Effective Online Knowledge Graph Fusion*

Haofen Wang[1], Zhijia Fang[1], Le Zhang[1], Jeff Z. Pan[2], and Tong Ruan[1]

[1] East China University of Science and Technology, Shanghai, 200237, China
[2] University of Aberdeen, UK

Abstract. Recently, Web search engines have empowered their search with knowledge graphs to satisfy increasing demands of complex information needs about entities. Each engine offers an online knowledge graph service to display highly relevant information about the query entity in form of a structured summary called *knowledge card*. The cards from different engines might be complementary. Therefore, it is necessary to fuse knowledge cards from these engines to get a comprehensive view. Such a problem can be considered as a new branch of ontology alignment, which is actually an on-the-fly online data fusion based on the users' needs. In this paper, we present the first effort to work on knowledge cards fusion. We propose a novel probabilistic scoring algorithm for card disambiguation to select the most likely entity a card should refer to. We then design a learning-based method to align properties from cards representing the same entity. Finally, we perform value deduplication to group equivalent values of the aligned properties as value clusters. The experimental results show that our approach outperforms the state of the art ontology alignment algorithms in terms of precision and recall.

1 Introduction

With the prevalence of entity search [1], 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 and Freebase 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. Such a card can be regarded as a query-based online form of the knowledge graph. Since a query might be ambiguous, it could return several cards corresponding to different real-world entities. Google returns three cards for the query "Fox" while Bing returns two more different cards. Even though the two cards represent the same entity, some property may just appear in one card. For example, only Google gives an attribute named "Daily sleep" in the card describing "Fox (animal)". So it is necessary to fuse knowledge cards from various search engines automatically to provide a more comprehensive summary with

* This work was partially supported by the National Science Foundation of China (project No: 61402173), the Fundamental Research Funds for the Central Universities (Grant No: 22A201514045) and the EC MSC K-Drive project (286348).

all important facts for a given entity. Also, search engines usually update their contents quickly so that the fused cards always contain up-to-date information.

Knowledge cards fusion can be regarded as an ontology alignment task. Different from traditional ontology alignment, cards are fused online when a query is submitted. Actually, it is a new branch of ontology alignment considering that input ontologies (i.e., cards) might be lack of schema-level information like concepts and domains or ranges of properties. Further, each input ontology (or a card) only contains a limited number of attribute value pairs. Instances might be expressed as string values in a card. Equivalent numeric values might use different units. Therefore, sophisticated ontology alignment algorithms working for large ontologies with rich information cannot be directly applied.

In this paper, we present the first effort to work on fusing knowledge cards from various search engines automatically. More specifically, we introduce an integrated approach with the following contributions. (1) We propose a novel *probabilistic scoring* method for knowledge card disambiguation. Two widely used measures namely the *commonness* score and the *relatedness* score in entity linking are combined to find the most likely Wikipedia entity a card should refer to. Therefore, different cards representing the same entity can be merged as aligned instances. (2) We design a *learning-based* method with four novel features to predict property alignments. The features include the property similarity and different aspects of similarities between values of two properties. In this way, we not only consider the similarity between two properties but also leverage their *context-based similarities.* (3) We normalize values of a same unit type and complete links for values representing same entities in a pre-processing step. As a result, equivalent values using different expressions are normalized into a same value or linked to a same entity. Both *data and unit normalization* and *missing link completion* can further increase the coverage of property alignment. Moreover, it helps group equivalent values of aligned properties into value clusters during value deduplication. (4) We carried out comprehensive experiments to test the effectiveness of card disambiguation, property alignment, and value deduplication on knowledge cards collected from a number of real entity queries. Furthermore, we convert the cards into ontologies and feed them into several state of the art ontology alignment tools. Our approach outperforms these tools in terms of precision and recall for both instance alignment and property alignment. The rest is organized as follows. Section 2 gives a brief overview. Section 3 introduces the approach details. Section 4 shows the experiment results. Section 5 lists the related work and Section 6 concludes the paper.

2 Approach Overview

2.1 Problem Definition

Input: Given an entity query, a search engine (e.g., Google) may return zero to several knowledge cards. Search on a specific KB like Freebase or Wikidata can also be regarded as a special case of search engine. Each card c describes one real-world entity e with a label on top. The card can also contain several attribute

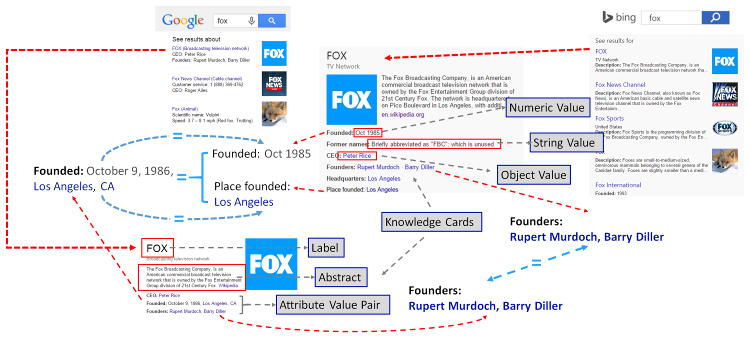

Founded: October 9, 1986, Los Angeles, CA
Founded: Oct 1985
Place founded: Los Angeles
Numeric Value
String Value
Object Value
Knowledge Cards
Founders: Rupert Murdoch, Barry Diller
FOX
Label
Abstract
Attribute Value Pair
Founders: Rupert Murdoch, Barry Diller

