Context-Sensitive Auto-Completion for Searching with Entities and Categories

Andreas Schmidt
Karlsruhe Institute of Technology &
University of Applied Sciences
Karlsruhe, Germany
andreas.schmidt@kit.edu

Johannes Hoffart
Max Planck Institute for Informatics
Saarbrücken, Germany
jhoffart@mpi-inf.mpg.de

Dragan Milchevski
Max Planck Institute for Informatics
Saarbrücken, Germany
dmilchev@mpi-inf.mpg.de

Gerhard Weikum
Max Planck Institute for Informatics
Saarbrücken, Germany
weikum@mpi-inf.mpg.de

ABSTRACT

When searching in a document collection by keywords, good auto-completion suggestions can be derived from query logs and corpus statistics. On the other hand, when querying documents which have automatically been linked to entities and semantic categories, auto-completion has not been investigated much. We have developed a semantic auto-completion system, where suggestions for entities and categories are computed in real-time from the context of already entered entities or categories and from entity-level co-occurrence statistics for the underlying corpus. Given the huge size of the knowledge bases that underlie this setting, a challenge is to compute the best suggestions fast enough for interactive user experience. Our demonstration shows the effectiveness of our method, and its interactive usability.

1. INTRODUCTION

Motivation. Searching in document collections by means of entities and semantic categories allows users to specify more precise search queries and improve retrieval effectiveness (e.g., [3, 9, 7]). The underlying assets, large knowledge bases (KBs) with entities organized in semantic types or categories (e.g., DBpedia YAGO or Wikidata), as well as methods for linking textual occurrences of entities to these KBs [14] are sufficiently mature to make this endeavor practically viable.

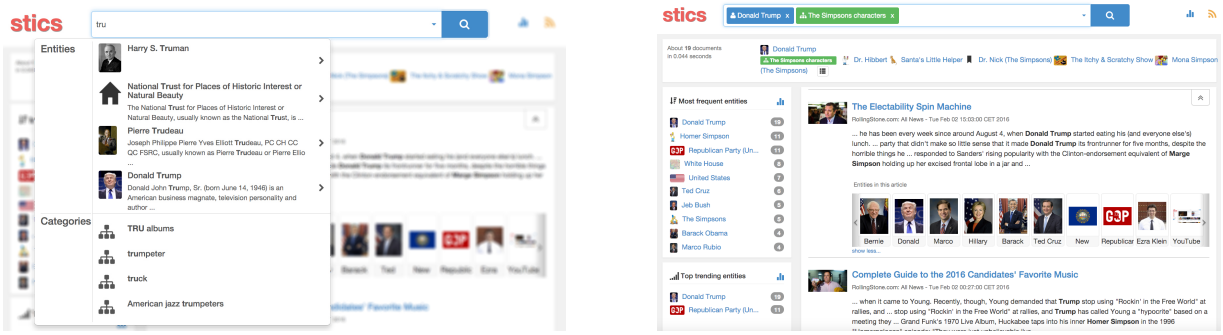
As an example, consider the screenshots of the STICS system [9] for news search, depicted in Figure 1. The user merely needs to start typing a name, and the system automatically suggests entities and categories that could be good completions of the input prefix. After making her choice (the politician Donald Trump in this case), the user may then select an entire category as a search criterion (here, the Simpsons characters), again after merely typing a prefix. The system will automatically match entities from that category and return documents with co-occurrences of the specified entities. Note that this interactive setting differs from traditional entity search where users merely enter ambiguous names and keywords [1].

Problem. Matching the user's input prefix against names of entities and categories (by prefix matching of individual tokens) often returns hundreds or thousands of auto-completion candidates. So the ranking of these candidates is crucial. A simple solution is to use the importance of entities as a criterion. However, the importance in the KB (e.g., derived from in-coming links in Wikipedia) is not necessarily in line with the frequency or prominence of an entity in the corpus. A good ranking thus needs to be *adaptive to the corpus*. In addition, the ranking should reflect the incremental nature of the user's input: after the first entity is chosen, the suggestions for the second one should be *sensitive to this context*. For example, after the user picked Donald Trump, an input prefix like "Sim" should not exactly prioritize famous singers like Paul Simon or Nina Simone, who are important but totally unrelated to Trump. Note that this incremental interpretation of the user's keystrokes and make this problem quite different from the traditional tasks of query segmentation (e.g., [8]) or entity linking for very short texts (e.g., [6]).

