PROPERTY RANKING APPROACHES FOR SEMANTIC WEB BROWSERS

A review of ontology property ranking algorithms

F.B.A. (Falco) Paul, BSc (837505652)
9 December 2014
Property Ranking Approaches For Semantic Web Browsers
A Review of Ontology Property Ranking Algorithms

Open Universiteit, faculteit Management, Science & Technology

Masters Business Process Management and IT

Supervising committee:

Supervisor: dr. (Lloyd) Rutledge
Second reader: dr. ir. F.J.M. (Frans) Mofers
Examinator: dr. ir. F.J.M. (Frans) Mofers
Course code: T9232B
Acknowledgements

A research project like this is never the work of anyone alone. The contributions of many different people, in their different ways, have made this possible. Not in the least by those that have paved the way in writing earlier related works.

I would like to extend my appreciation especially to my supervisor Lloyd Rutledge for making this research possible. I greatly appreciated his support, guidance and advice throughout the research project, as well as his pain-staking effort in proof reading the drafts. In addition I also thank Frans Mofers for his critical feedback and advice to improve the quality of this thesis.

Last but most certainly not least, I would like to thank my family for all the patience and understanding shown during the two years that this effort took. I know that at times you all suffered from a lack of attention.
Summary

The Semantic Web is a collaborative movement led by the World Wide Web Consortium (W3C) that promotes common formats for data on the World Wide Web. By encouraging the inclusion of semantic content in web pages, the Semantic Web aims at converting the current web of unstructured documents into a “web of data”. The Semantic Web builds on the W3C’s Resource Description Framework (RDF). RDF is a family of specifications used to model information through a variety of syntax formats. The RDF metadata model is based upon the idea of making statements about resources, also known as subjects, in the form of subject-predicate-object expressions. These expressions are called triples in RDF terminology and typically describe real world objects. The subject denotes the resource and the predicate denotes traits or aspects of the resource and expresses some relationship between the subject and the object.

The Web Ontology Language (OWL) is a family of dialects designed for defining and instantiating formal Semantic Web ontologies. Ontologies are essentially formalized data models that represent a set of concepts within a domain as well as the relationships between those concepts. An ontology class individual (or instance) is defined using RDF triples. An OWL class is a specification for a group of individuals that share common characteristics or properties. Ontologies define the structures, rules, functions, restrictions and axioms to enable reasoning power with respect to its individuals.

Semantic Web browsers are tools used to visualize and navigate ontologies and RDF data. OWL has no native support to specify salience, ranking information or display hints for ontology class properties. Semantic Web browsers therefore miss critical interfacing hints to optimize their interface with respect to an OWL ontology being examined by a user. Semantic Web browsers have to rely on bespoke proprietary mechanisms to address the issue of view property selection and ranking. Often Semantic Web browsers will use an (naïve) alphabetic ranking to list the object properties of an ontology class; a practice that results in suboptimal user experiences.

How can Semantic Web browsers optimize their data presentation if they do not have the information that enable the generation of streamlined views? This research has sought to address the issue of property ranking (and selection) when confronted with missing information, specifically in order to support automated view generation within Semantic Web browsers. To what extent can an automated ranking approximate a human designed ranking? Specifically if we want to base the computation of the object property ranking from the ontology alone, so without resorting to the inspection of RDF triples.
This paper presents a thorough overview of previous work related to this topic. Our study of applicable work has revealed that there is only a limited amount of directly applicable research in this area. Furthermore, most of this earlier work is somewhat fragmented in nature. Our study of previous work ties the various initiatives together through a literature classification framework.

As a result of our literature gap analysis we developed four ontology property ranking algorithms that base the ranking on the textual analysis of the given ontology. We have also evolved an existing heuristics-based approach towards property ranking that derives a rank order based on the structural analysis of an ontology. We introduce these algorithms in this paper in conjunction with a full methodic review. Each ranking algorithm is studied in detail using 19 performance indicators in order to perform a holistic review of the key ranking performance dimensions.

