# Entity Attribute Ranking Using Learning to Rank

Esraa Ali Annalina Caputo Séamus Lawless

ADAPT Centre, School of Computer Science and Statistics,
Trinity College Dublin, Dublin, Ireland
{ esraa.ali, annalina.caputo, seamus.lawless }@adaptcentre.ie

## Abstract

Commercial search systems have recently begun using entity cards to support their ranked results lists. In this work, we propose an automated method for entity card building. We model this problem as a learning to rank approach applied to entity attributes. We introduce a new set of features that utilizes semantic information about entities as well as information from top-ranked documents using a generic search engine. The entity label is submitted as a search query to a general search engine and the top ranked documents are used to add context to the ranking process. The ranking model identifies and ranks the most important attributes (facts) of an entity. In order to experiment with our approach, we initially collected a dataset by exploiting Wikipedia infoboxes, whose ordering of attributes reflects the collaborative effort of a large community of Wikipedia users. Moreover, we are currently conducting a crowd-sourcing experiment to build a human labeled ground truth for our dataset. Our preliminary results demonstrate that our approach effectively replicates human ranking.

## 1 Introduction

As search behavior often involves more complex activities than straightforward fact-finding [WR09], modern search systems have evolved from relying solely upon ranked lists for results presentation, to adding other verticals for different data types (e.g. images, videos, research literature, and news). Studies have found that more than half of queries submitted to search engines are about entities [PMZ10]. Entity cards have been developed as a new way to retrieve and represent search results to satisfy such entity-oriented queries.

An Entity Card (EC) attempts to summarise the key facts about a certain entity. It can include images, textual data, maps or other data types. ECs are usually extracted from knowledge bases, but they can also include data from heterogeneous sources [BCMT13]. A recent study showed that entity cards have a strong effect on the way users interact with results [BZJ16]. They are useful as they assist users in disambiguating their queries and they directly provide basic information about an entity without the need to visit another page. ECs also include diversified information which supports user exploration and discovery needs.

Entity attribute ranking can be deployed to generate Entity Cards. It is used to collect and rank the most important information about an entity from knowledge bases. Attribute ranking is also useful in other applications. One of which is query expansion; for instance ranking the most important relevant attributes can be useful for

In: L. Dietz, C. Xiong, E. Meij (eds.): Proceedings of the First Workshop on Knowledge Graphs and Semantics for Text Retrieval and Analysis (KG4IR), Tokyo, Japan, 11-Aug-2017, published at http://ceur-ws.org

the users when they formulate their search query. It is also beneficial for faceted browsing to provide entity summaries. To the best of our knowledge, the current processes used to generate entity cards use templates for each category of entity [RCB$^+$11, Hen13]. In real-life, a single entity can belong to several categories. In this scenario, it is difficult to build templates for all possible combinations of categories, especially while considering that the importance of categories can vary according to the entity under observation. For example, Barack Obama is both a former President of the United States of America and a book author. In this instance, information related to his presidency is more important than information related to the books he has authored.

In this research, we focus on the problem of automating the process of entity card formation using a semantic knowledge base. This is achieved by ranking the most important attributes for each specific entity. We study if Learning to Rank (LTR) algorithms can accurately replicate human ranking. And within the LTR framework, we introduce a new set of features that exploit semantic information about entities. We also include features extracted from top results of a general search engine. The entity label is submitted to the search engine and the top retrieved results are then used to support the ranking process. The main hypothesis here is that the rank from a generic non-personalized search engine indicates the general importance of attributes. Entity attributes (or their values) which appear in the top results are assumed to have higher importance. In addition, for entities which belong to several categories, the top results can give a hint on which category is most relevant to the entity.

In order to evaluate our approach, we constructed a ground truth dataset. We began by utilizing Wikipedia infobox[1] attributes and their ordering. The ordering of facts in each Wikipedia infobox, and the decision regarding which pieces of information appear in it, are determined through discussion and consensus among the editors of each individual article. In addition to this evaluation, we are currently working on the generation of a human-annotated ranking dataset by exploiting a crowd-sourcing platform.

The main contribution of this research is the introduction of novel features for a learning to rank approach to the ranking of entity attributes. The proposed features incorporate information about attribute importance in general, and their importance with respect to different entity categories. Our preliminary results illustrate the complexity of the problem. The remainder of this paper is arranged as follows; The next section gives a brief overview of relevant research. Then, we present our methodology. In Sect. 4, our experimental setup is detailed. Finally, conclusions and the plan for ongoing research are discussed in Sect. 5.