Solution. This demo paper presents a system that provides auto-completion suggestions for entities and categories in a corpus-adaptive and context-sensitive manner. Our solution builds on ideas from prior work on auto-completion (e.g., [4, 2]), but extends them to the underexplored realm of knowledge-driven search with interactive speed and usability. For corpus-adaptivity, the ranking of candidate entities is based on the importance in the underlying document collection, also taking into account the temporal dimension. For context-sensitivity, our ranking considers the semantic relatedness to other entities and categories already chosen by the user for her input so far. As basis for our solution we use the previously developed STICS [9], which ranks entity and category suggestions independently of the given context, purely based on statistics derived from the KB, not by statistics gathered form the document collection. Our demo is available online at **http://stics.mpi-inf.mpg.de**.

SIGIR '16, July 17-21, 2016, Pisa, Italy

DOI: http://dx.doi.org/10.1145/2911451.2911461



(a) Conventional Suggestion (b) Adaptive Suggestion

Figure 1: STICS news search with auto-completion suggestions for entities and categories.

2. KNOWLEDGE-DRIVEN SUGGESTIONS

The goal of the knowledge-driven auto-completion is to suggest, for a given prefix and specified entites and categories, only entities and categories that lead to non-empty results for the document collection being searched. Thus the suggestions depend on the retrieval model. In our system we retrieve documents that contain all specified entities and at least one entity of each category. Additionally, suggestions should be ranked so that entities and categories that are salient with respect to the collection are available for quick selection by the user.

Without any context, we should suggest entities and categories occurring frequently in the collection, this guide the user to useful queries for the collection. As soon as context is given in the form of entities or categories, there are multiple possibilities of what to suggest, and how to rank the suggestions. Given a category, an intuitive strategy could be to suggest categories that co-occurr in documents. However, categories are only present indirectly in the document, via the entity. Thus, we suggest categories that are associated with entities co-occurring with entities of the given category, in the same indirect manner.

More formally, the auto-completion should provide suggestions based on the input data (E, C, p), where E is the set of entities and C is the set of categories already specified by the user (both might be empty), and p is the current prefix that the user has already typed. An additional input is the type relation from the KB, which associates each entity with a set of categories (e.g. the entity `Hillary Clinton` has the categories `politician` and `lawyer`). The output is a tuple (E^*, C^*), containing two ordered lists of suggested entities and categories. For the actual ranking, we distinguish two cases:

E and C are empty: Without any KB context, E^* and C^* simply contain all entities and categories ranked by the global document frequency, where the frequency of a category is counted whenever an entity of the given category is present in a document. The more frequent an entity or category, the higher in the list. Both lists are then filtered by p to produce the final output.

E or C are non-empty: First the set of all context entities E_c is formed as the union of all entities in E and all entities in any category $c \in C$. E^* is then populated with all entities co-occuring with any $e \in E_c$, ranked by the maximum relatedness score (see Section 3). E^* is then filtered by p.

C^* is derived from E^* by adding for all of the categories of each $e \in E^*$. C^* is ranked by the sum of the relatedness scores of the contained entities, then filtered by p.

The filtering of entities and categories based on one or more prefixes is done in the following way: On average, every entity in the YAGO system is described by 2.7 words (category: 3.8 words). To match the given prefix(es), a prefix match between a single words describing the entity or category is done. If more than one prefix is given, each prefix must match at least one word from the description of the entity or category. The "coverage" of the prefixes is used as ranking criterium in a linear combination with the score described above.