An important finding from this research is that three out the five studied ranking algorithms can approximate a human designed ranking to a moderate extent. In addition, we show that these approaches offer a significant improvement over alphabetical ranking, with the heuristics-based ranking algorithm being the overall best performing method.

We believe that our study results\(^1\) can help tool designers within the Semantic Web community to optimize their ontology presentation methods in order to improve the overall user experience.

\(^1\) Research artefacts: see http://is.cs.ou.nl/OWF/index.php5/Masters_Thesis_Falco_Paul
Table of contents

Acknowledgements ........................................................................................................4
Summary ..........................................................................................................................5
Table of contents ...........................................................................................................7
List of figures ...................................................................................................................9
List of tables ..................................................................................................................11
Chapter 1 Introduction ................................................................................................12
  1.1 The Semantic Web ...............................................................................................12
  1.2 The research problem .........................................................................................13
  1.3 Research approach summary .............................................................................16
  1.4 Study research goals .........................................................................................21
  1.5 Research questions ............................................................................................24
Chapter 2 Related and prior work .............................................................................25
  2.1 Introduction ..........................................................................................................25
  2.2 Literature study methodology ..........................................................................25
  2.3 Approaches to entity property ranking ...............................................................28
  2.4 Conclusions of related work and gap analysis ...................................................35
Chapter 3 Evaluation model .......................................................................................40
  3.1 Introduction ..........................................................................................................40
  3.2 Mapping DBpedia onto Wikipedia templates ....................................................43
  3.3 Test data set construction (English) ..................................................................45
  3.4 Test data construction (multi-lingual experiment validations) .........................49
  3.5 Evaluating ranking algorithms ..........................................................................51
  3.6 Rank correlation metrics ....................................................................................51
  3.7 Proximity and group stability metrics ................................................................54
  3.8 Classification / IR metrics ..................................................................................56
  3.9 Comparing ranking algorithms .......................................................................58
  3.10 Benchmarks / extremes ....................................................................................59
Chapter 4 Conducted experiments ............................................................................65
  4.1 Introduction ..........................................................................................................65
  4.2 Terminological-based ranking: word frequency matching .........................65
4.3 Terminological-based ranking: N-gram probability ..................... 73
4.4 Terminological-based ranking: term similarities ..................... 78
4.5 Terminological-based ranking: dynamic corpus ..................... 83
4.6 Ontology structure analysis ranking: heuristic approach ........ 91
4.7 Multi-lingual evaluation for ranking algorithms .................. 97

Chapter 5  Research summary ............................................. 102
  5.1 Conclusions............................................................. 102
  5.2 Future work............................................................ 104

Chapter 6  Appendixes ....................................................... 105
  6.1 Appendix 1: references ............................................ 105
  6.2 Appendix 2: typical search terms per research question ........ 108

Glossary............................................................................. 110
List of figures

