Implementing Search Engine Knowledge Cards

Advisor: Dr. Carlos Rivero
By: Kumar Abhishek
1. Abstract

Web search engines have been the most common source of online knowledge. They fetch the most relevant and closest information for the query provided by the user by using online knowledge graphs. Every query entered by a user is transformed in a structured form, which is known as the knowledge card. These cards describe attributes of the entity queried to the search engine and also the relationship with different related entities. Different search engines like Google and Bing transform the query into different structured forms known as knowledge cards and these cards from different engines can be fused up, if compatible to get a much better comprehensive view about the queried entity. A query can be considered as an ambiguous query and can result in creating more than one knowledge card for the entity.

2. Introduction

Entity search on various search engines like Microsoft’s Bing and Google have been a very common way of retrieving information online and a large segment of queries entered in the search engines are for getting information about entity. Request are made to search engines like google and they return a set of responses related to the queried entity. The process of putting a query and getting the information about the entity consist of a lot of steps. Once a query is submitted to search engines like google and Bing, the search engine uses the publicly available knowledge base like Freebase and Wikipedia and extracts the information and creates a structured summary from the information which is known as the knowledge card. This summary is available publicly in the JSON format in the form of API’s of the search engines. This knowledge card having information about the entity is then sent to the browser and displayed as response for the entity.
Query entered as a request could be ambiguous as a query can mean different entities. Example, query “Apple” can mean fruit apple or the company Apple (www.Apple.com).

![Knowledge card and properties displayed on browser](image)

In this paper, we will be discussing overall workflow of Integrating search engine knowledge cards. Specifically, we will be discussing the integrated approach by overcoming the following challenges.

1. Research and implementation of “Retrieval of Knowledge cards automatically from Google and Bing search engines”
2. Implementing algorithms and formulas for card Disambiguation. Card Disambiguation is the process of finding an entity in the knowledge base like Wikipedia, based on the card label (text). It makes sure both search engine returns the knowledge card for the same entity.
3. Implementing Algorithms for property Alignment where value has to be normalized for the units which are of same type and this technique will be implemented in the preprocessing step. Example, for query “RIT” the value for the property ‘acceptance rate’ for google knowledge card is 57% whereas the value for the same property in Bing knowledge card is 57.40%. So the property is same in both the knowledge cards but to fuse them up it should be normalized as the values are in different format.
4. Algorithms are implemented for Value deduplication approach where different expressions still have the same value and are categorized into clusters.
3. Background

In this section a brief background on the research done on topic ‘Effective Online Knowledge Graph Fusion’ is highlighted. This paper[1] talks about fusing data on the go based on the query entered by the user. Authors of this paper discussed the overall workflow of the approaches used to fuse knowledge cards online. They explained the efforts required to fuse knowledge cards automatically from different search engines using knowledge card extractor. Once cards extracted, card disambiguation approach was used and algorithms like probability score for ambiguity removal and other related formulas were implemented. Card Disambiguation is the process of finding an entity in the knowledge base like Wikipedia, based on the card label (text). The methods used for card disambiguation are commonness and relatedness. Commonness gives the number of links for a Wikipedia entity and relatedness is a measure of closeness of an entity to the object value.

The next approach explained and implemented after Card disambiguation is property alignment where values are normalized for different card results and learning based property alignment method algorithm and formulas were implemented. Once above discussed approaches are implemented different expressions having the same value that is categorized to value clusters and this approach is known as value deduplication.

4. Project Goal

To retrieve knowledge cards of Google and Bing search engines, integrate them to achieve a better comprehensive view about the entity.

5. System Specification

Operating System: OSX, Windows
Coding language: JavaScript, Java
Frameworks: AngularJS, ExpressJS
Environment: Database

6. Implementation Overview

6.1 Retrieving Knowledge cards automatically from Google Search Engine

Google Knowledge Graph Search API helps us find entities from the Google Knowledge Graph and is a read only API. This API gives a ranked list of the most notable entities which matches the criteria.