Fig. 1. Knowledge cards from Google and Bing when searching "fox"

value pairs AVP_c=$\{avp_1$,avp_2,. . .,$avp_m\}$ in which each pair is composed of a property p and a set of corresponding values $V_{p,c}$=$\{v_1$,v_2,. . .,$v_k\}$. Among them, if v links to a knowledge card representing another entity, we call v an *object* value. Otherwise, if a value v represents some numeric value of a data type like length, currency, or date, it is called a *numeric* value. The remaining values are *string* values. Besides, a short abstract might be provided to describe the card.
Output: One or more merged knowledge cards $\{c_{m1}$,c_{m2},. . .,$c_{mi}\}$ are returned. Each c_{mi} corresponds to a set of cards representing the same entity from different engines (e.g., c_{gi}). Here, c_{gi} can be a card returned by Google. In c_{mi}, each avp_{mi} becomes a cluster of the original avps. More precisely, equivalent properties $\{p_1$,p_2,. . .,$p_j\}$ are aligned together to constitute a merged property p_{mi} of avp_{mi}. Furthermore, the value sets of these properties are grouped into a merged value set V_m. Each member of V_m is a value cluster containing equivalent values of aligned properties from different cards.

Taking "fox" as an example query, Figure 1 shows a list of possible knowledge cards returned by each engine. It also shows the details of two cards from Google and Bing respectively. The figure illustrates the label, the abstract, attribute value pairs as well as properties and different types of values in these pairs of an individual card. Since these two cards represent the same entity, they can be merged together. Here, we show two aligned pair examples: one is a one-to-one mapping between two "Founders" properties and their values, while the other is a one-to-many mapping which will be explained later.

2.2 Challenges

In order to fuse knowledge cards from various search engines for an entity query effectively, we face several challenges which are listed as follows.

Ambiguous cards from a same query: Since an input query can be ambiguous, it may returned several knowledge cards. As shown in Figure 1, Google returns three different cards and Bing returns five cards. In addition, a card representing the same entity "Fox Broadcasting Company" may have a different label "FOX

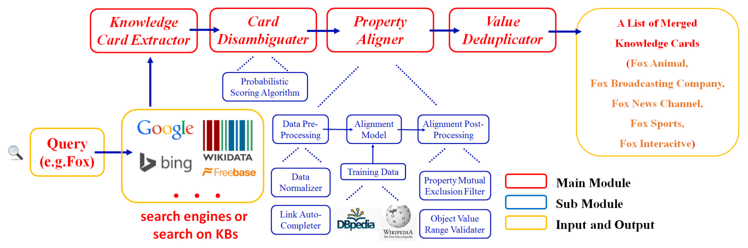

Knowledge Card Extractor
Card Disambiguater
Property Aligner
Value Deduplicator
A List of Merged Knowledge Cards (Fox Animal, Fox Broadcasting Company, Fox News Channel, Fox Sports, Fox Interacitve)
Probabilistic Scoring Algorithm
Query (e.g.Fox)
Google
bing
WIKIDATA
Freebase
search engines or search on KBs
Data Pre-Processing
Alignment Model
Alignment Post-Processing
Data Normalizer
Link Auto-Completer
Training Data
DBpedia
Wikipedia
Property Mutual Exclusion Filter
Object Value Range Validater
Main Module
Sub Module
Input and Output

Fig. 2. Overall workflow of our approach to fuse knowledge cards

(Broadcasting television network)". How to merge cards into different entities correctly is challenging. It can also be treated as an instance alignment task.

Same value but different expressions: Even two cards are merged correctly, they might have equivalent properties or values with different expressions. For example, a card "Inception (2010)" returned by Bing has a property named "Estimated budget" while the corresponding card from Yahoo! describes the same meaning by using "Budget". Furthermore, one value is expressed as "$160 million USD" while the value of the "Budget" property is "$160,000,000". The similar situation also occurs when expressing other kinds of values.

One to many mappings: A property of one card can be aligned with one or several properties of another card. As shown in Figure 1, the property "Founded" of the card "FOX (Broadcasting television network)" introduces the founded date and the founded place of the company. It should be aligned with two properties "Founded" and "Place founded" of the card "Fox Broadcasting Company" from Bing. In most cases, the labels of properties to be aligned are not the same, sometimes even totally different. Moreover, these properties may share very few values in common. The above two factors make property alignment difficult.

2.3 Workflow

As shown in Figure 2, there are three main components, namely *Card Disambiguater*, *Property Aligner*, and *Value Deduplicator*, to fuse knowledge cards. When submitting a query, knowledge cards along with other related data are first fetched from the search engines through the *Knowledge Card Extractor*. Then the *Card Disambiguater* identifies corresponding entities in Wikipedia for these cards based on a probabilistic scoring algorithm. In this way, we can merge cards if they represent the same entity. Before aligning properties of these merged 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-processings including *Property Mutual Exclusion Filter* and *Object Value Range Validator* are carried out. Finally, the *Value Deduplicator* groups equivalent values of aligned properties into value clusters.

