An Approach to Diversify Entity Search Results

Imène Saidi
University of Oran, LITIO Laboratory
BP 1524, El-M'Naouer, 31000
Oran, Algeria
saidi.imene@univ-oran.dz

Sihem Amer-Yahia
CNRS, LIG
Grenoble, France
Sihem.Amer-Yahia@imag.fr

Safia Nait Bahloul
University of Oran, LITIO Laboratory
BP 1524, El-M'Naouer, 31000
Oran, Algeria
Nait-bahloul.safia@univ-oran.dz

Abstract – Having named entities (person, country, company...) in response to users' queries is becoming more and more important in search engines. Indeed, in some cases users are not searching for a ranked list of documents, but for specific information (i.e. entities). In this work, we assume that users are interested in finding entities (e.g., name of a politician) and related entities (e.g., country of the politician 'x', name of the wife of the politician 'x'...) and the documents related to each entity. The user can then search entities by keywords or entities. Our goal is to return to the user diverse and relevant entities and documents. We then use the different types of an entity (Washington: city, person) and different categories of documents (Sport, Politics...) to diversify the results. In this paper, we develop a search semantics based on the types and categories of ranked results of entities. This new approach provides a variety of interpretations of relevant results. We conduct user studies to show the effectiveness of our approach and the quality of the results.

Keywords – Entity search, Diversification of search results, Indexing corpora, Information retrieval.

1. INTRODUCTION

Information retrieval as it is widely used today is not always suitable for some needs. In many different domains such as medicine and news, data is organized around topics, which are in the form of categories (health in the medical field, politics in news, etc.) or entities (name of a hospital, name of a politician, etc.). What users seek in this case is not a ranked list of documents, but information they contain (categories, entities) [1]. For example, in the medical context, users may be interested in discovering documents that contain entities related to a query, e.g., entities related to a specific disease. We name such entities: contextual entities (i.e. entities appearing in the context of the disease, such as the symptoms of a disease). In news, users may want entities related to the revolutionary wave that swept different countries (names of heads of state, countries, dates, places, etc.).

In this paper, we consider the problem of finding documents and entities related to user queries. Entities may or not be known to users, so users should have the choice to express their queries in various ways: Search With Entities (SWE) when the user knows entities and wants more related information (contextual entities and documents), KeyWord Search (KWS) when entities are unknown. Our work is made possible by the apparition of automatic annotation systems such as: Open Calais [2], Alchemy API [3] and Zemanta [4]. These systems annotate semi-structured or unstructured documents and attach rich semantic metadata to documents by categorizing them and extracting named entities they contain. Our proposition is to build a system for searching entities using the annotator and support different types of queries formed by entities or keywords and return related entities with documents of each entity (exploration by entity). Entities may have several types (Washington: city, person) and appear in documents belonging to different categories (Politics, Business ...). This presents two major challenges. The first challenge is to associate keywords or entities that constitute the query with different types of entities. The second is to use the identified types of entities and the categories of documents that contain them to

present the results in the best way to users. These challenges reflect an issue that has been presented in different works which is: diversity of query results. Indeed, the different interpretations of the same entity (e.g., city, person, organization ...) and the categories of documents (e.g., Politics, Medicine ...), can be used to diversify the documents to be returned.
In the next section, we present some related works to diversity. Section 3, is reserved for the description of the context of our work. In section 4, we present our approach and in section 5 some experiments. A conclusion and future work are proposed in the end of this paper.