Figure 1: the Semantic Web stack ........................................... 12
Figure 2: TopBraid browser with Edward Kennedy Semantic Web data .......... 14
Figure 3: Google Knowledge Graph for Edward Kennedy........................ 14
Figure 4: summarized problem definition .................................... 15
Figure 5: high level research approach ...................................... 16
Figure 6: ontology classes associated with matching ranking baselines .......... 18
Figure 7: Wikipedia article for Edward Kennedy with Infobox (right) .......... 19
Figure 8: ranking algorithm maturing ........................................ 20
Figure 9: literature study outline ............................................ 25
Figure 10: key data structures involved in the construction of the test data set ...... 40
Figure 11: interaction with data structures from research project perspective ..... 41
Figure 12: DBpedia maps ontology classes to Wikipedia Infobox templates .... 42
Figure 13: DBpedia mapping (simplified) for Infobox template "Infobox_brain" . 43
Figure 14: simplified extract of Infobox template "Infobox_brain" ............... 43
Figure 15: summary of the DBpedia ontology class "Brain" ..................... 44
Figure 16: research test data set construction .................................. 45
Figure 17: KPI metrics for perfect ranking benchmark ......................... 60
Figure 18: KPI metrics for reverse ranking benchmark ......................... 60
Figure 19: KPI metrics for random ranking benchmark .......................... 61
Figure 20: KPI metrics for half-perfect ranking benchmark ..................... 62
Figure 21: KPI metrics for half-reverse ranking benchmark ..................... 63
Figure 22: KPI metrics for alphabetical ranking benchmark ...................... 64
Figure 23: KPI metrics for word frequency ranking (Wikipedia frequency list) . . 68
Figure 24: KPI improvements for frequency ranking over alphabetical ranking ... 69
Figure 25: paired samples t-test for word frequency ranking .................... 69
Figure 26: Kendall $\tau$ differences distribution for word frequency ranking .... 70
Figure 27: KPI metrics for word frequency ranking (Bing frequency list) .... 70
Figure 28: KPI metrics for word frequency ranking (COCA frequency list) .... 70
Figure 29: KPI metrics for reverse ranking (Wikipedia frequency list) ......... 71
Figure 30: KPI metrics for average term frequency (Wikipedia frequency list) .. 72
Figure 31: KPI metrics for metrics joined probability ranking (body) ............ 74
Figure 32: KPI improvements for N-gram ranking over alphabetical ranking ...... 74
Figure 33: paired samples t-test for N-gram ranking ............................ 75
Figure 34: normalized Kendall $\tau$ differences distribution for N-gram ranking ... 75
Figure 35: KPI metrics for joined probability ranking (titles) ................... 76
Figure 36: KPI metrics for joined probability ranking (anchors) .................. 76
Figure 37: KPI metrics for conditional joined probability ranking (body) ....... 77
Figure 38: KPI metrics for reverse-sorted joined probability ranking (body) .... 77
Figure 39: KPI metrics for term similarity-based ranking ....................... 82
Figure 40: dynamic terminological ranking algorithm ........................... 83
Figure 41: KPI metrics for dynamic terminological ranking ..................... 90
Figure 42: KPI metrics for heuristics-based ranking ............................ 95
Figure 43: KPI improvements for heuristic ranking over alphabetical ranking .... 95
Figure 44: paired samples t-test for heuristic ranking ................................................. 96
Figure 45: normalized Kendall τ differences distribution for heuristic ranking .... 96
Figure 46: governing logic for multi-lingual experiment ............................................. 98
Figure 47: metrics for alphabetic ranking in a multi-lingual ranking baseline ...... 99
Figure 48: metrics for alphabetic ranking in an English ranking baseline .......... 99
Figure 49: multi-lingual frequency ranking (Δ compared to alpha ranking) .... 100
Figure 50: multi-lingual N-gram ranking (Δ compared to alpha ranking) ........ 100
Figure 51: multi-lingual heuristic ranking (Δ compared to alpha ranking) ....... 101
Figure 52: heuristic ranking t-test details (multi-lingual context) ....................... 101
Figure 53: differences distribution for heuristic ranking (multi-lingual context) 101
List of tables

Table 1: entity property ranking algorithms .......................................................... 28
Table 2: sample mapping of Infobox properties to ontology class properties ....... 45
Table 3: Infobox templates for the same topic may rank properties differently .... 49
Table 4: multi-lingual data sets .............................................................................. 49
Table 5: available ranking variants for a given DBpedia class ......................... 50
Table 6: schematic form for perfect ranking ......................................................... 59
Table 7: schematic form for reverse ranking ......................................................... 60
Table 8: schematic form for random ranking ........................................................ 61
Table 9: schematic form for half-perfect ranking .................................................. 62
Table 10: schematic form for half-reverse ranking ............................................... 63
Table 11: word frequency ranking in schematic form ......................................... 65
Table 12: N-gram ranking in schematic form ........................................................ 73
Table 13: different meanings for the term “model” ............................................. 79
Table 14: multi-lingual experiment approach ....................................................... 98
Table 15: comparison of ranking algorithms ....................................................... 102
Chapter 1 Introduction

1.1 The Semantic Web

The promise of the Semantic Web is advanced knowledge processing through rich Linked Data structures to enable a level of information connectivity that far surpasses the capabilities available through the conventional web standards. The term Linked Data is used here to describe a set of recommended best practices for exposing, sharing, and connecting pieces of data, information, and knowledge within the Semantic Web. When stripped down to its bare essence the Semantic Web [1] is predominantly a technology stack depicted in Figure 1 that extends existing web standards to enable data linkage, data exploration and data analysis across applications and communities. Each Semantic Web standard layers additional depth and breadth to the processing capabilities within the Semantic Web context. The Semantic Web standards enable application architects to create new and novel solutions that allow users to benefit from new forms of data processing.

