KCF.js: A Javascript Library for Knowledge Cards Fusion*

Haofen Wang
East China University of Science and Technology
130 Meilong Road
Shanghai,China
whfcarter@ecust.edu.cn

Zhijia Fang
East China University of Science and Technology
130 Meilong Road
Shanghai,China
fzjacky@mail.ecust.edu.cn

Tong Ruan
East China University of Science and Technology
130 Meilong Road
Shanghai,China
ruantong@ecust.edu.cn

ABSTRACT

Recently Web search engines have built knowledge graphs to support entity search and to provide structural summaries called *knowledge cards* for entities mentioned in queries. Different knowledge cards might be complementary or even have conflicts on values of the equivalent property. Thus, it is essential to achieve a more comprehensive fused card from those individual cards representing the same entity. In this paper, we present a system with technical details of card disambiguation, property alignment, value deduplication and card ranking to fuse knowledge cards from various search engines. We further develop a Javascript library called KCF.js based on the card fusion engine and demonstrates its usability via three possible applications.

1. INTRODUCTION

With the prevalence of entity search, a large portion of Web queries are to search entity related information. To support the ever growing information needs, search engines leverage public available knowledge bases such as Wikipedia[1] and Freebase[2] to build their own knowledge graphs. When submitting a query to Google (Bing or Yahoo!), the engine will provide a structured summary called *knowledge card* describing attributes of the given entity and relations with other entities. Since a query might be ambiguous, it could correspond to several real-world entities. Therefore, for a given query, a search engine may return zero to several knowledge cards. Google returns three cards (i.e., "Fox Broadcasting Television Network", "Fox News Channel" and "Fox (Animal)") for the query "Fox" while Bing returns five cards with two additional cards named "Fox Sports" and "Fox Interactive". Even though the two cards represent the same entity, some property may just appear in one card. For example, Google is the only engine to give an attribute named "Daily sleep" in the card describing "Fox (animal)". Moreover, values of the equivalent properties from different cards might have different expressions or conflict with each other. The scientific name of "Fox (animal)" is expressed as "Vulpini" in Google but "Canidae" in Bing. In this sense, it is necessary to fuse knowledge cards from various search engines automatically to provide a more comprehensive summary with all important facts for a given entity. Also, search engines usually update their information quickly so that we can always get up-to-date information from the fused cards.

On the other hand, aggregated search tries to combine and rank results from different sources in a unified way. While aggregated search has been studied for years, the original can only deal with aggregation among search snippets and the recent developments [3, 4] begin to consider the heterogeneous entity retrieval and on-the-fly entity consolidation problem. Compared with search snippets or entities, knowledge cards are more structured with richer information. These structured information is good for navigation and querying. But it brings additional challenges when fusing knowledge cards. To the best of our knowledge, there is no existing research working on knowledge card fusion in either entity search or aggregated search. Compared with the state of the art knowledge fusion work (e.g., KnowledgeVault [1]), cards are fused in an online manner when a query is submitted.

In this demonstration, we present a system to fuse knowledge cards from various search engines. Based on the card fusion engine[3], we further develop a Javascript library called *KCF.js*, which is under the GNU General Public License[4]. It can be downloaded via the following link[5]. Besides the Javascript library, a live demo with sample queries as well as an example video are also provided. Figure 1 shows the overview of KCF.js and its possible applications. As a basic function, our fusion engine merges cards representing the same entity from different engines into a fused card. Developers can easily equip in-site search with our fused knowledge cards. When selecting a fused card (e.g., the card about "Fox Broadcasting Company"), the detailed information of the card is displayed. In particular, when a mouse hovers on a value, different expressions of the same value from the original search engines are shown. For instance, the value "Los Angeles, California, United States" is expressed as "Los Angeles, California" in Bing. Besides search, KCF.js

*This work was partially supported by the National Science Foundation of China (project No: 61402173).

[1] http://www.wikipedia.org/

[2] https://www.freebase.com/

[3] The fusion engine extends the implementation presented in our previous work [5] with sophisticated ranking functions.

[4] http://www.gnu.org/licenses/gpl.html

[5] http://www.nlp-bigdatalab.com/kcf/

WWW'16 Companion, April 11–15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4144-8/16/04.
http://dx.doi.org/10.1145/2872518.2890556 .