To access this API, we created a google account and a project in the Google developers Console. Before sending request to knowledge Graph Search API of google, we need to activate the access
to this API and need to tell google about the client. We achieved this by using the console of google developer to create a project. In this console we gave the API access information and then registered our application. To get the API key we need to open the credentials page on the Developer’s console and then click the Add credentials link, fill in the details and get the API key to be used to retrieve this knowledge card.

Then the API’s can be invoked by sending HTTP request and parsing the received response using REST. REST stands for Representational State Transfer and is an architecture style to provide a convenient approach to request and modify data. In google API context, HTTP is referred to retrieve and modify data stored by google. Resources in this system are stored as a data source and the client has to send a request to server to perform actions like retrieving or updating and the server after performing the action, sends the response back to the client. In google RESTful API the client performs the GET action using HTTP to retrieve knowledge graph data and the URL created is of the form ‘https://kgsearch.googleapis.com/v1/entities:search?query=your_query &parameters’.

6.2 Retrieving Knowledge cards automatically from Bing Search Engine

Microsoft’s Bing Search API provides the same search experience as search engine Bing.com. This API sends query to Bing and retrieve a list of relevant search results including webpage, images, videos and more. To access Bing’s API, we need a subscription key for which we need to sign up to Microsoft account, click request new trials and see the list of services available. This list provides detail of the number of transactions which is allowed per second and per month. Clicking on subscribe button after selecting the services will create a primary and a secondary key. To call the BING search API we need to add this key in the ocp-Apim-Subscription-Key header. Client has to perform the GET action using HTTP to retrieve knowledge graph data and the URL created is of the form”https://api.cognitive.microsoft.com/bing/v5.0/search?q=rit&count=1&offset=0&mkt=en-us&safesearch=Moderate”.

6.3 Implementing Card Disambiguation Formula and Algorithm

Card Disambiguation is known as an entity linking process where an entity is searched in the knowledge base like Wikipedia and Freebase based on the card label (text). Among the available free knowledge base, Wikipedia is the most known knowledge base due to the huge availability of entities and the countless entities being added every day. So we selected Wikipedia as our Knowledge base to perform Card disambiguation process. To get Wikipedia knowledge cards we need to access Wikipedia search API’s which provides the same search experience as the Wikipedia’s search results. This API sends query to Wikipedia and retrieves a list of relevant search results with information like total number of hits found in the
Wikipedia, title of the entities, snippet of the text based on which this search result is selected, size word count and the timestamp.

Wikipedia’s search results can be accessed by using MediaWiki API which is a web service which provides access to data, features and meta-data of wiki using HTTP. As a client we perform GET request actions over HTTP and the URL created to get the search result data of 'http://en.wikipedia.org/w/api.php?action=query&format=json&list=search&origin=*&srsearch=apple&srlimit=20'. The action parameter in the header is used to get information for the query as ‘action= query’ and the parameter format is used to get the output in the format specified. Major output formats to get the response is JSON, PHP, and XML. Srsearch header specifies the name of entity which has to be used as the text to retrieve information about and origin parameter is used to let the Wikipedia API know the origin of the application which is trying to access the API and * means to allow to get the access from any location and srlimit restricts the number of responses number of knowledge cards to be shown as output.

6.4 Implementing Relatedness Formula/ Algorithm

Knowledge card contains many attribute value pairs (AVPc) where P stands for properties and Vpc stands for corresponding value set. Among all these property value pairs for an entity, if value of a property links to a knowledge card which represents some other entity, then such values are called object values and if the value represents some numerical value like length or currency, then such values are known as numeric values. Relatedness is a measure of closeness of an entity to the value of the object.

To get object value pairs for an entity we need to access DBpedia Api’s which provides the list of all the corresponding attribute property for an entity and the related values to them. This Api gets all the properties and values of an entity using Wikipedia data.

