagged using Dexter, each entity is then mapped to a single category. The system ranks terms from these categories and uses the top scored terms for expansion. We choose top K terms to expand the query. Terms are ranked using the approach above.

We refer to our approach as **ENT** in the tables. To construct a dictionary of task phrases we use publicly available 2006 AOL query logs which consists of 20 mil search queries issued by over 657000 users within 3 months. We empirically determined the frequency thresholds to filter terms for tasks in each category. For each category, we retain terms with tf-Idf greater than 9. This was selected by manually sampling and inspecting term quality of some categories. Table 2 summarizes the resulting task phrase dictionaries. In the

[3]statistically significant values are marked with * and highest values are in bold

Table 3: Comparison of Retrieval results [3]

	2011 Session Track				2012 Session Track			
METHOD	**MAP**	**P@5**	**NDCG**	**NDCG@10**	**MAP**	**P@5**	**NDCG**	**NDCG@10**
No_Exp	0.1004	0.1759	0.3021	0.1662	0.1618	0.3247	0.3465	0.2402
HTC_50	0.103	0.1828	0.3071	0.1655	0.1624	0.3294	0.349	0.2453
HTC_100	0.1025	0.1862	0.3067	0.1674	0.163	0.3318	**0.3507**	0.2486
QCC_50	0.1048	0.1828	0.3068	0.1686	**0.1634**	**0.3341***	0.3483	**0.2502***
QCC_100	0.1036	**0.1897***	0.3069	0.1676	0.1631	0.3294	0.3495	0.2452
Ent	**0.1086***	0.1828	**0.3105***	**0.1709**	0.1609	0.3153	0.3464	0.2258

following section, we briefly explain the task extraction baselines used to compare our approach.

4.1 Task Extraction Baselines

As baselines, we use task identification approaches proposed in [5] to find tasks in a session. They refer to a set of (consecutive or otherwise) queries with same intent in a session as a single task. Lucchese *et al.* [5] explore several clustering methods to identify these tasks. We use QCC-wcc **(QCC)**, query clustering based on weighted connected components, a graph based algorithm to identify tasks. It builds a graph $G = (V, E)$, whose nodes V are queries in a session and edges E are weighted by the similarity of the corresponding nodes. The aim is to drop edges with low similarity, and to build clusters on the basis of the strong edges which identify the related query pairs. We also compare our approach with QCC-htc **(HTC)**, which clusters queries based on head-tail components, a variation of the connected components based algorithm, which does not need to compute the full similarity graph. We use the same similarity functions proposed in the paper for both the algorithms. We use similar parameter settings from [5] for both methods.

We use the following setup to compare our tasks with [5]. We begin by building a task index using tasks extracted by QCC and HTC on a large query log. This is done to improve real time task identification for a query. For an input query, we retrieve top K tasks. The system then extracts and ranks terms from these tasks based on their frequency in this set.

5. RESULTS AND DISCUSSION

The results of both query term prediction and query expansion are shown in Table 1 and Table 3 respectively.

Query Term Prediction: We use Session track 2011 and 2012 dataset with 145 sessions, with total of 456 terms, i.e. an average of 3 terms per session. The table shows precision values for term prediction at various cutoffs. For a given query, we retrieve top 50 and 100 tasks using QCC and HTC to extract terms. We follow the method in Section 4 to score terms from entity task dictionaries. Entity based task dictionaries (ENT) perform significantly better for 2012 sessions but do not outperform the baselines QCC and HTC for 2011 sessions.

Query expansion: We compare the both QCC and HTC with 50 and 100 tasks each to retrieve top terms. We also report results with no expansion- **No_Exp**. We varied the value of K and found K=25 to be ideal as addition of more terms did not affect the retrieval performance. The results indicate that task based query expansion is effective in improving performance. The results, however, indicate a mixed performance of our approach on 2011 and 2012 session dataset. While, we outperform the baselines on 2011 queries, for 2012, retrieval performance does not improve with expansion.

Table 4: Evaluation on 2012 Exploratory Queries[3]

	MAP	**Ndcg@10**	**Ndcg**
No_Exp	0.2108	0.3427	0.3873
HTC$_{100}$	0.2171	**0.3689***	0.3973*
QCC$_{100}$	0.2153	0.3593*	0.3949*
ENT	**0.2207***	0.3535*	**0.4025***

On manually inspecting expansion terms, we observed that task based query expansion is effective for queries exploratory in nature. Exploratory queries are more ambiguous in nature, thus adding terms from different tasks related to the query would improve performance. On the other hand, for specific queries, adding terms from different tasks will only harm retrieval. Hence, task specific expansion does not do well on specific queries. In 2012 dataset, there are 19 exploratory queries, for which our method outperform the baseline. The results are shown in Table 4. We shall perform further experiments to confirm this hypothesis.

6. CONCLUSION

In this work, we explored entity specific task extraction from search logs. We evaluated the quality of extracted tasks with Session track data. Empirical evaluation indicates that terms associated with entity oriented tasks can improve query expansion, especially for queries that are exploratory in nature. It can also predict subsequent query terms in a session.

7. REFERENCES

[1] D. Ceccarelli, C. Lucchese, S. Orlando, R. Perego, and S. Trani. Dexter: An open source framework for entity linking. In *ESAIR*, 2013.

[2] P. Ferragina and U. Scaiella. Tagme: On-the-fly annotation of short text fragments (by wikipedia entities). In *CIKM*, 2010.

[3] W. Hua, Y. Song, H. Wang, and X. Zhou. Identifying users' topical tasks in web search. In *WSDM*, 2013.

[4] M. Ji, J. Yan, S. Gu, J. Han, X. He, W. V. Zhang, and Z. Chen. Learning search tasks in queries and web pages via graph regularization. In *SIGIR*, 2011.

[5] C. Lucchese, S. Orlando, R. Perego, F. Silvestri, and G. Tolomei. Identifying task-based sessions in search engine query logs. In *WSDM*, 2011.

[6] C. Lucchese, S. Orlando, R. Perego, F. Silvestri, and G. Tolomei. Discovering tasks from search engine query logs. *ACM Trans. Inf. Syst.*, 2013.

[7] D. Milne and I. H. Witten. Learning to link with wikipedia. In *CIKM*, 2008.

[8] Y. Xu, G. J. Jones, and B. Wang. Query dependent pseudo-relevance feedback based on wikipedia. In *SIGIR*, 2009.