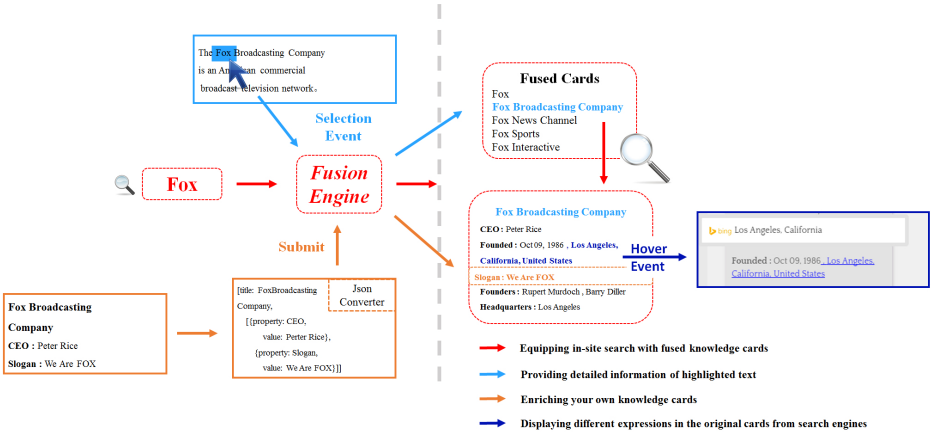

Figure 1: The overview of KCF.js and its possible applications

can also be applied in some other interesting applications for developers. When a developer implements the text selection event in some existing text editor using the library. Whenever a user highlights some words in such a text editor, our library triggers the card fusion engine. Detailed Information of highlighted text will be provided in a real-time manner. Moreover, if some user creates a customized card with the title of an entity and several manually created attribute-value pairs (AVPs), and publishes it to some Web page, we can use the library to find the corresponding fused card as the enrichment to the user-defined card.

2. WORKFLOW

As shown in Figure 2, there are four main components, namely *Card Disambiguater*, *Property Aligner*, *Value Deduplicator*, and *Knowledge Card Ranker* of the fusion engine. When submitting a query, knowledge cards along with other related data are first fetched from the three search engines through the *Knowledge Card Extractor*. Specially, the highlighted text or the title provided in the customized content is also regarded as an input query. Then the *Card Disambiguater* makes use of object values (i.e., referring to other knowledge cards) in a card and the Wikipedia link (i.e., the hyperlink to Wikipedia provided by most cards) to identify its corresponding entities. In this way, we can merge cards together if they represent the same entity. Before aligning properties of these fused cards, the *Property Aligner* performs a pre-processing step for data normalization and link completion. In the following step, we design a learning-based method to predict whether two properties can be aligned. In particular, mappings from Wikipedia infobox properties to ontology properties in DBpedia are used as training data to learn the prediction model. In order to further increase the accuracy, post-processing steps including *Property Mutual Exclusion Filter* and *Object Value Range Validator* are carried out. The *Value Deduplicator* groups equivalent values of aligned properties into value clusters. Finally, the *Knowledge Card Ranker* considers different aspects to rank value clusters within an aligned property. The value scores are combined with the property score to help rank the attribute value pair group accordingly. The scores of all pairs within a card are further aggregated as the card score for card ranking if multiple merged cards are returned.

3. APPROACH

3.1 Card Disambiguation

A knowledge card from a search engine is usually provided a Wikipedia link referring to a unique Wikipedia page. In this case, the card is disambiguated. Otherwise, we need to identify the corresponding entity the card refers to.

Actually, card disambiguation can be treated as an entity linking problem. Due to the wide coverage of Wikipedia, it is selected as our target KB. The card label is set as a mention m which is used to be computed for the commonness score. Since the card label might be ambiguous, m can refer to several entities (e.g., "Fox" could be "Fox Animal" or "Fox Broadcasting Company"). In order to determine the most likely entity the card should correspond to, we additionally consider the influences of object values. If an entity is tightly connected with the corresponding entities of these object values, it has a high possibility to be the target of the card. As a result, those object values are utilized to compute the relatedness score (e.g., the entity"Fox Broadcasting Company" achieves the highest relatedness score with the entity "Peter Rice"). Thus, we combine these two scores as a final score of a possible entity. In our implementation, we simply choose the linear sum as the aggregation function with equal weights for both scores. If the final score of the top entity is greater than a given threshold, it is returned as the disambiguated entity of the card.