3 Approach Details

3.1 Card Disambiguation

Card disambiguation can be treated as an *entity linking* problem, i.e. linking a mention found in text to entities defined in a target KB. Due to the wide coverage of Wikipedia, it is selected as our KB. There are about 4.8M entities in Wikipedia and it's continuously growing. A sizable entities can be dealt with in this step. We use the card label as a mention m for disambiguation. Then we adopt $commonness(m, e) = \frac{|L_{m,e}|}{\sum_{e'} |L_{m,e'}|}$ as commonness score [14] to measure the strength m links to a Wikipedia entity e. Here $|L_{m,e}|$ is the number of links in Wikipedia with the target e and the anchor text m.

Since the card label might be ambiguous, m can refer to several entities E_m. In order to determine the most likely entity the card should correspond to, we additionally consider the object values of the card as its context. 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. For this purpose, we adopt $relatedness(e, v) = 1 - \frac{\log(1+\max(|L_e|,|L_{e_v}|))-\log(1+|L_e \cap L_{e_v}|)}{\log(|WP|)-\log(1+\min(|L_e|,|L_{e_v}|))}$ as the relatedness score [15] to measure how close an entity $e \in E_m$ is to an object value v. Here $|L_e|$ ($|L_{e_v}|$) is the number of links with the target e (or the corresponding entity e_v of v) respectively, $L_e \cap L_{e_v}$ is the intersection of links with the target e and e_v, and $|WP|$ is the total number of Wikipedia entities.

We further adopt $relatedness(e, V_o) = 1 - \prod_{v \in V_o}(1 - relatedness(e, v))$ to measure the relatedness between e and a set of object values V_o in the card. $relatedness(e, V_o)$ is indeed the probabilistic sum of all relatedness scores between e and each object value in V_o. We use $\hat{e} = \arg\max_e(commonness(m, e) \times relatedness(e, V_o))$ as the final score of a possible entity. Finally, the entity with the highest score greater than a threshold is selected as the disambiguation result of the card. Note that if a possible entity of the card does not co-occur with any corresponding entities, the final score is degraded to its commonness score.

For any $v \in V_o$, in order to get the corresponding Wikipedia entity, we can leverage object values in the card v links to and use the same formula to disambiguate v first. After we get the most likely entity e_v it refers to, we can get the relatedness score for $relatedness(e, v)$. So it is a recursive process. In our implementation, we simply choose the maximal relatedness score between a possible entity of v and e for $relatedness(e, v)$ as an approximation.

3.2 Aligning Properties Between Cards

Pre-processing The focus of this step is to normalize values of different types. More specifically, for a string value, it is lowercased. If it contains any delimiter, the value is segmented into different parts by the delimiter and it will be normalized by lowercasing all its characters. A numeric value belongs to a particular type and is often associated with some unit. For instance, a currency "\$160 million USD" from Bing is expressed as "\$160,000,000" in Yahoo!. Since value expressions vary a lot from one unit to another, we need prepare specific normalization rules for each unit. According to the unit distribution statistics reported in Section 4.3, by considering only several units such as date time, currency, length and weight, we can deal with a large proportion of numeric values in knowledge cards. For an object value, we try to add a missing link to the correspding Wikipedia entity. The link completion process is same as that of card disambiguation introduced in Section 3.1.

Learning-based Property Alignment In this sub-section, we introduce the details of our learning-based method to check whether a property pair can be aligned. If one property is aligned with two or more properties from a second card, we will consider it as a one-to-many mapping. 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 help predict property alignments.

- *Property Similarity* (PS). It measures the similarity between two properties p_1 and p_2. We consider two kinds of similarities: the lexical similarity (sim_{ls}) and the semantic similarity (sim_{ss}). The former works well if the property labels (l_{p_1} and l_{p_2} for p_1 and p_2 respectively) are close in their lexical forms. The latter can be a complementary to discover semantically similar properties in different expressions. So we first use $sim_{ls}(p_1, p_2) = w \times \frac{|substr(l_{p_1}, l_{p_2})|}{\min(|l_{p_1}|, |l_{p_2}|)}$ to calculate the PS value for p_1 and p_2. If the value is below a threshold, $sim_{ss}(p_1, p_2)$ is further used as the value. $|s|$ is the length of a string s, $substr(s_1, s_2)$ returns the longest substring of s_1 and s_2, and w is a weight. We set $w = 1$ if two strings are equal. If s_1 is a prefix or a suffix of s_2, w is set to 0.8. While s_1 is a substring (except prefix or suffix) of s_2, w is 0.6. Otherwise, $w = 0.4$. The reason to set different weights is because we assign different priorities to exact match, prefix or suffix, substring, and overlap. For sim_{ss}, we adopt the WUP measure [20] $sim_{ss}(p_1, p_2) = \frac{2 \times depth(LCA(s_{p_1}, s_{p_2}))}{depth(s_{p_1}) + depth(s_{p_2})}$ to calculate the relatedness by considering the depths of the two synsets and the depth of their lowest common ancestor (LCA) in WordNet. s_{p_i} is the most likely synset of p_i in the WordNet taxonomy.
- *Value Overlap Ratio* (VOR). If two properties do not have any value of the same type (i.e., string, numeric, and object), they are unlikely to be aligned. Let T_p be the value type set of a property p. For instance, if p has one numeric value and two object values in a card, $T_p = \{n, o\}$. The larger overlap T_{p_1} and T_{p_2} have, the higher coherence two properties achieve. We use the Jaccard similarity $VOR(p_1, p_2) = \frac{|T_{p_1} \cap T_{p_2}|}{|T_{p_1} \cup T_{p_2}|}$ to calculate the overlap.
- *Value Match Ratio* (VMR). It further considers the match ratio of value pairs of a property pair. The higher the match ratio, the more chance the pair can be aligned. We use the equation $VMR(p_1, p_2) = \frac{|MP_{(p_1,p_2)}|}{|CP_{(p_1,p_2)}|}$ to measure this similarity. A value pair is a *match pair* if the two values are of the same type and their similarity is above a matching threshold. $MP_{(p_1,p_2)}$ is the set of