Semantic Web data is described using RDF triples. Each triple is a combination of a subject (aka ‘identity’), a predicate (‘property being described’) and an object (‘value’). Triples form the core of the Semantic Web on which the technology stack further elaborates.

Figure 1: the Semantic Web stack

2 http://en.wikipedia.org/wiki/Semantic_Web_Stack
Triples in the Semantic Web are typically further specified and interlinked by knowledge engineers using ontologies. An ontology is commonly defined as a “formal, explicit specification of a shared conceptualization” [2]. Ontologies allow modeling of everyday real concepts using machine-readable data structures. Ontology models typically offer well known constructs from information science such as classes, objects, instances, properties and relationships to define rich data structures.

A set of applicable standards known as the Web Ontology Language (OWL) family has evolved over time to enable the authoring of rich data models. The OWL family contains various variant syntaxes and specifications, and a full description is far beyond the scope of this paper. Generally, the data described through an OWL-based ontology is interpreted as a set of individuals in conjunction with a set of assertions that relate these individuals to each other. An OWL-based ontology structure may become so rich that software reasoners can actually infer various kinds of indirect facts and relationships from the models various constituents. Further information on all of these topics can be found on the W3C internet pages3.

1.2 The research problem

A typical use case within the Semantic Web involves the browsing of Linked Data. Users operating within typical browsing scenarios are predominantly concerned with the key properties of a topic or ontology under review. In most use cases it would be inappropriate to show all the triples of an entity to the user as some subjects can contain hundreds of triples. For example, consider a user browsing the current US president’s Linked Data description. This user would find more than 200 facts attached to that entity. In addition, most users will prefer (and to some extent, expect) a logical ordering of these facts.

Although the Semantic Web technology stack is very expressive, the OWL standards cannot be used to express interfacing or visualization hints, data views or forms to support typical human interactions with ontology data. The OWL family of languages does also not allow the specification of (specific or relative) importance with respect to properties. OWL ontologies lack directly available information to determine which properties of a class are of particular importance within the context of a user interface. As a result, interaction tools such as Semantic Web browsers cannot automatically generate topic-attuned views to interface with the user since the necessary display order for properties is simply missing. Semantic Web browsers thus have to rely on proprietary mechanisms to address the issue of property ranking4 and selection. Various browsers now simply rank the properties alphabetically. The resulting views, be it a form, report, online view or editor, may not need the real user requirements or expectations.

3 http://www.w3.org/2001/sw/
4 This research will further use the more precise term ‘ranking’ as an alternative for ‘ordering’
As stated, it is typical to find that Semantic Web browsers present users with naively ordered lists of properties. Consider Figure 2; perhaps a typical ‘category example’ where a Semantic Web browser5 displays data for a Linked Data entity. The naively alphabetic property ordering in the right pane is not assisting the usability in our view. Users do not want the raw information but instead need the information to be restructured and summarized [3]. Note in this context that summarization is a special case of ranking. Consider now instead Figure 3 which displays a Google Knowledge Graph6. Google injects Knowledge Graphs in search results to enhance the user experience with summarized semantic-search information gathered from a wide variety of sources [4]. The ranked property list in Figure 3 in our opinion is definitely more useful to the casual user than the alphabetic list in Figure 2.