## 2 Related Work

This work is concerned with one aspect of entity card design, which is the ranking and selection of the most important attributes about an entity from a knowledge base. ECs can also include other pieces of information like maps, images and user reviews [BZJ16], those aspects are beyond the scope of our experiment.

A heuristic-based approach for attribute ranking was introduced by [Pau14]. It divides the attributes into priority buckets according to their name. Each priority bucket is identified by a set of keywords. For example, any attribute that contains the word "name" or "label" will be added to bucket A. The method assumes full awareness of the knowledge base ontology, and the priority buckets have to be defined carefully in advance, which is hard to achieve in Linked Open Data (LOD) datasets like DBpedia.

Lee et al. developed a system to extract attributes from free text and rank them using a typicality score [TL13]. The authors employ a typicality score in order to measure how typical an attribute is to an instance or concept on the basis of their co-occurrence in the dataset. IBMiner is another system that improves knowledge bases using text mining techniques [MGZ13]. It introduced a tool for infobox template suggestion. It collects attributes from different sources and knowledge bases. The attributes are ordered by popularity based on their co-occurrences in the dataset. These approaches require expensive processing and indexing for the document set; they are usually very domain specific and hard to scale for large document collections. A recent approach combines attributes generated from structured knowledge bases with those generated from the documents retrieved from the general web [JDW17]. This idea is particularly interesting because it is scalable and combines the advantages of both data sources. Although the approach was introduced in the query facet mining context, we plan to apply this idea in an entity attribute ranking context.

In earlier work, Kopliku et al. study attribute acquisition techniques from HTML tables from the Web [KBPS11]. The research also investigates computing attribute relevance by linearly combining features from DBpedia, Wikipedia and attribute frequency in relevant documents to the entity. The first two features are binary ones; they only mark if the attribute exists in DBpedia and Wikipedia or not. The attributes were

[1] https://en.wikipedia.org/wiki/Help:Infobox

labeled as either relevant or not. The main focus of the paper is the identification of relevant attributes rather than the ranking of known attributes for each entity. Vadrevu et al. [VTS16] introduce a score that is computed for each attribute and entity pair. It uses an intermediate feature set to compute a single relevance score. Features are generated only from knowledge graphs. That score is passed to a learning to rank algorithm to enhance the ranking results. The scoring method takes into consideration the different categories of the entity without assigning weights to each of them (it treats all categories equally). We follow a similar combined approach between scoring and learning to rank. We developed a weighted version of their score, in which each category is assigned an importance weight. We compare the contribution of the weighted score to the attribute ranking with respect to the unweighted version. In addition to this, we also introduce new features from knowledge graphs and top search results. Another learning to rank approach introduced by Atzori and Dessi [AD14, DA16] combines features from the knowledge base with other external data sources. Like our method, this approach focuses on ranking attributes which already exist in DBpedia and targets solving the same task. We exploit this method as a baseline set of features in our evaluation. We also include a different set of features which incorporates the top ranked search results to enhance the ranking process.

The task of evaluating and assessing attribute ranking algorithms is extremely difficult in the absence of a standard dataset. Researchers addressing this problem constantly use different methodologies and target ground truths to evaluate their approaches. Moreover, many evaluations are conducted on relatively small datasets [AD14, DA16]. In those experiments, a small group of people –5 as reported– labeled the entity attributes. The resulting set of labels is inherently subjective and often presents low agreement values. When the dimension of the dataset is limited (less than 100 entities), these disagreements may significantly affect the experimental results. Human annotators were also used by Kopliku et al. in their evaluation [KBPS11]. We plan to adopt a crowd-sourcing approach to collect a more significant volume of human ranking annotations for entity attributes.

Paul [Pau14] suggested collecting a ground truth dataset from Wikipedia infoboxes. The infobox design is the result of a long term collaborative effort by a large community of users. The key assumption is that infobox templates typically list attributes in order of perceived importance for that entity and that this ranking should be valid for generic reviewing purposes. Our ground truth building process was inspired by this idea.

## 3 DBpedia Attribute Ranking

We exploit a learning to rank approach for the ranking of entity attributes. This experiment starts with building a dataset from DBpedia, then extracting features for each entity and attribute pair. The extracted feature vectors are then fed as input to the LTR algorithms. Finally the trained model is evaluated and the feature set is analyzed to study the usefulness of each feature. We calculate the following features:

A - **AttributeFrequency.** A normalized frequency of the attribute over the knowledge base, DBpedia in our case. It favors common attributes in the knowledge base.