Our system also supports "time travel" search. In this case, we are only interested in results appearing during an interval of time or at a specific point in time. This can easily be achieved by further filtering the set of documents that should be considered in a query. Restricting the news articles to consider automatically restricts the suggestions.

3. RELATEDNESS MEASURE

The relatedness between two or more entities is based on the distance of their occurrences in documents. The first step is to extract tuples of two or more entities occurring in a window of words of predefined size (e.g. 50 words). This is applied to all documents in the collection, and we derive an aggregated relatedness measure as follows.

Per document, each extracted n-tuple has a weight w that decreases with the maximum distance d between any two entities in the tuple: $w = \log \frac{1}{d}$. Across the entire collection, these weights are summed up over all documents where the n-tuple occurs.

4. INTERACTIVE PERFORMANCE

The auto-completion suggestions are always used in an interactive manner, where response times of below 100ms should be achieved. Consequently, a naive approach where in a first step all documents satisfying the previously entered entities and categories are retrieved, then co-occurring entities are extracted from this document set, is far to expensive. Thus we precompute the relatedness scores of all co-occurring entities. The starting point for building a datastructure supporting the suggestion lookup are the n-tuples

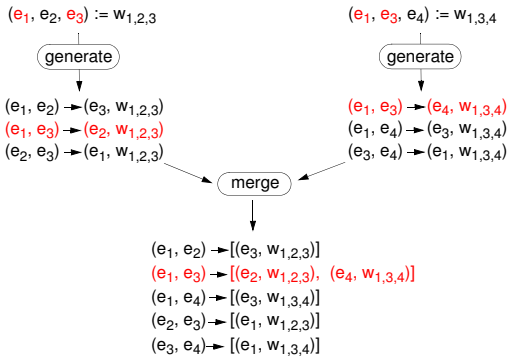


Figure 2: Data structure for generating entity suggestions.

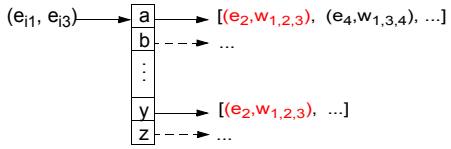


Figure 3: Splitting of entity-list along prefixes.

of entities extracted in Section 3. Consider for example the triple (e_1, e_2, e_3) with weight $w_{1,2,3}$. The existence of this triple states that there is at least one document in the collection in which the entities e_1, e_2, and e_3 occur.

Having a query with already specified entities e_1 and e_3, we can suggest entity e_2, as we know that there is at least one document satisfying this query. Having another precomputed triple (e_1, e_3, e_4) we can further suggest e_4. The construction of the suggestion-datastructure is shown based on the two tuples (e_1, e_2, e_3) and (e_1, e_3, e_4) in Figure 2.

For every n-tuple $(e_1, ..., e_n)$ with weight $w_{1,...,n}$, from the relatedness measure, we generate a dictionary consisting of n entries, where the key is formed by $n-1$ entities and the value is the entity which is missing in the key, together with the weight $w_{1,...,n}$ of the origin n-tuple. This is done for all tuples from the entity-indexing step. After that, the dictionaries are merged, so that the values become lists, consisting of weighted entities.

To reduce the number of entities which must be filtered by prefix we split the value-list by the prefixes of the entities. Figure 3 shows the additional datastructure for key-entry (e_1, e_3) which splits the entries by a one-character prefix. The reason for $(e_2, w_{1,2,3})$ occurring twice is that the description of the entity consists of multiple words. The example shows one word starting with 'a' and another one with 'y' (e. g. Yello Air Taxi). The length of the character prefix can be varied to split entity lists that are too long into lists of shorter length that meet the performance requirement.