5 http://www.topquadrant.com/technology/topbraid-platform-overview/
6 http://www.google.com/insidesearch/features/search/knowledge.html
As another example of this problem, consider an ontology that describes movies. Movie ontologies will have classes such as Movie, Director, Actors, Editor, Producer and Screenplay author. A Semantic Web browser cannot rely on the ontology to supply clues as to the display order, or more formally, the rank order, of the Movie class from our example. A Semantic Web browser has no knowledge whether it should first display the sound editor or the director property of a movie. Obviously, a casual data observer will probably care more for the director property, but a semantic browser has no direct access to this knowledge.

Usability is a topic within the Semantic Web scientific community that has not seen very extensive research. Our focus in this research is to seek methods that alleviate the restricted means in the OWL language family to specify interface ranking hints for properties with respect to user interfacing. The current property ranking algorithms in most Semantic Web browsers clearly require improvements as unfiltered alphabetically sorted list of properties are not very helpful to expand the Semantic Web user experience. The full problem chain is listed below in Figure 4.

\[\text{Figure 4: summarized problem definition}\]

The aim of this research is to develop ranking algorithms that approximate human ranking for the purpose of enhancing the automated generation of user interfaces within Semantic Web browsers. We will now state the formal research problem definition:

To what extent can an ontology property ranking algorithm approximate a human-designed ranking to support automated view generation in Semantic Web browsers?

\[\text{7 Within the scope and context of this paper the term properties is typically loosely used to describe both the object type and the data type properties of a given ontology class.}\]
1.3 Research approach summary

The previous sections were used to present the research problem that this paper addresses. Most of the subsequent chapters in this paper will provide details with respect to how the various research constituents have been structured. Before we deep dive into the finer details of the study we will now provide an overview of how this research has been conducted. This section therefore offers a high level roadmap that outlines how we approached the presented research problem.

As with most studies, this research too started with a comprehensive literature study that we fully present in Chapter 2. Literature studies help to gain an understanding of the topic of interest and its contextual scope, but they are also essential to identify research opportunities and gaps that may exist within the literature. We employed a thorough study of prior work to understand how ranking and the related problem of feature summarization\(^8\) has been employed and studied before. Our literature review also thoroughly assessed how rankings have been evaluated in other studies. This comprehensive study of prior work gained us insight with respect to how the research goals and theoretical questions could be addressed. These goals are defined more thoroughly in section 1.5.

We now first outline the high level research approach in Figure 5.

![Figure 5: high level research approach](image-url)

Obviously these phases were not strictly sequential but the diagram serves to illustrate the overall order in which the activities took place. Each block is discussed in far more detail in the following chapters. The overall utilized research design is quantitative in nature. We employed the constructive research\(^9\) method as the governing approach for this study. As a further research method classification we note that the analytical techniques and perspectives typically employed in Design Science Research (DSR) [5] approaches have been used in this research too.

\(^8\) The selection of the most important properties of some entity
\(^9\) Research procedure for producing innovative constructions, intended to solve problems faced in the real world and, by that means, to make a contribution to the theory of the discipline in which it is applied.
The literature study conclusions are presented in full detail in section 2.4 but we present the key outcomes already at this point, as they shaped the overall research approach:

- We identified various existing approaches that address the problem of property ranking within the Semantic Web domain as well as within related domains. One particular paper that we found very interesting describes a heuristics-based approach to ranking, which is described in more detail in paragraph 2.3.2. We have augmented and extended this approach to make it more generic and applicable.

- There is a limited amount of research that employs natural language analysis to derive ranking information - a gap in the existing literature on which we picked up. Four out of the five ranking algorithms that we studied in this research are therefore based on textual analysis of terminology that we extract from ontologies.

- Ranking is subjective in nature, and the ‘right’ ranking depends on contextual circumstances. For example, an expert user may have a different view on which properties are more important than a non-expert user.

- Various metrics may be computed to compare the correspondence (or distance) between two given rankings. Important rank correlation indicators are the Kendall Tau and Spearman Rho metrics (these are described in section 3.6).

- One method to determine which of two rankings is the better ordering is through using a group of persons assigned to the task of rating rankings. Unfortunately, this is a very costly, but even more so, extremely time-consuming process. Hence, we decided to bring this type of evaluation out of scope.