B - **IAF.** Inverse Attribute Frequency for the attribute over DBPedia. This feature assigns high values to attributes which are rare in the dataset, as they might better characterize this specific entity.

C - **AF.IAF.** The attribute frequency-inverse entity frequency measure is calculated by multiplying the previous two features. This feature integrates the information gained from the two previous features. We include the three features in our experiments and investigate how informative each one is in the feature analysis phase.

D - **CountAttributeVal.** To obtain this feature, we count the number of triples this attribute appears in for this specific entity. If the attribute is repeated for the same instance, this might indicate higher importance. If it is repeated too much this might be a negative sign. For example, the World War Two entity has the attribute <dbp:battle>. This attribute has too many values to be shown in an entity card.

E - **AvgCountAttributeVal.** This is the average of the *CountAttributeVal* values. It is obtained by finding the number of triples this attribute has in DBpedia, then dividing this number by the number of entities which contain this attribute.

F - **RankingScore.** This score is inspired by [VTS16], it takes into consideration the different entity categories while calculating the ranking score for the attribute. For example, considering an entity that belongs both to an author and a TV host category. As the number of persons in DBpedia is significantly larger than the number of TV hosts, attributes such as <dbp:Date_of_birth> will dominate other TV host related attributes. The suggested score linearly combines the AF.IAF calculation of each attribute to its category.

G - **WeightedRankingScore.** This is the same as the previous score with the exception that it assigns a weight, or importance, to each category. For example, for an entity like "Barack Obama" the attributes related to his presidency are more important than those related to his book authorship. Since the number of entities with the type "president" is less than those with "book author", the previously calculated frequencies will not have considered such a case. The feature gives higher weights to the most relevant categories.The weights can be calculated using one of the following methods:

(i) A naive approach to compute these weights based on the inverse category frequency. The idea is that less frequent categories are better discriminators of entities.

(ii) A more educated method for weighting the category would infer the category relevance from the top-ranked search results. It could use Paragraph Vector to build a fixed vector representation for both the entity and the category from the top ranked documents [LM14]. Then, the weight of a category can be computed on the entity base as the similarity between these two vectors.

H - **SearchEngineOccurance.** The entity label is submitted to a generic search engine and the top documents are retrieved. The occurrences of the attribute, or its value, are counted for each document. The feature is computed as the sum of this count over all documents multiplied by the document rank. The task of finding the attributes or their values in the text is not a trivial one. Simple string matching would be extremely noisy. This feature will use semantic parsing to extract and link triples from the text. This research is ongoing at the time of writing.

Our approach aims to experiment with several learning to rank algorithms and then to study the characteristics and performance of each algorithm on our dataset. We want to diversify the nature of the LTR methods used, which are either based on neural networks, decision trees, ensembles, or boosting. Features G (ii) and H are still under development, they will be part of the extensive evaluation performed on the crowd-sourced experiment that we are currently planning.

## 4 Experimental Setup and Preliminary Results

The initial experimental evaluation consisted of the following process. First, a dataset of entities was collected from DBpedia and the related Wikipedia infoboxes. The assumption is that the ordering of the facts inside the infobox represents their perceived relevance or importance for that entity [Pau14]. We randomly selected entities from 62 different categories. For each entity, its attributes are mapped into a Wikipedia infobox and assigned a rank if they appear in the infobox. Each mapped attribute is assigned with its order of appearance as a field in the Wikipedia infobox. This order is treated as the target label. We convert the target label into a relevance score by reversing the order of ranking. Then, feature vectors are calculated for each entity and attribute pair. All the remaining DBpedia unmapped attributes are added with a score equal to 0 as target label. During experimentation we faced some issues in mapping the attributes. In addition to that, some redundancies discovered in DBpedia attributes were removed. The dataset contains 9,133 entities with 276,534 attributes. LTR algorithms are used for training and testing the shuffled data using ten-fold cross-validation.

Table 1 shows results for the features from A to G(i) (see section 3). We intend to include the last two features, G(ii) and H, in our final evaluation. All measures are calculated for the top 10 ranked attributes. We report only the average values across the ten folds, although we also computed the standard deviation, which never exceeded 0.02. This indicates values very closely clustered around the average.