3.2 Aligning Properties Between Cards

3.2.1 Pre-processing

The focus of this step is to normalize values of different types. More specifically, for a string value, if it contains any delimiter (e.g., space, underline, colon, minus, and dot), the value is segmented into different parts by the delimiter. The string is also normalized by lowercasing all its characters. For a numeric value, it belongs to a particular type and is often associated with some unit. For instance, a height "30-50cm" from Google is expressed as "1.1-1.6ft" in Bing. By considering several units such as date time, currency, length and weight, we can already deal with a large proportion of numeric values in knowledge cards. For an object value, we try to add a missing link if the corresponding card has no

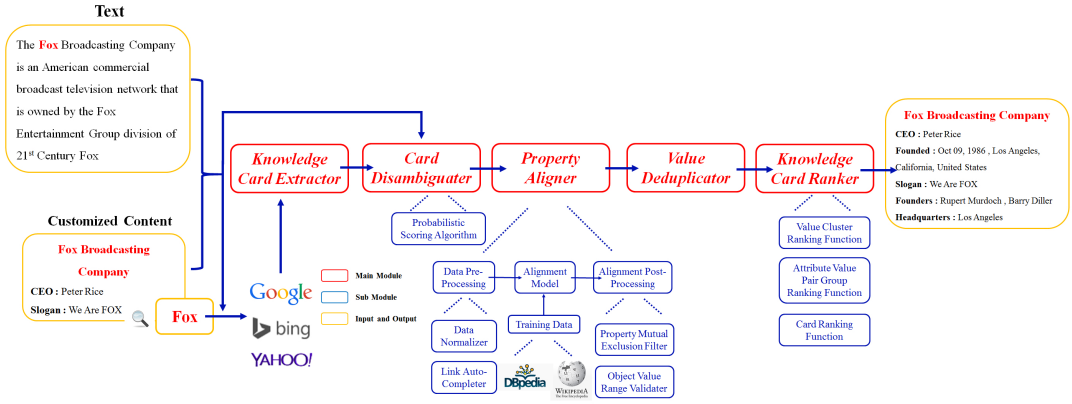

Figure 2: Overall workflow of fusing knowledge cards

Wikipedia link. The whole process is the same as that of card disambiguation introduced in Section 3.1.

3.2.2 *Learning-based Property Alignment*

In this section, we introduce the details of our learning-based method to check whether a property pair can be aligned. In particular, we design four novel features to constitute the learning model. Besides one property-related feature, we also consider several value-related features because values of a property can be regarded as its context to describe the property so that they will help predict property alignments.

- *Property Similarity.* It measures the similarity between two properties. The more similar they are, the more likely to be aligned. We consider two kinds of similarities: the lexical similarity based on edit distance and the WordNet-based semantic similarity.

- *Value Overlap Ratio.* If two properties do not have any value of the same type (i.e., string, numeric, and object), they are unlikely to be aligned. We use the Jaccard similarity to calculate it.

- *Value Match Ratio.* This feature further considers the match ratio of value pairs of a property pair. A value pair is matched if the two values are of the same type and their similarity is above a matching threshold. We then design specific similarity functions for different value types. For object values, we choose the ESA (Explicit Semantic Analysis) measure [2] to compute their similarities. The higher the match ratio, the more chance the property pair has to be aligned.

- *Value Similarity Variance.* The last value-related feature measures the similarity distribution of match pairs. The smaller the variance, the more match pairs have high similarities, which indicates that the property pair is more likely to be aligned.

3.2.3 *Post-processing*

The learning model might make incorrect predictions to some property pairs, which causes some false positive examples. In order to increase the precision of property alignment, we design two heuristic rules to filter out as many false positives as possible.

- **Property mutual exclusion filtering.** Observing that a knowledge card returned by a search engine only contains a limited number of highly selected properties, it is unlikely for the card to display redundant properties. Therefore, properties in an original card can be assumed to be distinct safely. That is, a property is disjoint with any other property in the same card, and cannot be aligned with any of these properties. For instance, "Spouse" and "Children" are two similar properties which are usually aligned by mistake. Generally, they always co-occur in a card describing a person. That is to say, the two properties fulfill the condition of the mutual exclusion so that they should not to be aligned.

- **Object value range validation.** If two properties can be aligned, their ranges should be compatible. That is to say, the categories of their corresponding values cannot be disjoint. For each object value, we use the categories of the corresponding Wikipedia entity. If the two category sets have no overlap, we think the categories of the value pair are disjoint. Take "Nelson Mandela" as an example card. "Mandela:Long Walk to Freedom" is a film storied by him while "Zindziswa Mandela" is one of his spouse. These two values share a same string of "Mandela" and both of them are related to "Nelson Mandela" which leads to an false positive example. However, they can be easily filtered out through the range validation.

3.3 Value Deduplication for Aligned Properites

After properties are aligned in a fused knowledge card, values of these aligned properties should be deduplicated so that equivalent ones in different expressions are grouped together into value clusters. We introduce a simple but effective method for this purpose. As mentioned in Section 3.2.1, values are normalized and the link of object values are completed. Therefore, it eases the deduplication of these values. If two object values link to the same Wikipedia entity, they are merged together. For the normalized numeric values or string values, we further use the corresponding matching threshold to check whether two values can be deduplicated. As an example, the value "Los Angeles, California, United States" and "Los Angeles, California" are duplicated values so that they are in a same value cluster.