- In order to automate the evaluation of ranking algorithms an appropriate “optimal” ranking is required that can act as a baseline reference point. This is also known as a golden standard or a ground truth set. By computing the distance of a derived ranking with respect to the baseline ranking we can determine how effective the algorithm is (closer to the baseline is better).

- The better the ranking baseline aligns with the “expected” ranking from users (that typically participate in the use case and overall context assumed in our research), the better.
The conclusions from the literature study made it clear that this research required a comprehensive test data set to enable advanced ranking algorithm evaluation. The research required an OWL ontology augmented with ranking baselines (as illustrated in Figure 6) to drive the ranking algorithms review processes. A substantial part of our research effort was devoted to construct such a data set.

![Figure 6: ontology classes associated with matching ranking baselines](image)

We constructed a test data set for the purpose of evaluating our ranking algorithms similar in structure to Figure 6 from classes that reside in the DBpedia ontology. DBpedia [6] is a crowd-sourced community effort to extract structured information from Wikipedia and make this information available on the Web. It extracts Infobox instance data from Wikipedia to expose that data as Linked Data triples. How DBpedia extracts this data is relevant to our study and we therefore explain this further in section 3.1.

Infoboxes (see Figure 7) are Wikipedia article callouts that summarize the key features of the article’s subject. Infoboxes are comparable to fact sheets that summarize the key features of some object. The display order for the properties within a particular Infobox template is decided by the Wikipedia topic community. The Infobox templates are free to use and have computer parsable structures that facilitate the automatic extraction of their properties. It is important to understand at this point that there is a key difference between Infobox templates and Infobox instances. A template defines how properties are displayed in an Infobox callout on a Wikipedia article. An Infobox (instance) defines the actual property-value pairs. We developed software that can parse these Infobox templates as to extract the rank position of its contained properties.

---

10 [http://wiki.DBpedia.org](http://wiki.DBpedia.org)
DBpedia publishes the mapping information that projects the properties of Infobox instances onto the properties of the DBpedia ontology classes. We build software that extracts these DBpedia mappings in order to relate the rank positions extracted from Wikipedia Infobox templates to the properties of the DBpedia ontology class properties. The ranking data from the Infobox templates is used in conjunction with the mapping data from DBpedia to extend the test data set conform to the model shown in Figure 6. The full test data set creation process is explained in detail in section 3.3.

The described test data set enables the computation of key ranking performance metrics with respect to the studied ranking algorithms. The different metric types are described in more detail in Chapter 3. The metrics denote the performance of the ranking algorithms versus the ranking baseline that is contained inside the Wikipedia Infobox templates (which is a human-designed ranking). We will postulate within this paper that this baseline is fit for our evaluation purpose. A secondary goal of this research was to design property ranking algorithms that improve over alphabetical ranking in a statistically significant manner. We view alphabetic ranking as a ‘standard’ to beat since this ranking algorithm is recurrently used by Semantic Web browsers. Effective algorithms that statistically outperform alphabetic ranking will be of particular interest to the Semantic Web tool development community. We give a full description of what we denote with statistical outperformance in section 3.9.
The final major ingredient of this research constitutes the studied ranking algorithms. We formulated most algorithmic variants by means of approaching the problem from different optimization angles. As explained we researched four methods that are predominantly text-driven as well as a heuristics-based method that examines the ontology structure. The text-driven algorithms were inspired partly by other works and partly through research in the field of Natural Language Processing (NLP) 12. Our research aimed to help bridge the gap in the literature with respect to the lack of described NLP-based approaches for the purpose of property ranking. The textual methods range from relatively simplistic, such as word frequency counting-based to rather complicated algorithms that examine search engine results from derived ontology terminology. Note that the five described methods are unrelated and have no interrelation dependencies.

Post the study of related work we conducted a comprehensive pre-study that yielded several ranking algorithm proof-of-concepts. These early conceptual ranking algorithms were further refined using an empirical approach where the evaluation framework was used to assess the performance of these methods. In Figure 8 we present an outline of how the ranking algorithms matured during the study. This cycle was reiterated as the research evolved to improve these ranking methods until no further improvements seemed possible.