Table 1: Results for different LTR methods evaluated at k=10.

| Ranker | $\rho$ | **nDCG** | **R@k** | **P@k** | **F@k** |
|---|---|---|---|---|---|
| MART | **0.94** | 0.84 | **0.83** | **0.92** | **0.88** |
| RF | 0.93 | 0.82 | 0.81 | 0.92 | 0.86 |
| LMART | 0.90 | **0.86** | 0.66 | 0.90 | 0.76 |
| CA | 0.68 | 0.50 | 0.03 | 0.78 | 0.39 |
| RB | 0.68 | 0.46 | 0.23 | 0.70 | 0.35 |
| LR | 0.52 | 0.38 | 0.16 | 0.72 | 0.26 |

LTR algorithms included in our evaluation are: LambdaRank (LR), MART, LambdaMART (LMART) , RankBoost (RB), RandomForest (RF), and CoordinateAscent (CA). We also experimented with RankNet, AdaRank and ListNet, but they performed poorly and are thus excluded from the results presentation.

Ordering plays a key role in our evaluation since our task aims to predict the ideal rank for a set of attributes. For this purpose, we choose the Spearman Rho ($\rho$) ranking correlation coefficient as an evaluation metric.

Moreover, in order to take into account relevance and ranking, we borrowed some metrics from those most widely used in the IR community: nDCG, Recall@K (R@k), Precision@K (P@k) and FMeasure@K (F@k).

The tree-based learning to rank methods (MART, LMART and RF) outperform the other methods across all evaluation metrics. In order to learn more about the usefulness of the used features, we follow the approach suggested by [GLQL07]. To obtain the feature importance the approach uses each feature as a ranking model, then the model is evaluated using the evaluation metrics. The score of evaluation is considered as the importance of the feature. In our experiments, we use Spearman Rho, nDCG@k and Precision@K as evaluation scores. Table 2 reports the results of this analysis. In order to further investigate the contribution of each feature to the final result, we experimented with each single feature within the LTR approach (using MART as ranker). The obtained results are presented in Table 3.

Table 2: Features as rankers results.

| **Feature** | $\rho$ | **nDCG** | **P@K** |
|---|---|---|---|
| AttributeFrequency | **0.67** | **0.49** | **0.78** |
| IAF | 0.00 | 0.00 | 0.00 |
| AF.IAF | 0.01 | 0.09 | 0.31 |
| CountAttributeVal | 0.03 | 0.14 | 0.33 |
| AvgCountAttributeVal | 0.00 | 0.08 | 0.20 |
| RankingScore | 0.41 | 0.31 | 0.65 |
| WeightedRankingScore(i) | 0.50 | 0.36 | 0.70 |

Table 3: MART results for each feature.

| **Feature** | $\rho$ | **nDCG** | **P@K** |
|---|---|---|---|
| AttributeFrequency | 0.73 | **0.60** | 0.82 |
| IAF | 0.75 | 0.58 | 0.83 |
| AF.IAF | **0.78** | 0.59 | **0.85** |
| CountAttributeVal | 0.13 | 0.23 | 0.46 |
| AvgCountAttributeVal | 0.52 | 0.38 | 0.71 |
| RankingScore | 0.50 | 0.40 | 0.71 |
| WeightedRankingScore(i) | 0.54 | 0.42 | 0.72 |

Overall, the different evaluation metrics in Table 2 show that no single feature outperforms the LTR results. It is also noted that WeightedRankingScore is more informative than the RankingScore. AttributeFrequency seems to be more useful than the IAF and AF.IAF, which both gave poor results. However, we noticed that when these features are removed the results decrease. Indeed, while these features are not able to give good performance when used as a ranker (Table 2), from Table 3 we can observe that these two features bear useful information for predicting the attribute rank in a LTR approach. The table gives an indication of how each feature contributes to the LTR algorithm results. It suggests that AF.IAF outperforms other features. However, we cannot neglect the fact that experimenting with different permutations of features could yield improved results. Tables 2 and 3 can be used as a starting point for such investigation. More advanced methods for feature analysis such as an ablation study or input sensitivity analysis could also be performed.

In addition to this evaluation, we are working on a crowd-sourcing experiment to collect human ranking annotations for attributes. In the experiment, given an EC, we ask the user to rank its attributes in order of perceived importance. We predefined a list of categories. For each category, we select two groups of entities from DBpedia: (1) entities that belong only to the considered category; (2) entities associated with more than one category. For each user, the pool of entity attributes will be randomly ordered to ensure an unbiased ranking. To detect users who randomly pick attributes, each user will be asked to rank an entity twice and the agreement between attribute ranks will be measured. The experiment aims to answer the following questions: 1) What is the appropriate number of attributes to be displayed in an EC? 2) What are the most important attributes for the selected entities? 3) Is the ranking strictly correlated with the category or does it also depend on the entity/instance?