match pairs of the pair (p_1,p_2) and $CP_{(p_1,p_2)}$ is the set of candidate value pairs. For example, if p_1 has one numeric value and one object value while p_2 has only one numeric value, then $CP_{(p_1,p_2)} = 1$. If the two values are dissimilar, then no match pair is found and $VMR(p_1, p_2) = 0$.
In order to get match pairs, we define a similarity measure for each specific value type. For string values, we use the same similarity measure used for property similarity. For the numeric ones, we used $sim_n(n_1, n_2) = 1 - \frac{dist(abs(n_1),abs(n_2))}{NormFactor}$ to calculate the similarity between n_1 and n_2.
Here $abs(n)$ returns the absolute value of n, $dist$ is the absolute difference between n_1 and n_2, and a $NormFactor$ is an normalization factor which picks the larger absolute value in general. Numeric values of the *date* type are special as each value contains three parts namely *year*, *month*, and *day*. Sometimes some date value (e.g., 2010-7) is even incomplete. Given a *date* type value pair with incomplete parts, we only focus on the common parts both values have during comparison. For a pair 2010-7-1 and 2011-6, the *day* part is ignored. Going back to Equation of $sim_n(n_1, n_2)$, we use 360 (counting 30 days per month) for $NormFactor$ instead. Taking the above value pair as the example, their distance is 330 and the similarity is 1/12 accordingly.
For object values, we choose the ESA (Explicit Semantic Analysis) [6] measure to compute their similarities. ESA computes semantic relatedness of natural language texts of arbitrary lengths. It represents the meaning of texts using relevant Wikipedia entity pages in form of concept vectors. It has been proved effectively for textual entailment and query expansion. Here, the similarity $sim_o(o_1, o_2)$ between two objects o_1 and o_2 of a pair is calculated by using the equation $sim_o(o_1, o_2) = \frac{\sum_{wc} wc(o_1)\cdot wc(o_2)}{\sqrt{\sum_{wc} wc(o_1)^2 \cdot \sum_{wc} wc(o_2)^2}}$.
It is actually the cosine similarity between concept vectors $wc(o_1)$ and $wc(o_2)$.

– *Value Similarity Variance* (VSV). It measures the similarity distribution of match pairs. The smaller the VSV, the more match pairs have high similarities, which indicates that the property pair is more likely to be aligned. We adopt the equation $VSV(p_1, p_2) = \frac{\sum(1-sim)^2}{|MP_{(p_1,p_2)}|}$ where sim is the similarity score (ranging from 0 to 1) of a match pair.

Post-processing 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.* Since a knowledge card only contains a limited number of highly selected properties, it is unlikely to display redundant properties. Thus properties in a card can be assumed to be distinct safely. That is, a property is disjoint with any other property in the same card. Based on this premise, given an aligned property pair predicted by the learning model, if one property happens to be in the disjoint set of another property, the aligned pair should be filtered out.
– *Object value range validation.* If two properties can be aligned, their ranges should be compatible. In another word, the categories of their corresponding

values cannot be disjoint. According to this principle, for any object value pair of two aligned properties, if their categories are disjoint, the property alignment should be removed. More precisely, 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.

3.3 Value Deduplication for Aligned Properites

After properties are aligned, values of these aligned properties should be deduplicated so that equivalent ones in different expressions are grouped together into value clusters. Here, we introduce a simple but effective method. As mentioned in Section 3.2, numeric values of the same unit type are normalized. Also, string values are lowercased and segmented into parts as new string values by predefined delimiters. For object values, links to their corresponding Wikipedia entities are completed. The above processing steps ease the deduplication of these values. That is, if two object values link to the same Wikipedia entity, they are merged together. For the two normalized numeric values or string values, we compare their similarity with the corresponding matching threshold to check whether the two values can be deduplicated.

4 Experiment

4.1 Experiment Setup

We selected a set of queries and submitted them to three search engines to collect knowledge cards to be fused. A query is chosen if at least two engines return knowledge cards for it. Secondly, a portion of selected queries should be ambiguous so that some engine will return several possible cards. Third, the returned cards should have different numbers of attribute value pairs (AVPs). We tried different titles of Wikipedia entity pages as well as disambiguation pages, and finally selected 26,583 different entity queries in total. Among these queries, about one fifth are ambiguous. Furthermore, We find that more than half of the cards are medium rich (includes AVPs ranging from 3 to 5), 19% are poor (fewer than 2 AVPs), and 24 percent are rich (more than 6 AVPs). We further randomly chose 154 queries from the above query set. The subset of queries conform the same richness distribution and have the similar percent of ambiguous queries. As a result, 464 knowledge cards are collected and manually labeled as ground truths to evaluate the performance of card disambiguation and property alignment. We downloaded mapping-based properties under DBpedia 2014 downloads[3] to collect Wikipedia infobox properties and the corresponding ontology properties in DBpedia ontology. Due to the large community of DBpedia, the collected mappings can be assumed to be of high quality. While it is impossible for these mappings to be 100 percent correct, they can still be used to train a robust learning model for property alignment. These cards are also converted

[3] `http://wiki.dbpedia.org/Downloads2014`

to ontologies as inputs of several ontology alignment tools. We finally compare the alignment performance of these tools with that of ours. All the data can be downloaded via the following link[4] for the purpose of experiment reproductivity. For more experiment details, you can refer to our technical report[5].