All property value pair of Wikipedia can be accessed by using DBpedia API which is a web service which provides access to data, features and meta-data of property value pairs from wiki using HTTP. As a client we perform GET request actions over HTTP and the URL created to get the property value result data is ‘http://dbpedia.org/data/entitiyname.json’ where entity name is the name of the entity for which we want to get the corresponding properties and their values. In the header tag ‘json’ is provided to get the data in Json format.

7. Results and Analysis

7.1 Knowledge Card of Bing and Google

A URL for HTTP GET request for the query ‘RIT’ is of the form ‘https://kgsearch.googleapis.com/v1/entities:search?query=rit&key=KEY&limit=1&indent=True’ and the Knowledge card created by google for the entity RIT is shown in Figure[5].
The number of knowledge cards returned by google and Bing for the same entity is very different as the creation of structured summary for any entity is based on the algorithms used by Google and Bing.

URL created by Bing for the same query ‘RIT’ is of the form https://api.cognitive.microsoft.com/bing/v5.0/search?q=rit&count=1&offset=0&mkt=en-us&safesearch=Moderate.'
For query “RIT” Google knowledge card shows information about the entity like ‘name’, ‘type’, ‘description’, ‘detaileddescription’, ‘image’ and ‘url’ whereas the knowledge for Bing displays a lot more structured information about the entity like ‘name’ and its synonyms, ‘displayUrl’, ‘deeplinks’, ‘images’, ‘relatedsearches’ and ‘rankingResponse’.

From Example, we can see that knowledge card of Bing and Yahoo represent the same entity “RIT” but some of the properties are different in both the cards and some are just available in one card. So Fusing up these two knowledge cards from different search engines will result in a better comprehensive summary with more facts and information for an entity. As data keep on updating in knowledge base very fast, the cards also gets updated and the fused up information is also up-to-date.

7.2 Card Disambiguation Formulas and Algorithms.

The methods used for Card disambiguation are commonness and relatedness. Commonness gives the no of links for a Wikipedia entity and relatedness is a measurement to calculate the closeness of an entity with its object value. Commonness score for an entity can be calculated by finding the number of links found in Wikipedia with the label ‘m’ and target ‘e’ and is referred by \( L_{m,e} \) and the number of links found without the text in the label is referred by \( L_{m,e}' \). So to get this score we take the count of entities found with the text in the label and divide with the summation of links without the text in the label of the knowledge card and divide them and get the commonness score \( \text{score}(m,e) = \frac{|L_{m,e}|}{|L_{m,e}'|} \) where ‘m’ is the card label and ‘e’ is the entity of the Wikipedia and ‘Em’ is the several entities present in the Wikipedia.

For example, we take the text ”Apple” and tried to get the knowledge card with ‘text= apple’ in the label of the entity. We took a small subset of 20 knowledge card for getting the commonness score and calculated the \( L_{m,e} \) value by extracting all the ‘title’ header for all the knowledge cards of Wikipedia in the form of Json response and put them in an array. Now we checked if apple is available as a text in the label for all the entities and extracted entities with the text and put that in another array.

Next step was to calculated the length of the array which has the entities information with the ‘text =apple’ in the label and then divided this count with the total no of entities produced and the resulted score was the commonness score. For our experiment with apple we got a commonness score of 0.8 as ‘\( L_{m,e} = 20 \)’ and value of \( L_{m,e}' =16 \) and then using the above formula, commonness score was calculated.