5. IMPLEMENTATION

For the implementation we use a MySQL 5.6 database with MyISAM tables. The core of the database are the tables storing the entity co-occurrence tuples, which are queried by dynamically created SQL statements. The structure of these table contains $n-1$ columns for the entity-ids (in ascending order) and two more columns for a related entity and the weight of this relationship, which is computed as explained in Section 3. On our demo dataset of more than 3 million documents nearly all queries can be answered in less than 100 ms. Some rare queries for one or two given entities with very high frequency (like `United_States`) and very short and popular prefixes (i.e. `'s'`, `'c'`) have runtime durations which are in the order of a second. However, as there are only a few thousand combinations like this, we generated another table which combines the critical entities and the corresponding prefixes. This way, the time constraint of 100 ms can be achieved.

The temporal adaptivity is integrated by splitting the entries in the co-occurence tables in slices of one month. This way, queries with time constraints only have to consider the entries satisfying the given time constraint.

6. DEMO SCENARIO

6.1 Dataset

As a demo dataset we use 3 million news articles from about 300 different news feeds which we have collected since mid 2013. In this dataset, about 57 million mentions have been linked to the YAGO [15] using the AIDA [10] entity linking system. YAGO contains 10 million entities, organized in over half a million categories organized in a taxonomy. For our data, about 600,000 distinct YAGO entities are marked up in documents. On average, a document contains 28 mentions of 9.5 distinct entities.

6.2 Auto-Completion Suggestions

Corpus-adaptive suggestions help users to explore corpora that they are – a priori - not familiar with. Consider a user searching for news about `Hillary Clinton`. If the suggestions were solely based on conventional ranking measures from the KB, typing "Cl" might lead to `Cleveland` or `Bill Clinton`. With our corpus-aware and time-adaptive suggestions, `Hillary Clinton` is now first, as she is featured prominently in news due to her presidential campaign.

Context-sensitive suggestions help to users to find entities that are specifically relevant for the user's current task. Consider again the user interested in `Hillary Clinton` and suppose the user subsequently types the prefix `'sa'`. In 2013 the top three suggestions are `Saudi Arabia` and `San Francisco`, reflecting that she was still the Secretary of State. This changed in 2015, as shown in Figure 4: now the top suggestions are `Bernie Sanders` and `Rick Santorum`, also candidates for the US presidential election.

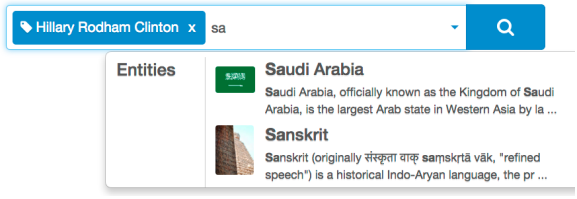
Anecdotal examples for the improvements by corpus- and context-adaptivity are shown in Table 1.

7. RELATED WORK

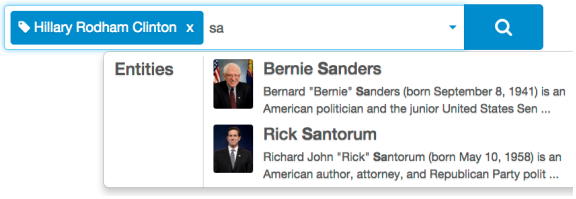
A related field of research is entity search (aka. expert retrieval) [1], which, however, has a different computational model: queries are keywords or phrases, and answers are lists of entities. In our setting, the user input is a set of entities and the search engine returns a set of documents.

Entity recommendation is another related topic (e.g., [5, 13, 11]). Here the task is to identify related entities in the context of a user exploring a knowledge base or using a search engine. These recommendations are meant to guide the user; they are not intended to map the user's input to best fitting entities. In contrast, our focus is on interactive auto-completion at real-time speed.

Context-sensitive auto-completion has been well studied for keyword queries over web pages and text documents (e.g.



(a) conventional Suggestion



(b) Adaptive Suggestion

Figure 4: Improving query formulation by adaptive auto-completion suggestions.