4.2 Card Disambiguation Evaluation

We compare our disambiguation method with two baselines in terms of accuracy and coverage. One baseline only considers the commonness score and the other uses the relatedness score. Here, coverage means the fraction of cards that have been disambiguated w.r.t. all cards while accuracy is the fraction of cards that are disambiguated correctly. The threshold is 0.01 to filter entities of low scores.

Figure 3 shows the comparison results. Using commonness score only can deal with the largest number of cards but achieves the lowest accuracy. In most cases, the card label is an anchor text linking to some Wikipedia entities so it can always return some entities as the disambiguation result. But the label is usually ambiguous and can refer to several entities. So using the label alone cannot distinguish among these possible entities. On the other hand, when considering the relatedness score only, the coverage becomes slightly lower but the accuracy increases significantly. This indicates that using object values in the card as its context can actually help filter unlikely entities the card might refer to. However, if two possible entities have similar relatedness scores, this baseline cannot decide which one to choose. For these cases, the commonness score might help. Therefore, our algorithm combines the strengths of both baselines. As a result, it gets 100 percent accuracy with very high coverage.

4.3 Unit Distribution Statistics

The same numeric values can be expressed differently using different units of the same type and should be normalized in a same unit, which has a positive impact on property alignment and value deduplication. It is impossible to enumerate all of them so we aim to prepare normalization rules for as few unit types as possible while still covering a sizeable cases. Since Wikipedia has a wide coverage and is an important source to build knowledge graphs, we can assume real data has the similar unit distribution. Thus, we collected 104,101 numeric values with units from Wikipedia infoboxes to analyze unit distributions. Statistically these units fall into nine types (i.e., currency, time, length, velocity, voltage, electric current, frequency, mass and area) and occupy almost 90 percent.

4.4 Property Alignment Performance

In this section, we first discuss how to tune different parameters for the learning-based alignment on DBpedia mappings. Then we apply the learned model to aligning properties of real world knowledge cards.

[4] `http://kcf.hiekn.com/download/experiment.tar.gz`

[5] `http://kcf.hiekn.com/download/tr.pdf`

Alignment Performance on DBpedia Mappings The DBpedia mappings dataset contains mappings from Wikipedia infobox properties to ontology properties in DBpedia. According to these mappings, we can get a large number of Wikipedia infobox property pairs in which each pair maps to the same ontology property. These pairs are used as positive examples for training a learning-based alignment model. The negative examples are those property pairs which have large values for any of the above introduced features but have not been declared to map to a same ontology property. We then discuss parameter tuning to learn a best alginment model without overfitting.

First, we determine matching thresholds for string, numeric and object values respectively to verfiy whether a pair of values of the same type can match. Changing thresholds will impact the MVR value of each property pair to be aligned. Instead of training another model to predict the most suitable thresholds, we analyze the precision distribution by setting different thresholds and pick the one achieving the highest precision. More precisely, we randomly select 200 value pairs for each type (i.e., numeric, string, or object) from positive and negative examples equally. Then we calculate the similarity of each value pair accordingly. Given a threshold, if the corresponding property pair is aligned having its similarity greater than or equal the threshold, or the property pair cannot be aligned and the similarity is below the threshold, we mark the value pair “T” (indicating a true positive or true negative). Otherwise, we mark it “F”. The whole process is repeated ten times. Then we get the average precision under this threshold by calculating the proportion of the number of “T”s to the total number (i.e., 200). The thresholds range from 0 to 1 and the step is 0.1. When the string threshold is 0.2, we get the highest precision. Similarly, we set the thresholds for numeric and object values as 0.8 and 0.5 respectively.

Second, we try to find an empirical value of the minimal size of training data to learn an “approximately best” alignment model. Here, we use Logistic Regression, one of the most widely used learning algorithms, to learn a model and test the performance of property alignment under different sizes of training data. More specifically, we randomly selected 200, 600, 2,000, 6,000, and 10,000 labeled property pairs from positive and negative examples equally. 5-fold cross validation is performed during model training. When adding the training data size from 200 to 6,000, the alignment performance improves significantly. But when we further increase the size to 10,000, the performance is almost unchanging. So we set 6,000 as the empirical training data size for further model selection.

We then use different learning algorithms to train various models. The alignment performance under these models are compared. Here, we choose Logistic Regression (LR), SVM, Decision Tree (DT), and Random Forest (RF) to compare. All the parameters are set as the ones used in the previous step. The learned RF model achieves the best F1-Measure. Since we have a post-processing step to filter out incorrect alignment candidates, we should pay more attention to recall. In this case, the selected model also has the best coverage.

Finally, we study the contributions of different features. Here, we choose five groups of feature combinations, namely all our features (All), all features except property similarity (All-PS), all features except value overlap ratio (All-

VOR), all features except value match ratio (All-VMR), and all features except value similarity variance (All-VSV). The same training data is used to learn five different random forest models.