2. RELATED WORK

In our work, we define a new problem of diversification which is different from the interpretation that has been given to it previously [5, 6, 7, 8]. Until now, diversity of query results was formulated as the problem of finding a set of documents relevant to the query that differ as much as possible from each other in their content. It has been proved that the problem of diversity of query results is NP-complete since it is to find a diverse subset of size N in a larger set. Thus, the problem is to define a threshold of relevance and calculate the sum of distances of content of the pairs of documents in the returned set (the distance may correspond for example to the cosine tf*idf vectors of documents). Several greedy algorithms for diversity [9, 10] have been developed. The most common is to find the N most relevant documents in a first step. The second step is to test if the replacement of one of the N documents by a new document certainly less relevant but whose relevance is greater than a threshold increases the diversity of the set of N documents. This phase is applied until the relevance of documents to be considered is no longer satisfying the threshold.
Diversity can be based on the meaning of a query (intent of the user) or the content of the returned results. It can be based on both too.
- Diversification based on meaning ([9, 11]) discusses the various possibilities of the user query (ambiguity) using probabilities on all disambiguisations of a query “the coverage of the query”. The goal is to return the most relevant results (close to the sense of the user).
- Diversification based on content ([6, 7, 8]) aims to reduce redundancy of information in the results. This is accomplished by avoiding documents that offer less information to users.
In our work, diversity is based on the types and categories used to annotate documents.

2.1. Contribution of our work

Previous works are based on the coverage of interpretations or the content of documents to diversify results and propose complex solutions to perform diversity. The goal of these solutions is to return a ranked and diversified list of documents. In our work, we use annotations to form diverse groups of documents using types or categories. We then use indexing and create various indexes to prepare data that allows us to address the complexity of the problem.
In this work, diversity is applied in two situations. In the first situation, contextual entities (related to our defined types of queries) are found using an index of entities. This is a kind of diversity of query interpretations, user does not know entities or do not give any precision of the query (e.g., the query is George Washington: is it George Washington president, George Washington University, George Washington Hospital ...). In the second situation, related documents of each entity are diversified and selected according to two conditions: either their relevance to the entity is above a threshold, or the type of the entity or the category of the document is unique (compared to other documents which are related to the same entity). In this case, an index of the categories and an index of entities that the documents contain are used. We can affirm that it is our definition of diversity that simplifies query processing.

3. CONTEXT AND PROBLEM DEFINITION

In this section, we give illustration examples for the different types of queries of our work. We end this section with our problem definition.

3.1 Illustration examples

We suppose that a user wants information about politics in America. The user can submit different queries (Figure 1).

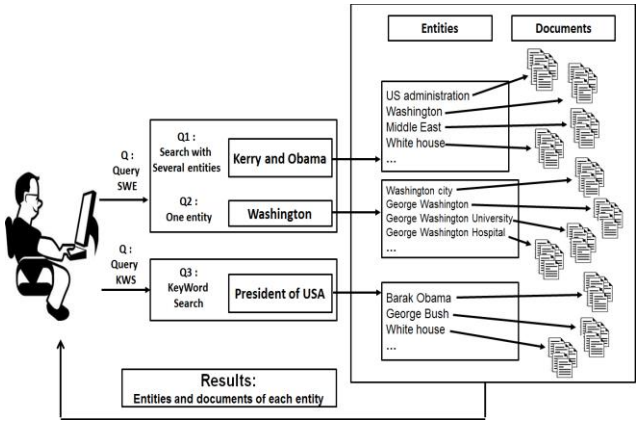

Figure 1: Illustration example

Scenario1. For the first query Q1, the user wants information about two politicians (he is looking for a link between the two personalities or for entities that are related to the two personalities): This case represents Search With entities (SWE). Results will be the related and diverse entities (contextual entities) that appear in the context of the query. The user can explore the documents of each entity.

Scenario2. The second scenario is a special case of SWE; it consists of searching with one entity. User wants information on a particular entity: in this case, information about "Washington". The user may want the city of Washington DC or he may search for the president George Washington, so we diversify the types of entities (diversification by type). Users may also be interested in composed entities (e.g., George Washington University, George Washington Hospital ...). We assume that it is more interesting to consider all these entities and return them to the user to increase the diversification of interpretations. We also assume that it is interesting to consider the different categories of documents related to an entity when it has only one type (e.g., if the query is George Washington University).

Senario3. For the third query Q3, the user wants to have information on a keyword query (KWS): e.g., president of USA. Results are the related entities and their documents. The results of each entity will be treated as the case of search with entity (diversification by type if the entity has several types or diversification by categories if the entity has one type).