Input		Suggested entities		Suggested categories	
Prefix	Context	Conventional	Adaptive	Conventional	Adaptive
"do"	e: `Hillary Clinton`	Copa **do** Brasil Rio Grande **do** Sul Sport Club **do** Recife	**Do**nald Trump Patti Solis **Do**yle United States **do**llar	**do**cuments legal **do**cuments **do**cumentaries	**d**onor **do**ctor L.A. **do**dgers owners
"co"	c: `president`	**co**ld war **co**lumbia **co**nservative	Nationalist **co**ngress Party Indian National **co**ngress African National **co**ngress	dramatic **co**mposition **co**ding system physical **co**ndition	**co**untry **co**untries in Europe **co**mmunist states
"c"	c: `president` e: `Syria`	**C** (programming) **C**roatia **C**hurch of England	David **C**ameron Bill **C**linton **C**NN	English **c**ricketers dramatic **c**omposition **c**hampionship	Central Asian **c**ountries Near Eastern **c**ountries Former British **c**olonies

Table 1: Conventional vs. adaptive auto-completion suggestions for given Prefix, Entity (e), and Category (c).

[4, 2, 16]). Here, the context is given by previous queries, where in our case the context is given in a single query itself by multiple entities or categories.

Prior work on entity-level auto-completion includes the STICS [9], SEMEX [12] and Broccoli [3] systems. STICS is our baseline; its auto-completion is solely based on global importance. SEMEX is solely based on string similarity between the user input and the entity suggestions; there is no consideration of the relatedness between entities. Broccoli is closest to our setting; its auto-completion works for entities, categories, relations, and words or phrases. It is corpus-aware, but does not consider the temporal dimension of how importance and relatedness vary over time. Also, its context model currently only uses co-occurrence counts based on Wikipedia and Freebase.

8. REFERENCES

[1] K. Balog, M. Bron, and M. de Rijke. Query Modeling for Entity Search Based on Terms, Categories, and Examples. *ACM Transactions on Information Systems*, 29(4), 2011.

[2] Z. Bar-Yossef and N. Kraus. Context-sensitive query auto-completion. In *WWW 2011*, 2011.

[3] H. Bast, F. Bäurle, B. Buchhold, and E. Haußmann. Semantic full-text search with broccoli. In *SIGIR 2014*, 2014.

[4] H. Bast and I. Weber. Type less, find more: fast autocompletion search with a succinct index. In *SIGIR 2006*, 2006.

[5] B. Bi, H. Ma, B. P. Hsu, W. Chu, K. Wang, and J. Cho. Learning to recommend related entities to search users. In *WSDM 2015*, 2015.

[6] M. Cornolti, P. Ferragina, M. Ciaramita, H. Schütze, and S. Rüd. The SMAPH system for query entity recognition and disambiguation. In *ERD 2014*, 2014.

[7] J. Dalton, L. Dietz, and J. Allan. Entity query feature expansion using knowledge base links. In *SIGIR 2014*, 2014.

[8] M. Hagen, M. Potthast, A. Beyer, and B. Stein. Towards optimum query segmentation: in doubt without. In *CIKM 2012*, 2012.

[9] J. Hoffart, D. Milchevski, and G. Weikum. STICS: searching with strings, things, and cats. In *SIGIR 2014*, 2014.

[10] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum. Robust Disambiguation of Named Entities in Text. In *EMNLP 2011*, 2011.

[11] J. Lee, A. Fuxman, B. Zhao, and Y. Lv. Leveraging knowledge bases for contextual entity exploration. In *KDD 2015*, 2015.

[12] J. Osterhoff, J. Waitelonis, and H. Sack. Widen the peepholes! entity-based auto-suggestion as a rich and yet immediate starting point for exploratory search. In *GI 2012*, 2012.

[13] R. Reinanda, E. Meij, and M. de Rijke. Mining, ranking and recommending entity aspects. In *SIGIR 2015*, 2015.

[14] W. Shen, J. Wang, and J. Han. Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions. *IEEE Trans. Knowl. Data Eng.*, 27(2), 2015.

[15] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A core of semantic knowledge. In *WWW 2007*, 2007.

[16] S. Vargas, R. Blanco, and P. Mika. Term-by-term query auto-completion for mobile search. In *WSDM 2016*, 2016.