The model using all features performs best. The F1-Measure scores of other models decrease to a certain extent, which indicates all features have some positive impacts to boost the performance of property alignment. Moreover, when removing property similarity from the feature set, the learned model has the lowest performance. This means that the property similarity feature is a key factor to judge if a property pair can be aligned.

Alignment Performance on Real Data From 464 knowledge cards, we randomly selected 3,487 attribute value pairs and asked students to manually label whether two properties in each pair can be aligned. As a result, 480 pairs are positive and 3,007 are negative. We applied the best model trained on DBpedia mappings to predict alignments on the above pairs. Moreover, we have two extensions. One considers pre-processing and the other further adds a post-processing step. Precision, recall, and F1-Measure are used for effectiveness study.

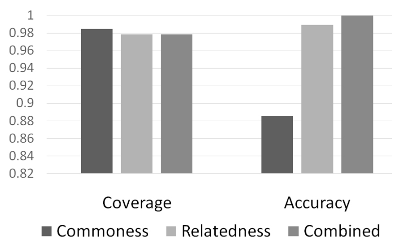


Fig. 3. Card disambiguation results

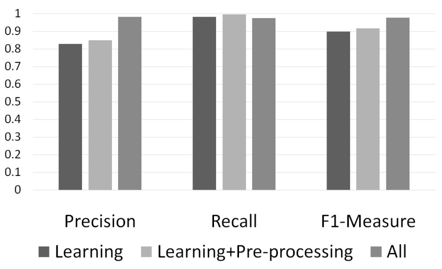
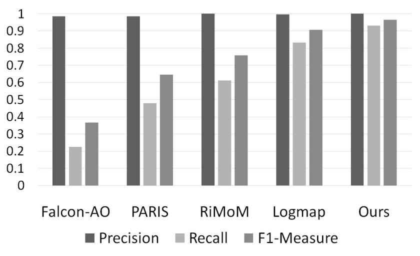


Fig. 4. Property alignment results

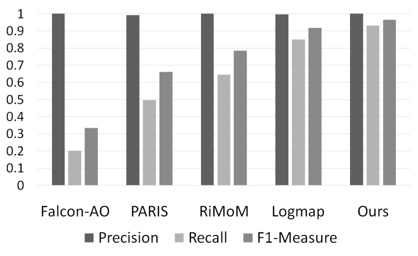
Figure 4 shows the alignment performance based on three methods. If an entity cannot refer to any Wikipedia entity, property alignment can still be executed. We get acceptable results without pre-processing. The model especially with pre-processing can actually predict aligned property pairs with a relatively high precision (more than 0.8) and almost perfect recall. Furthermore, after post-processing, the precision increases significantly at the expense of a slight drop of recall. This shows the effectiveness of post-processing used in post-processing. Regarding recall decrease, there exist two possible reasons. One is two similar object values are categorized into different types. The other is the disjoint sets are too tight so that the mutual exclusion filter kicks out some similar properties.

4.5 Effectiveness of Value Deduplication

Through deduplication, values associated with the same aligned property are clustered into different groups within an attribute value pair in a fused card.

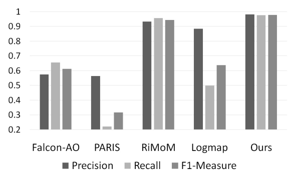


(a) Original ontology

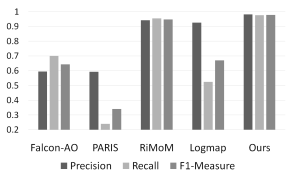


(b) + value and unit normalization

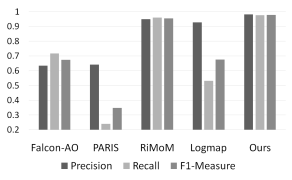
Fig. 5. Performance comparison of instance alignment



(a) Original ontology



(b) + link completion



(c) + data normalization

Fig. 6. Performance comparison of property alignment

Our experiments were carried out on the aligned attribute value pairs in the previous section (where 480 pairs are collected). As a result, we receive 1,431 value clusters which consist of at least one value. It is important to verify both precision and recall of this step. We manually labeled the correct clusters in which all the related candidate values are clustered correctly. Statistically, the recall of this step reaches 73.17% while precision is 98.9%. Due to the preprocessing, we have normalized both object values and numeric values so that values with same expressions can be easily deduplicated. However, there still left a certain amount of values which are not clustered. After analyzing bad cases, we find that some values may be originally out-of-date or inaccurate. For instance, a knowledge card of Arrian who is a Greek historian is presented by all the three search engines. Google and Bing show his death date as 175 AD while the same property is displayed as 160 AD in the Yahoo!'s card. Since the similarity between these two dates is below the threshold, they are clustered in two groups, which reduces the recall. On the other hand, some related values are not clustered correctly due to the failure in link completion. Considering two equivalent object values, if one has completed the link while the other does not, such a case will fail and reduce the recall.

4.6 Performance Comparison with Ontology Alignment Tools

Actually, card disambiguation and value deduplication can be treated as instance alignment tasks. So we compare our approach with the state of the art ontology matching tools for both instance alignment and property alignment. Here, we selected RiMoM [12], Logmap [11], Falcon-AO [10], and PARIS [19] as the tools for alignment performance comparison. The former two are among the top-3 tools of the OAEI champaign[6] in recent three years. The latter two also support both alignment tasks and have been widely used in practical applications.