A URL for HTTP GET request for the query ‘Apple’ is of the form ‘http://en.wikipedia.org/w/api.php?action=query&format=json&list=search&origin=*&srsearch=Apple&srslimit=20’ and the Knowledge card created by MediaWiki for the entity Apple is shown in Figure[7].
Card Disambiguation property makes sure that the knowledge card returned by Google and Bing represent the same entity so that Property Alignment and Value Deduplication process can be implemented once we know both knowledge card refers the same entity. So to achieve this from Google and Bing knowledge card, we extracted the name, description and Url from the google knowledge card and then extracted all the different names and the displayUrl of the entity given by Bing Knowledge card and put them in an array. After implementing this, we compared if name of the entity produced by google is one of the names produced by Bing. Once it matches and returns true then we take the entity url from Google and check if the displayUrl of the Bing is a subset of the google Url. Once Url and name matches for both the Json responses, there are very high chances that they represent the same entity making sure that there is no ambiguity present for the resulted knowledge cards from Bing and Google and property Alignment and Value Deduplication process can be implemented next.

7.3 Relatedness Formulas and Algorithms results

Relatedness is a measure of the property closeness for any entity to its object value. The label for a card can be ambiguous resulting in several entities with the same label. So we need to find the closest entity the knowledge card should correspond to. To get this result, we need to consider object values as its context for the knowledge cards. For an entity there are object value and if these entities are tightly connected with them, then there is a very high possibility of that knowledge card being the target. For this purpose, relatedness score is used which is a measure to find the closeness of an entity to its object values. Relatedness score of an entity can be calculated by taking the number of links found in the DBpedia data with the label Le and Lev is the number
of all property value links available with the DBpedia. Maximum and minimum value among Le and Lev needs to be calculated and we also need to calculate the intersection value of Le and Lev. Once we get all these values calculated we can find the relatedness score $[1] = 1 - \frac{\log(1 + \max(|Le|, |Lev|)) - \log(1 + |Le|) / \log(WP)}{\log(1 + \min(|Le|, |Lev|))}$ where WP refers the total number of Wikipedia entities.

Using the above formula for entity “Apple Inc.” maximum value among Le and Lev is 26 and minimum is 25. So calculating the relatedness formula gives the value 0.9400227470393991.

A URL for HTTP GET request for getting property value for entity ‘Apple Inc’ is of the form ‘http://dbpedia.org/dataApple_Inc..json’ and the Knowledge card created by DBpedia for the entity Apple Inc is shown in Figure[8].

![Figure8: DBpedia property value knowledge card](http://dbpedia.org/property/keyPeople": [
  {
    "type": "literal",
    "value": "*",
    "lang": "en"
  },
  {
    "type": "uri",
    "value": "http://dbpedia.org/resource/Luca_Maestri"
  },
  {
    "type": "uri",
    "value": "http://dbpedia.org/resource/Arthur_D._Levinson"
  },
  {
    "type": "uri",
    "value": "http://dbpedia.org/resource/Jonathan_Ive"
  },
  {
    "type": "uri",
    "value": "http://dbpedia.org/resource/Jeff_Williams_{Apple}"
  },
  {
    "type": "uri",
    "value": "http://dbpedia.org/resource/Tim_Cook"
  }
],

Figure8: DBpedia property value knowledge card

Above diagram shows the property key people for the entity Apple Inc and the value pairs related to this property. Figure[9] displays the property value pair in simple html format. Considering this example, relatedness is used here to check the properties like key People where Tim_Cook is an object value, so if the knowledge card of Tim_cook is measures and there is a strong connection of Apple Inc with Tim_Cook then relatedness formula will give a high score to Apple Inc.
7.4 Lexical Similarity formula implementation and results.

This is a post processing step used for aligning properties of different knowledge cards. Different knowledge cards of different search engines represent the value of the entity properties in different representational format which needs to be aligned before integrating the knowledge cards so that duplication of attribute value pairs could not be a challenge. For example, the value for the entity ‘Amazon’ and for property ‘CEO’ the value by Bing is “Jeff Bezos” and Google is “Jeff Bezos (May 1996-)”. So if we need to integrate the above property value we have to align the values of CEO to either the format of Google or Bing.