![Figure 8: ranking algorithm maturing](image)

12 The field of Natural Language Processing (NLP) focusses on computational linguistics with the goal to devise techniques to automatically analyze large quantities of spoken (transcribed) or written text in ways that parallel what happens when humans perform this task.
The ranking algorithms were reviewed through the evaluation framework so that improvements could be defined over a series of consecutive development cycles, until such point that we observed no further significant improvements and alternative effective property ranking variants seemed unlikely to further evolve.

The result of this iterative process, in combination with the initiating literature study, yielded sufficient information to answer the theoretical questions. The results of the research as summarized in this paper in Chapter 5, including a discussion towards possible future work for potential follow-up studies.

1.4 Study research goals

One of the key aims of this study is to assess if specific property ranking algorithms can support automated view generation specifically in the context of Semantic Web browsers. The typical use case that we foresee for these views involves “casual users” reviewing the data or structure of some ontology of interest. We envision a target user audience that typically has limited or generic knowledge of the ontology at hand. The underlying rationale is that we see Semantic Web browser as principally generic tools, somewhat comparable to search engines, but within a Semantic Web context. We believe that these users will expect that the presented views are organized, optimized and presented towards the “crowd”, rather than for domain specialists or topic experts. The product of our research, which is essentially ranking output, should therefore target Joe’s in the street intuitions with respect to expected rank order. To judge the effectiveness of our ranking output we must therefore compare our produced rankings against a baseline that is fit for this particular purpose.

We also specifically constrained the ranking algorithms from executing any type of ontology instance data inspection in order to derive ranking knowledge. This is another key distinction with many related papers in the domain of the Semantic Web that deal with ranking. The constraint to focus on the ontology structure for inferring ranking information is deliberate and intentional, and related to the use case that we described above. We believe that a casual user will expect swift and ideally near real-time performance from his Semantic Web browser.

13 Typically we found methods that rank properties through inspection of RDF triples.
The typical use case for Semantic Web browsers that we foresee, looking towards the future, will most likely be about predominantly ad-hoc usage, probably very much like todays web browsers are used. The total user experience, including the browsing performance, will determine which browsers will become popular. As ontology instance data inspections can become quite time consuming for large ontologies, we purposely excluded all forms of instance data introspection for the above reasons.

A final key constraint in the research is that all ranking algorithms operate within the context of English language, much for the same reasons as above, but also because most of the research in this area is English-driven. Do note however that we reviewed the best performing ranking algorithm respective to several non-English ranking baselines as to obtain a wide perspective on overall ranking performance.

As stated in section 1.3, rankings may be evaluated via an objective or a subjective approach. Subjective ranking evaluations require a group of persons assigned to the task of rating rankings. Typically, this is a difficult, costly but above all time consuming approach. There are also several (sometimes subtle) side effects that must be taken into account before drawing conclusions from this type of approach. But above all, the development of ranking algorithms without a near real time manner to appraise and improve these methods is next to impossible. Effective evaluation of ranking algorithms requires the definition of appropriate performance indicators to score the method effectiveness. The evaluation and validation process will assess various aspects of ranking algorithms. As stated before, the overall objective is to determine to what extent automated ranking algorithms can approximate human designed rankings. This validation will include standard statistical and information-theory ranking comparisons in conjunction with specific ‘user centric’ metrics. The definition of a ranking algorithm evaluation mechanism is therefore another important goal of the study.

In order to compare and evaluate multiple ranking algorithms we require an appropriate objective golden standard, or an ‘ideal’ ranking target, as a reference point. The definition of a golden standard ranking baseline allows for measuring the distance of computed rankings versus the agreed ranking reference point. We explained in the previous section that Infobox templates define a set of core properties that are common to a group of related articles. As an example, consider the Infobox template Planet\(^{14}\). This template defines properties such as discovery date, discoverer, category and so forth. Most Wikipedia articles that deal with planets, such as the article for Pluto\(^{15}\), attach this Infobox template as a callout. Infoboxes are not required nor prohibited for any Wikipedia article. Whether to include an Infobox, or which Infobox template to include, or even which parts of the Infobox template to use, is determined entirely by discussion and consensus among the editors of a Wikipedia article.