We convert cards returned by one engine for a query into an ontology in the OWL format. Each card is treated as an instance with several attribute value pairs. Both object values and numeric values are treated as instances as well. For each numeric value, it is represented as an instance of a certain unit type (a concept defined in the unitontology[7]). The string values are treated as literals. If a property is associated with instances as its values, it is an *ObjectProperty*. Otherwise, it is a *DatatypeProperty*. Considering that there may be more than one kind of values in one property of a card, we divide this property into several ones in which each new property is associated with either instances or literals. A label is provided for each instance or property. Together with another ontology returned by a different engine for the same query, the above tools can be executed to get alignment results for instances and property pairs between two ontologies.

Here, we collected 204 cards from Google and 130 cards from Bing for all 154 queries used in previous experiments. Among 204 cards, there exist 848 properties with 1,377 values in all. Similarly, 610 properties with 913 values are found in 130 cards. We asked four students to manually label alignment ground truths. As a result, we totally get 596 instance alignments in which 108 are card pairs referring to the same entities and 488 are value pairs with two equivalent object values or numeric values. We also get 427 aligned property pairs.

Instance Alignment We record the precision, recall, and F1-measure of instance alignment run by each tool based on the above introduced ontologies as inputs. As shown in Figure 5(a), our approach not only achieves the best accuracy, but also the highest coverage. Among the selected tools, Logmap performs best especially in terms of recall. This is because Logmap can identify equivalent values of some particular unit type with the help of a Hermit reasoner. For instance, Logmap can align 105.3 mi^2 with 105.30 sq miles while the other tools fail. Given an alignment returned by our approach (one value is 0.056-0.11kg and the other value is 0.12-0.24lb), none of the tools are able to find out this alignment. This shows that Logmap might be able to deal with abbreviations or alias of some units, but unit conversion is still out of its ability scope.

Besides the original ontologies, we further add our pre-processing results to enrich these ontologies. As a result, the enriched ontologies contain normalized values and units, which are of higher quality with more unified vocabularies. When using the enriched ontologies as inputs of these tools, almost all tools have performance increases in terms of recall (shown in Figure 5(b)). This indicates that these tools do not complete missing links of object values, which can actually reduce the ambiguities, and thus have positive impacts on instance alignments.

[6] `http://oaei.ontologymatching.org/`

[7] `https://code.google.com/p/unit-ontology/`

Property Alignment Similarly, during property alignment, we not only use the original ontologies as inputs for these tools, but also feed them with two enriched version of ontologies. One enrichment is to add the card links to the original ontologies. Another is to further add our pre-processing results (i.e., value and unit normalization) similar to the enrichment made for instance alignment.

Figure 6(a), Figure 6(b), and Figure 6(c) show the property alignment performance comparison results of these tools and our approach based on original ontologies and two versions of enrichments respectively. From these figures, we can find that all tools are benefited from the two versions of enriched ontologies and gain precision and recall improvements. Our approach gets the best performance again. Among these tools, RiMoM is the best. The performance gap between RiMoM and ours mainly lies on dealing with one-to-many property alignments. For example, Both "Born" with two object values "July 18, 1918", "Mvezo, South Africa" and "Died" with two object values "December 5, 2013", "Houghton Estate, Johannesburg, South Africa" of a card should be aligned with the property "Lived" (who has an object value named "Jul 18, 1918 - Dec 05, 2013 (age 95)" from another card about Nelson Mandela. But RiMoM fails to find such an aligned property pair. Moreover, a small number of bad cases happen when one property has multiple values while the other with the same property name only has one value to be aligned. For example, a property named "Previous offices" with one value "Representative (NY 9th District) 1993-1999" cannot be aligned with the property of the same name with multiple values "Representative NY 9th District (1993 - 1999)", and "Representative NY 10th District (1983 - 1993)". Instead, we are able to solve the above bad cases of RiMoM thanks for our learning-based property alignment method.

5 Related Work

There are three lines of research related to our work. They are entity search, Web data fusion, and ontology alignment in the following subsections respectively.

5.1 Entity Search

Entity search has attracted more and more attentions from both academia and industry. Jeffrey Pound et al. [16] provided a solid framework for ad-hoc object retrieval. Jeffrey Dalton et al. [4] developed a method for coreference aware retrieval over a collection of objects containing a large degree of duplication. Roi Blanco et al. [2] proposed a content-based recommendation algorithm to provide a list of related entities for an input entity query. Roi Blanco et al. [3] presented an evaluation framework for repeatable and reliable semantic search evaluation. All these work focus on entity retrieval and ranking.

More recently, Daniel M. Herzig et al. [9] proposed different language models to tackle vocabulary and structure mismatches among different data sources for heterogenous entity retrieval. In [8], he further proposed a novel method for on-the-fly entity consolidation during federated entity search. The above two works consider instance alignment only during the query time. Thus, there is no existing research work on fusing knowledge cards from various search engines for an entity query.

5.2 Web Data Fusion

Essentially, knowledge cards fusion is a kind of data fusion considering both property alignment and instance alignment.

Xuan Liu et al. [13] proposed SOLARIS which starts with returning answers from the first probed source and refreshes the answers as it probes more sources. While it considers online data fusion, it expands to more sources iteratively. Different from it, we extract knowledge cards from multiple search engines (sources) at the same time before further fusion. D.Rinser et al. [17] leverages the inter-language links to identify equivalent entities. In our work, the link-completer plays the same role but uses a different method. R.Gupta et al. [7] focuses on creating a rich-attribute ontology by extracting attributes from query-stream and plain-text while our work invests heavily in ontology alignment.