The first step of property alignment is to check if the string size is same for both the values of Google and Bing search engine. We implement the lexical similarity formula for this and check the results with the results in paper[1]. The formula of lexical similarity is $SIMls(p1, p2) = w \times \frac{|substr(lp1, lp2)|}{\min(|lp1|, |lp2|)}$ where $w$ stands for weight which is 1 if the string from both the search engines are same. This weight is considered to be 0.8 if the first string(s1) is suffix or prefix of second string(s2). Weight is 0.6 if string 1 is substring of s2 but not just the suffix or prefix. If all the above conditions do not match, then we take the value as 0.4 for weight. P1 and p2 are the property values and Lp1 and Lp2 are the length of the value for the properties.
7.5 Experiment results and comparison.

In this section we will be comparing some of the results obtained by using the commonness and relatedness formula for some specific entities. The entities that we took for experiments are ‘Amazon’, ‘Apple_Inc.’, ‘New York Times’, ‘Rit’ and ‘Uber’

<table>
<thead>
<tr>
<th>Entity</th>
<th>Commonness</th>
<th>Relatedness</th>
<th>Combined</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>0.8</td>
<td>0.71</td>
<td>0.74</td>
</tr>
<tr>
<td>Apple</td>
<td>0.35</td>
<td>0.85</td>
<td>0.68</td>
</tr>
<tr>
<td>New York Times</td>
<td>0.52</td>
<td>0.44</td>
<td>0.38</td>
</tr>
<tr>
<td>RIT</td>
<td>0.95</td>
<td>0.77</td>
<td>0.8</td>
</tr>
<tr>
<td>Uber</td>
<td>0.50</td>
<td>0.37</td>
<td>0.42</td>
</tr>
</tbody>
</table>

Above table represents the commonness, relatedness and combined results of different entities. Commonness score tells how many entities have the same label as the entity asked by a user. This score represents the percentage of labels with the same name out of total Wikipedia results. Relatedness score measures how object value pairs are related with the same and combined results are the average score of combined results. High scores represent how accurate the entity is found in the Wikipedia.

Below graph shows the table results in a graph format.
7. Conclusion

In this paper we discussed some of the approaches to get knowledge cards of search engine like Google and Microsoft Bing. Important property value pairs like entity URL and entity name were extracted from the above knowledge cards to compare and check, if they represent the same entity. We implemented the commonness and relatedness formula to check and compare our results with the results achieved in paper[1]. To implement card disambiguation formulas, we had to retrieve Wikipedia and DBpedia knowledge cards using HTTP Rest Api’s. Wikipedia knowledge card results were used to retrieve the entity labels to be used in the commonness formula. DBpedia knowledge card which shows the relation among the property value pairs of any object were used to implement the relatedness formula.

We also implemented lexical similarity formula which is a part of the property alignment techniques where it checks if the values of the properties returned for Google and Bing entity are same by comparing their length and all the string subsets. Our results achieved using the implemented formulas produce close results when compared with results of paper[1].

9. Future Work

In the future, property alignment formulas like semantic similarity, value overlap ratio(VOR), value match ratio(VMR) and value similarity variance(VSV) could be implemented. Once above formulas are implemented successfully and the result are compared with the paper[1] then post processing steps like property mutual exclusion filtering and object value range variation heuristic rules should be implemented so that precision score for property alignment can increase.

Value deduplication process can also be implemented after implementation of Card Disambiguation and Property Alignment formulas and algorithm. Value deduplication of entities are required so that the duplicated properties with the same object and value properties could be grouped together.

User interface of current application could also be enhanced. Current application can take any entity name as an input from the user and can calculate commonness, relatedness and lexical similarity score for any entity. It also shows the ‘GET’ response results of Google Bing, Wikipedia and DBpedia knowledge card in JSON format.
10. References

[1] Haofen Wang, Zhija Fang, Le Zhang, JeffZ. Pan and Tong Raun. Effective Online Knowledge Graph Fusion. East China University of Science and Technology, China and University of Aberdeen, UK