3.2 Problem definition

We consider a query $Q = \{t_1 \dots t_n\} / ti \in E \cup K$, E is a set of entities and K a set of keywords.
The goal is to return entities and relevant documents organized by entity. We first explain how entities are found then we detail the score of documents that incorporates relevance and diversity.

3.2.1 Entity Search

Given the above query Q, the purpose is to find a set of entities $R \subseteq E$ related to the query Q and for each entity $e \in R$, classify the associated documents. To cover the different queries we consider, we define R as:

$$R = \begin{cases} entities\ of\big(TopK(Q)\big)\ if\ \forall\ ti \in Q, ti\ \in K \\ \text{related entities(Q)} \quad if\ \forall\ ti\ \in Q, ti \in E \end{cases}$$

where entities of(TopK(Q)) are the entities that appear in the Top K documents that answer the query Q (case KWS) and related entities(Q) are the entities that appear in the context of the query Q, i.e. entities that exist in the best documents that match the query Q when it consists of several entities (case SWE).
Indexes will help find documents related to entities. Indexes will be described in the Section 4.2.2.
R is a set of entities that answer the query extended by contextual entities that appear in the context of the query i.e. entities of the best documents that are related to entities of the query or entities of top k documents that answer the query. In a specific case, when the query is formed by one single entity, R is the set of entities composed of the entity of the query (entities that start, end or contain the entity of the query).

3.2.2 Finding Related Documents

For each entity $e \in R$, we identify a set $S = \{d_1 \dots d_m\}$ of documents to be returned.

A document d is returned with the entity e if a function we named *Diverse_type_cat* is true or if a function we named *Relevance* is true.

Diverse_type_cat is true in several cases:

- If the entity e has several types (diversification by type):

Diverse_type_cat is true if a type of entity e of the document d does not exist in *groupType.*
$groupType \subseteq e.types$; $e.types$ are the types of an entity e. *groupType* is updated by the types found for the entity e in the documents of the results. Formally:

$$For\ e.types\ > 1 \mid e\ \in R: \nexists\ type\ \in groupeType$$
$$type\ \in e.types \wedge (e,type) \in d.entities,$$

d.entities are the entities of a document d.
Example: entity e is "Washington", this entity has several types: 'person' and 'city'. For example, if the type 'city' does not exist in *groupType*, the document d that contains the entity of type 'city' will be selected in the result. This increases diversity of types, *Diverse_type_cat* is then true.

- If the entity has only one type (diversification by category):

Diverse_type_cat is true when the category c of a document d does not exist in a group named *groupCat* $\subseteq C$; C the set of the categories of the documents of the corpus. *groupCat* contains the

categories of results found for the entity *e*. Formally:

$$For\ e.types = 1\ and\ e \in d.entities \mid e \in R:$$
$$\nexists\ c \in groupCat\ /c = d.category, groupCat \subseteq C$$

Example: entity *e* is "Barack Obama", this entity has one type: 'person', so the different categories of documents are considered. For example, if the category 'business' does not exist in *groupCat*, the document *d* of category 'business' will be selected in the result.
This increases diversity of categories, *Diverse_type_cat* is then true.

When the document *d* is taken in *S*, *groupType* is updated by the type of the entity *e* found in *d* or *groupCat* is updated by the category of the document *d*.

Relevance is true if:

1. The document *d* answers the query *Q and* contains an entity $e \in R$. Formally:

$\exists\ ti \in Q \mid ti \in \{d.keywords \cup d.entities\} \wedge \exists\ e \in d.entities \mid e \in R$.