3.4 Ranking

In this section, we introduce different ranking functions for value clusters as well as values in each cluster, attribute value pairs, and fused cards. Instead of designing three independent ranking functions, we consider the card richness when ranking values, combine value cluster scores with property scores for ranking attribute value pairs, and aggregate scores of all pairs into card ranking. More details of these ranking functions are described in the following subsections.

3.4.1 Value Ranking

The original rank position reflects the ranking preference of each search engine for the value. If the value occurs at multiple cards and all these cards rank the value at top positions, the value should be ranked high in the corresponding fused card. Moreover, the original rank positions are not treated equally. We prefer to trust a knowledge card if most of its properties also appear in the fused card.

Besides, we also consider two aspects for value ranking. Search engines usually list related entities under "people also search for". If the value happens to be in such a list, it is a signal indicating the importance of the value as well as its relevance to the query or the card. On the other hand, sometimes an object value may link to another knowledge card in which there exists another object value linking back to the current card. For example, the knowledge card of "Nelson Mandela" has an object value named "Mandela: Long Walk to Freedom". Meanwhile, the card describing "Mandela: Long Walk to Freedom" also mentions "Nelson Mandela". As the author, he is certainly important to this film. So the object value should be ranked high due to its high representativeness to "Nelson Mandela".

Each value is given an score which considers all the above factors. The value with the highest score is selected as the representative value of the corresponding value cluster. If a value cluster contains many equivalent values of high scores, it will be ranked high accordingly.

3.4.2 Attribute Value Pair Group Ranking

In this stage, we first compute the score of each attribute value pair (AVP). The score considers both property factors and factors of their corresponding value clusters. Before ranking attribute value pairs, we are aware that pairs are organized in adjacent positions of a card when they are approximate semantically. Thus, we utilize the original ranking positions as well as domains and ranges of AVPs to check whether they should be placed in one group. We then aggregate the scores of AVPs in a group (i.e., arithmetic mean) as the score of the group. For instance, if most search engines put the similar properties "born" and "die" at the first and the second places when describing a person, these two properties will be placed in a group. And then the group is ranked top while the positions of the two AVPs are determined by their own scores.

3.4.3 Card Ranking

An ambiguous query may lead to a set of knowledge cards to be returned. In this case, there might exist several fused cards. Similar to attribute value pair group ranking, two aspects are taken into account for card ranking. First, if a rich card has a large overlap with the fused card, the position of the card in a search engine has a strong influence on the ranking position of the fused one. That is, if the card is ranked high, the fused one should also be ranked in a high position. Second, fused cards with values of high-quality should also have more chances to become the top ones.

4. DEMONSTRATION

This demonstration will show how KCF.js is used in three different application scenarios.

- *Equipping in-site search with fused knowledge cards*

  More and more sites provide a search bar to search in-site contents. KCF.js can provide fused knowledge cards as comprehensive summaries of entities related to the input query. The query submitted to the search bar is also the input of our library to invoke the card fusion engine. As a result, the fused cards are returned in the JSON format and rendered in a panel of the search result page. Moreover, when your mouse hovers on a particular value in a fused card, a tip will popup to show its corresponding expressions in the original cards from different search engines.

- *Providing detailed information of highlighted texts*

  When writing an article, a reader may want to learn more about some entity he is not familiar with. Instead of searching several engines and merging results manually, embedding our Javascript library into the text editor offers a fully automatic way to do so. Whenever he highlights some words from the input text, the selection event is triggered. The highlighted text and the surrounding contents as context for card disambiguation are fed to the fusion engine, and the disambiguated card with detailed information is returned.

- *Enriching your own knowledge cards*

  Users might want to add structural information of an entity and publish these data contents as a customized knowledge card to some Web page. They do not necessarily have to add every property by themselves. Instead, they can just make a partial card and use the fused card returned by our library to provide extra contents as supplements. This could be regarded as a special case of adding a new source provider. Similarly, KCF.js can also be used to enrich infoboxes of Wikipedia pages.

5. REFERENCES

[1] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *KDD*, pages 601–610, 2014.

[2] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, pages 1606–1611, 2007.

[3] D. M. Herzig, P. Mika, R. Blanco, and T. Tran. Federated entity search using on-the-fly consolidation. In *ISWC*, pages 167–183. 2013.

[4] D. M. Herzig and T. Tran. Heterogeneous web data search using relevance-based on the fly data integration. In *WWW*, pages 141–150, 2012.

[5] H. Wang, Z. Fang, L. Zhang, J. Z. Pan, and T. Ruan. Effective online knowledge graph fusion. In *ISWC 2015*, pages 286–302.