## 5 Conclusions

In this paper we presented a new approach for entity attribute ranking. Novel features were proposed for each entity and attribute pair. The features are then used to train learning to rank models. To validate our method, we collected a dataset from Wikipedia infoboxes and DBpedia. This was a useful exercise in and of itself, to understand the nature of DBpedia data and to asses the ability of features to capture a given rank. Our preliminary results have shown that tree-based LTR algorithms can achieve good ranking performance, when compared to collaborative human ranking. Moreover, during the dataset collection, we encountered redundancies in DBpedia attributes and noticed that different methods for handling redundancies resulted in quite different performance. A preliminary evaluation of these different methods for handling redundancy has illustrated how the quality of the knowledge base used can affect the performance of the LTR methods.

The experimental evaluation highlighted that statistical features calculated only from the knowledge base can provide meaningful results. It is our intuition that complementing these features with others extracted from external data sources (like the top ranked search results) will enrich the ranking process. As future work we plan to conduct an extensive evaluation on our crowd-sourced ground truth. An experiment is underway which aims to collect entity attribute ranks directly from end users.

**Acknowledgements**

This research is supported by the ADAPT Centre for Digital Content Technology, which is funded under the Science Foundation Ireland Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Fund. Our work has also received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 713567.

## References

[AD14] Manfredo Atzori and Alessia Dessi. Ranking dbpedia properties. In *Proc. of WETICE*, pages 441–446. IEEE, 2014.

[BCMT13] Roi Blanco, Berkant Barla Cambazoglu, Peter Mika, and Nicolas Torzec. Entity recommendations in web search. In *Proc. of ISWC*, pages 33–48. Springer, 2013.

[BZJ16] Horatiu Bota, Ke Zhou, and Joemon M. Jose. Playing your cards right: The effect of entity cards on search behaviour and workload. In *Proc. of CHIIR*, pages 131–140. ACM, 2016.

[DA16] Andrea Dessi and Maurizio Atzori. A machine-learning approach to ranking rdf properties. *Future Gener. Comput. Syst.*, 54(C):366–377, 2016.

[GLQL07] Xiubo Geng, Tie-Yan Liu, Tao Qin, and Hang Li. Feature selection for ranking. In *Proc. of SIGIR*, pages 407–414. ACM, 2007.

[Hen13] J.W. Henry. Providing knowledge panels with search results, May 2 2013. US Patent App. 13/566,489.

[JDW17] Zhengbao Jiang, Zhicheng Dou, and Ji-Rong Wen. Generating query facets using knowledge bases. *Transactions on Knowledge and Data Engineering*, 2017.

[KBPS11] Arlind Kopliku, Mohand Boughanem, and Karen Pinel-Sauvagnat. Towards a framework for attribute retrieval. In *Proc. of CIKM*, pages 515–524. ACM, 2011.

[LM14] Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196, 2014.

[MGZ13] Hamid Mousavi, Shi Gao, and Carlo Zaniolo. Ibminer: A text mining tool for constructing and populating infobox databases and knowledge bases. *Proc. VLDB Endow.*, 6(12):1330–1333, 2013.

[Pau14] Falco Paul. Property ranking approaches for semantic web browsers-a review of ontology property ranking algorithms. 2014.

[PMZ10] Jeffrey Pound, Peter Mika, and Hugo Zaragoza. Ad-hoc object retrieval in the web of data. In *Proc. of WWW*, pages 771–780. ACM, 2010.

[RCB$^+$11] F. Radlinski, N. Craswell, B. Billerbeck, M. Shokouhi, S. Ahari, N. Agrawal, T. Hoad, S. Zhou, and M.A. Awan. Entity detection and extraction for entity cards, December 15 2011. US Patent App. 12/813,390.

[TL13] Haixun Wang Taesung Lee, Zhongyuan Wang. Attribute extraction and scoring: A probabilistic approach. In *Proc. of ICDE*, January 2013.

[VTS16] Srinivas Vadrevu, Ying Tu, and Franco Salvetti. Ranking relevant attributes of entity in structured knowledge base, January 5 2016. US Patent 9,229,988.

[WR09] Ryen W White and Resa A Roth. Exploratory search: Beyond the query-response paradigm. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 1(1):1–98, 2009.