2. The score of the document *d* for a query *Q score(d,Q)* exceeds a threshold. *score(d,Q)* is the sum of scores *score(d,e)* of the same entity *e* in the document *d* when the entity has several types (diversification by types), or it is the score of the category of the document *score(d,c)* when the entity has only one type (diversification by category). This definition ensures a maximum of diversity with relevance of the selected documents, when the documents $d \in D$ are pre-ranked according to their relevance. Formally:

score(d, Q) $> \theta$ where θ is the minimal threshold of relevance (selected by experiments).

score(d,Q)=
$$\begin{cases} \sum score(d,e) \in d.entities\ if\ e.types > 1 \mid e \in R \\ score(d,c) \in d.category\ if\ e.types = 1 \mid e \in R \end{cases}$$

Choosing a document to be taken in *S* is made according to *diverse_type_cat* to have all various types of an entity or the categories of documents, even if the score of the corresponding document does not exceed the relevance threshold (because the document is unique). The choice is made according to *Relevance* to have the most relevant documents (score is greater than a threshold) on the same type of entity e or on the same category *c*. This means that if a document d is selected by checking *diverse_type_cat*, its type or its category is new in the collection to be returned to user (to increase diversity) or the document is relevant to the type or the category.

4. DIVERSIFICATION OF ENTITY SEARCH RESULTS

For some data sources such as forums and news articles, we assume that it is more interesting to interpret user queries by the entities that sources contain. Entities may have several types and documents have different categories, we exploit that to diversify results. In this section, we summarize our approach in a conceptual architecture and we describe the offline processing.

4.1 System Architecture

The following Figure (Figure 2) shows the conceptual architecture of our system and summarizes our approach.

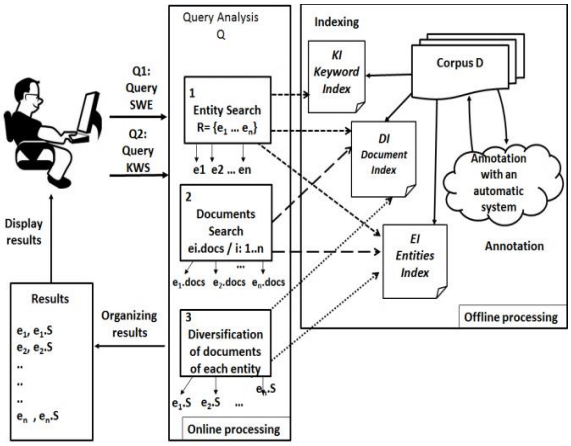

Figure 2: System architecture

We consider a corpus D of semi structured documents (forums, newsgroups, etc.). Our approach is to make an offline processing to prepare ad-hoc indexes for online processing (Figure 2):

- We start by annotating the corpus using an automatic annotation system (Open Calais API) to extract entities, types and categories of documents with their scores.
- We create different indexes to store information, i.e. keywords, entities, types, categories and scores. The scores are: score(tf*idf) of a term (case KWS query), score(d, e) of entity and score(d,c) of category. Three indexes are created: an inverted index for Keywords (KI: Keyword Index), an inverted index for types of entities (EI: Entities Index) and an index for entities and categories of

documents (DI: Document Index) i.e. what are entities of a document and what is its category.

Query processing is done in 3 steps: Entity search, Document search and Document diversification per entity.

4.1.1 Entity search

In online processing, we use our indexes to interpret queries using the entities.

Two types of the queries are considered: Search With Entities (SWE: Query Q1) and KeyWord Search (KWS: Query Q2).

SWE queries: the idea is to diversify interpretation by searching entities (Entity Search in Figure 2) to return a set of entities R = {e1, e2... en} such as:

- R are the entities that appear in the best documents that contain the entities of the query, if the query is of type SWE.
- A special case of this type SWE could be search by one entity, R is then equal to the entity itself extended by the composed entities, i.e., entities that start, end or contain the entity of the query.

KWS queries: if the query is of type KWS, we use Top K query processing algorithm of Fagin et. al [12] to have a ranked list of documents. From this list, entities are extracted using index DI and returned as the set R. We suppose that entities that appear in the best documents that answer the query are relevant or contextual (appear in the context so could interest the user). If the query is a mixture of entities and keywords, it is considered as a keyword query (KWS).
We assume that when the user searches with entities (SWE), he is looking for a relationship or wants to make a comparison between entities of the query (e.g., Sarkozy and Merkel, Renault or Peugeot, infection and tumor...).