More recently, Stefanidis et al. [18] described some efficient block-based entity resolution on the Web of data. All the above two work are about entity matching without considering property or class alignment, which are necessary in knowledge cards fusion.

5.3 Ontology Alignment

The closest work to ours is ontology alignment. PARIS [19], Falcon-AO [10], RiMOM [12], and Logmap [11] are ontology matching tools for the automatic alignment of entities, properties and classes from multiple ontologies. These tools get satisfactory results in the recent OAEI campaigns. They are selected to compare with our approach on alignment performance of knowledge cards fusion. Different from traditional ontology alignment settings, in our problem, schema-level information such as domains and ranges of properties are not provided. Sometimes, links to some object values are missing. Lack of such ontological knowledge, these tools fail to return important instance alignments or property alignments. Thus knowledge cards fusion can be seen as a new branch of ontology alignment task requiring new methods to deal with the above challenges.

6 Conclusion and Future Work

In this paper, we presented the first effort to work on fusing knowledge cards from various search engines. We proposed a probabilistic scoring method for card disambiguation. A learning-based method is then applied to align properties coming from different cards. Finally, we deduplicate the values of aligned properties and group these values into clusters. Knowledge cards fusion is actually a kind of online data fusion task involving both instance alignment and property alignment. Compared with several state of the art ontology alignment tools, our approach achieves better accuracy and wider coverage. As for future work, we plan to handle inconsistent [5] cards and to design ranking functions to rank values, attribute value pairs, and cards respectively so that we can return the most relevant fused cards with highly informative information for entity search.

References

1. Balog, K., Meij, E., de Rijke, M.: Entity search: building bridges between two worlds. In: Proceedings of the 3rd Semsearch Workshop. p. 9 (2010)
2. Blanco, R., Cambazoglu, B.B., Mika, P., Torzec, N.: Entity recommendations in web search. In: ISWC 2013, pp. 33–48. Springer (2013)
3. Blanco, R., Halpin, H., Herzig, D.M., Mika, P., Pound, J., Thompson, H.S., Tran, T.: Repeatable and reliable semantic search evaluation. Web Semantics: Science, Services and Agents on the World Wide Web 21, 14–29 (2013)
4. Dalton, J., Blanco, R., Mika, P.: Coreference aware web object retrieval. In: Proceedings of the 20th ACM CIKM. pp. 211–220 (2011)
5. Flouris, G., Huang, Z., Pan, J.Z., Plexousakis, D., Wache, H.: Inconsistencies, negations and changes in ontologies. In: Proceedings of AAAI. pp. 1295–1300 (2006)
6. Gabrilovich, E., Markovitch, S.: Computing semantic relatedness using wikipedia-based explicit semantic analysis. In: Proceedings of the 20th IJCAI. pp. 6–12 (2007)
7. Gupta, R., Halevy, A., Wang, X., Whang, S.E., Wu, F.: Biperpedia: An ontology for search applications. Proceedings of the VLDB Endowment 7(7), 505–516 (2014)
8. Herzig, D.M., Mika, P., Blanco, R., Tran, T.: Federated entity search using on-the-fly consolidation. In: ISWC 2013, pp. 167–183. Springer (2013)
9. Herzig, D.M., Tran, T.: Heterogeneous web data search using relevance-based on the fly data integration. In: Proceedings of the 21st WWW. pp. 141–150 (2012)
10. Hu, W., Qu, Y., Cheng, G.: Matching large ontologies: A divide-and-conquer approach. Data & Knowledge Engineering 67(1), 140–160 (2008)
11. Jiménez-Ruiz, E., Grau, B.C., Zhou, Y.: Logmap 2.0: towards logic-based, scalable and interactive ontology matching. In: Ontology Matching. pp. 45–46 (2011)
12. Li, Y., Li, J.Z., Zhang, D., Tang, J.: Result of ontology alignment with rimom at oaei'06. In: Ontology Matching. p. 181 (2006)
13. Liu, X., Dong, X.L., Ooi, B.C., Srivastava, D.: Online data fusion. Proceedings of the VLDB Endowment 4(11) (2011)
14. Medelyan, O., Witten, I.H., Milne, D.: Topic indexing with wikipedia. In: Proceedings of the AAAI WikiAI workshop. pp. 19–24 (2008)
15. Milne, D., Witten, I.H.: Learning to link with wikipedia. In: Proceedings of the 17th ACM CIKM. pp. 509–518 (2008)
16. Pound, J., Mika, P., Zaragoza, H.: Ad-hoc object retrieval in the web of data. In: Proceedings of the 19th WWW. pp. 771–780 (2010)
17. Rinser, D., Lange, D., Naumann, F.: Cross-lingual entity matching and infobox alignment in wikipedia. Information Systems 38(6), 887–907 (2013)
18. Stefanidis, K., Efthymiou, V., Herschel, M., Christophides, V., Others: Entity resolution in the web of data. In: WWW (Companion Volume). pp. 203–204 (2014)
19. Suchanek, F.M., Abiteboul, S., Senellart, P.: PARIS: Probabilistic Alignment of Relations, Instances, and Schema. PVLDB 5(3), 157–168 (2011)
20. Wu, Z., Palmer, M.: Verbs semantics and lexical selection. In: Proceedings of the 32nd ACL. pp. 133–138 (1994)