4.1.2. Document search

After finding the set R of entities, the related documents to each entity {e_1.docs, e_2.docs... e_n.docs} are found (searching documents in Figure 2). Indexes EI and DI are used.

4.1.3. Document diversification per entity

After finding the documents of each entity of R, they are diversified:

- If the entity has several types: diversification of documents is made according to the type of entity, "at least one document" of each type must be returned to the user to ensure maximum diversity, other documents will be selected according to relevance, i.e., their score(d, e) must exceed a threshold.
- If the entity has only one type: diversification of documents is made by categories of the related documents to the entity, "at least one document" of each category must be returned to the user to ensure maximum diversity, other documents are selected according to relevance, i.e., their score(d, c) must exceed a given threshold.

The condition "at least one document" ensures that the unique documents (unique type of entity or unique category of document) will be considered, even if their score is not high (does not reach the threshold of relevance). This maximizes diversity.
After organizing the results, we return to the user a set of entities {e_1, e_2... en} and a set of documents for each entity {e_1.S, e_2.S... e_n.S}.

4.2 Offline processing

The corpus of documents D is preprocessed in an offline phase in order to create the necessary indexes.

4.2.1 Document annotation

With the advent of automatic annotation systems, it is possible to attach semantically rich metadata to documents. That allows to categorize documents and find entities they contain (people, places, organizations, etc.). In this work we used Open Calais API [4] to annotate a corpus of files (semi or unstructured), to find existing named entities, categories of documents and scores. Annotation of corpus is an important step in our approach; it prepares necessary information for online processing.

4.2.2 Indexing

Necessary indexes are:

- KI (Keyword Index): An inverted index that matches each keyword k with the documents that contain it and a score (score of k to a document d). This index is necessary for the application of Top K processing and will be used for the KeyWord Search (KWS queries).

k: {*(d, tf.idf(k,d))* }, *tf.idf(k,d) = score(d, k) / tf.idf* (term frequency - inverse document frequency).

Example: president: {(14, 9), (57, 3), (84, 18)}.

- EI (Entity Index): To build this index, the API Calais is used for annotation (Named

entities extraction with types and scores). The inverted index EI stores the types of the entities e with documents containing them and the score (score of the type of the entity *e* in the document *d*, i.e. score(d,e) which is calculated by the automatic annotation system). This index is used to retrieve documents related to entities in the case SWE and documents of each entity *e* of the set *R*. *e: { (d, type, score(d,e))*}.

Example: Barack Obama: {(575, person, 0.332), (810, person, 0.341), (881, person, 0.331)}.

- DI (Document Index): This index contains the entities of a document and its category. Open Calais is also used for the extraction of the categories of documents with their scores. This index maps each document with its category and a score (score(d,c)). It will be used to find the categories of documents in the case of diversification by categories and entities of a document.
  d: { *({e}), (c, score(d,c))* }.

Example: news.xml: {{Washington, white house ...}, (politics, 0.91)}.

5. EXPERIMENTS

We conduct a set of experiments to demonstrate the usefulness of our proposed approach and the quality of results. First, using a set of Reuters' articles, we demonstrate that the generated results are relevant to users. Second, we show that users prefer our proposed approach (over 60% of cases) to a classical approach that returns a ranked list of documents with snippets.

Implementation setup: we have implemented a Java prototype reflecting the processing of our approach. The prototype can handle user queries. It offers to the user a choice between searching with entities and keyword search. Different results are returned, i.e. entities and relevant documents to the found entities. Documents are diversified to increase both the relevance and diversity of results. All experiments were conducted on an Intel Core i5 workstation with a speed of 2.5 GHz and 4GB Memory running Linux Ubuntu 13.04.

Datasets: we conducted our experiments on two different datasets: a collection of newsgroups (20NewsGroups) and a collection of Reuters articles (Reuters-21578).

20NewsGroups: the 20 Newsgroups data set is a collection of approximately 20,000 newsgroup documents, partitioned across 20 different newsgroups (about 1000 messages per group).

Collection of Reuters: this corpus is a collection of texts downloaded from the website of the Institute of Computer and Electronic Gaspard-Monge3. This collection of texts was extracted automatically from Reuters-21578 collection4. This categorization is given in a file named categorisation.txt where each line corresponds to the categorization of text. This dataset contains about 20,000 file. This corpus was also chosen for its richness and variety of categories.

5.1 Relevance

To verify the quality of results produced by our approach, we conduct another user study on the relevance of the found entities (the set R of entities) and the returned documents (diversified documents). For this test, we used the corpus Reuters (also chosen for its richness in entities and categories).

5.1.1 Relevance of entities

We asked users (10 students) to identify among a set of returned entities (R), the number of relevant entities and the number of contextual or composed entities (when the query is composed of one entity). Different queries have been submitted to cover the different types of queries (5 queries for each type: search with entities, search with one entity and KeyWord Search).
Table 1 presents the results of search with entities (SWE).

Table 1: Queries of SWE

Themes	***Number of found entities "R"***	***Relevant entities***	**Contextual entities**
Ford and Chevrolet	2	83.33%	16.66%
Lincoln and Bush	3	66.66%	11%
Infection and tumors	8	50%	33%
Volvo USA	26	37.15%	25.61%
Washington and Baghdad	47	26.23%	27.65%

In Table 1, we calculate the average of relevant entities and the average of contextual entities using the votes of users, we then calculate the percentage of relevant entities and the

percentage of contextual entities, the rest are insignificant entities i.e. entities that appear in the context of the query but are not related. For the first query "Ford and Chevrolet", the number of the found entities is 2, this means that *R*= {Ford, Chevrolet}. No other entities were found, so most users (83.33%) consider these 2 entities as relevant and 16.66% as contextual because they were looking for other results related to these entities of the query. In this case, there are no insignificant entities[1]. We can notice that for all queries, at least 26% of returned entities are relevant and more than 25% are contextual. In some cases, the percentage of non-significant entities exceeds 40% (query "Washington and Baghdad"), this is due to the nature of the query which is general and not specific (the meaning of this query differs from a user to another). Relevance in this case is relative. Table 2 presents the results of search with one entity.

Table 2: Queries of search with one entity

Themes	*Number of found entities "R"*	*Relevant entities*	Composed entities
Chevrolet	6	72.16%	16.66%
Lincoln	10	36.36%	48.48%
America	15	36.4%	42.10%
Ford	25	57%	35%
Cancer	74	31.03%	56.31%

In Table 2, we calculate the average of relevant entities and the average of composed entities using the votes of users and then calculate the percentage of relevant entities and the percentage of composed entities. We can notice that for all queries, at least 31% of returned entities are relevant and more than 16% are composed entities. The percentage of non-significant entities is negligible (less than 15%) except for the query "America" which is a general query. Entities related to this query (composed entities) appear in several categories and different types, (e.g., America Online, America West Airlines, South America, etc.). So entities related to "America" are very different from each other, some entities are then non-significant (21.06% of entities, see Table 3) because they don't match with the different needs of users.

Table 3 presents the results of KeyWord Search.

Table 3: Queries of KWS

Themes	*Number of found entities "R"*	*Relevant entities*	Contextual entities
Ford car	14	45.21%	28.57%
Patient disease	35	39.02%	22%
Car dealer	36	31.47%	20%
Buy Ford	50	29%	29.32%
Prime minister	80	27.08%	35.82%

In Table 3, we calculate the average of relevant entities and the average of contextual entities using the votes of users and then calculate the percentage of relevant entities and the percentage of contextual entities. For all queries, at least 27% of returned entities are relevant and more than 20% are contextual entities. The quality of results is less than the two previous types of queries because interpretation of keywords is more complex than entities, improvement of this part will be considered in future work as well as filtering insignificant entities of the two previous types of queries.

5.2 Usefulness

We evaluate the usefulness of our approach of diversification of entities and documents. We aim to analyze whether users prefer our proposition of interpreting users queries using the entities of the sources and organizing the documents of each entity, against the simple approach that returns ranked list of documents using Top K processing. In this simple approach, the score of a document is its TF*IDF, the k best documents (with the highest scores) are returned to users. We want to know if for some domains our approach is more useful to the user than the classical approach so we used the corpus 20 Newsgroups (chosen for its richness in categories). In this corpus, data is organized into 20 different newsgroups, each corresponding to a different category. We submitted different queries. Table 4 summarizes the results of this test.

[1] Entities that are not relevant or contextual are insignificant.

Table 4: User's preference: Our approach VS Classical approach

Themes	*Query*	*Our Approach*	Classical Approach
Computer	Disk drive	70%	30%
Business	PC speaker	80%	20%
Recreation	Coach of the year	60%	40%
Politics	Prime minister	70%	30%
Science	Infections and tumors	60%	40%
Religion	Catholic university	50%	50%

We achieve a survey by presenting to users the results of our approach and the results of the classical approach using queries of different domains (Table 4). We ask users to indicate their preference by choosing the results of the first or the second approach. From the results, it is clear that users prefer our approach which is more informative, to the classical approach, thus confirming our motivation. Our approach is more interesting to users specifically for domains where documents are focused on a specific category and contain many entities (computer, politics). For literary documents (religion), entities are less useful. We also observe that when the user's knowledge is limited on a subject, our approach brings more novelty by presenting entities that may appear in the context of the search.

6. CONCLUSION

In this work, we presented an approach for diversifying search results that leverages differences between entities and documents, i.e. types of entities and categories of documents. In our work, we exploit the different categories and types of annotations extracted by automatic systems and previously stored in indexes. This allows to by-pass the complexity of the problem of diversity known as an NP-complete problem. The definition of diversity in our work allows to index processing and simplify the complexity of queries. To facilitate query processing, a pre-processing is done offline on the corpus of documents to index necessary information.
The idea is to interpret keyword queries (KWS) or the queries composed of entities (SWE) by relevant entities (entities that exist in the documents that match the query). We also propose to organize and diversify the documents of each returned entity by the various types of entities if the entity has several types or categories of documents if the entity has only one type. Users will then have a list of relevant entities and the possibility to explore the documents of each entity. Experiments show the effectiveness of our approach and the quality of results. Our approach is easier for exploration of results, clearer, and in general more helpful than simple relevance ranking of documents.
For a continuation of this work, we will improve and extend the annotations part by the use of other annotation systems such as Alchemy API ([4]). We also plan to improve our algorithm to filter insignificant entities and to consider the large scale problem.

7. REFERENCES

[1] T. Cheng, X. Yan, and K. C.-C. Chang. Supporting entity search: a largescale prototype search engine. In SIGMOD Conference, pages 1144-1146, 2007.

[2] Open Calais, http://www.opencalais.com/documentation/calais-web-service-api.

[3] Zemanta, http://developer.zemanta.com/.

[4] Alchemy api, http://www.alchemyapi.com/products/features/entity-extraction/.

[5] L. Qin, J. X. Yu, and L. Chang. Diversifying top-k results. PVLDB,5(11):11241135, 2012.

[6] A. Angel and N. Koudas. E_cient diversity-aware search. In SIGMOD Conference, pages 781792, 2011.

[7] C. Yu, L. V. S. Lakshmanan, and S. Amer-Yahia. It takes variety to make a world: diversi_cation in recommender systems. In EDBT, pages 368378, 2009.

[8] Y. Zhang, J. Callan, and T. Minka. Novelty and redundancy detection in adaptive filtering. In SIGIR 02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, pages 8188, 2002

[9] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In WSDM, pages 514, 2009.

[10] S. Gollapudi and A. Sharma. An axiomatic approach for result diversification. In WWW, pages 381–390, 2009.

[11] K. Liu, E. Terzi, and T. Grandison. Highlighting diverse concepts in documents. In SDM, pages 545–556, 2009.

[12] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. J. Comput. Syst. Sci., 66(4):614656, 2003.