\(^{14}\) http://en.wikipedia.org/wiki/Template:Infobox_planet

\(^{15}\) http://en.wikipedia.org/wiki/Pluto
A key feature of Infobox templates that is specifically important for our research is that they explicitly define the display order of the features. Infobox templates for most Wikipedia articles have often been fine-tuned over the course of many years by an active and enthusiastic community of usually expert users. Typically, numerous iterations of collaborative editing by many users have resulted in optimized and consensus-based Infobox templates. Wikipedia templates are generally not modified by ‘anonymous’ Wikipedia article editors. Instead, contributors tend to seek opinions from other editors before embarking on a design of a new Infobox template, or the optimization of an existing one. New template prototypes (or changes) are usually proposed to the appropriate Wiki-Project as to obtain group consensus before deployment. Hence, a fair amount of thinking and design has often gone into the process of designing Infobox templates.

We posit for the above rationale that ranking data extracted from Infobox templates can be perceived as a ranking preference of domain specialists within their specific categories. In addition, we also postulate that this data set is usable as a ranking preference of ‘the masses’ given the intent of Inboxes within the Wikipedia context. We finally propose that ranking data extracted from Wikipedia Infobox templates is fit for the purpose of evaluating the ranking algorithms within the context of this research. A goal of the literature study is to validate that the given assumptions are correct and beyond a reasonable doubt. If this proofs to be valid, than the ranking data extracted from Infobox templates can act as reasonable source for ranking evaluation under the premise that we evaluate ranking within the given context.

There are several other aspects of Infobox templates that we further review as part of this research. One particular facet of concern is the fact that each language variant\textsuperscript{16} of Wikipedia can specify proprietary Infobox templates. These templates can vary in terms of content and ordering of properties. For example, consider the German and English Wikipedia articles for Berlin\textsuperscript{17}. Both articles utilize different (although closely related) Infobox templates, but also differ in the ranking of the contained properties. Ranking of properties in even closely related Infobox templates can thus differ across different languages. An Infobox template therefore represents one of many possible human ordering of the given category of data. As we know that Wikipedia language variants can have different rankings for the same Wikipedia articles we need to understand how this affects the usage for ranking evaluation purposes.

\textsuperscript{16} Each Wikipedia language can define a variant article and associated Infobox.
\textsuperscript{17} http://en.wikipedia.org/wiki/Berlin and http://de.wikipedia.org/wiki/Berlin
1.5 Research questions

The research goals that we defined in the previous section gives rise to several theoretical questions that we sought to answer in our study. The methods presented by Saunders et al [7] assisted in deriving the key research questions from the high level research goals. These theoretical research questions are important as sufficient conclusive answering brings insight into the mechanics of our study goal constituent’s parts, and aid in the road mapping towards reaching the research goals.

1.5.1 Research questions that relate to ranking algorithms in general

1. What does “property ranking” exactly entail (for an entity, such as an ontology class), and which methods and techniques are described in scientific literature with respect to ranking of properties?

2. Which potential ranking techniques may proof useful to further research?

3. Can we improve on existing ranking algorithms, or alternatively, specify effective new ranking algorithms?

1.5.2 Research questions that relate to the usage of Infobox ranking data

4. What is the validity of the ranking information extracted from Infobox templates for obtaining a target truth set to evaluate ranking algorithms in generic user interface applications?

5. How do Infobox template language variants influence the evaluation of ranking algorithms?

1.5.3 Research questions that relate to the evaluation of ranking algorithms

6. What metrics are useful to assess an ontology property ranking algorithm in generic user interface scenarios?
Chapter 2  Related and prior work

2.1 Introduction

We now present a review of related work that has been previously submitted to the scientific community within the field of property ranking. An important goal of the literature research is to identify any gaps within the existing academic works. These gaps are discussed later in the appropriate sections within this work. We start with a section that covers the overall literature research methodology which explaining the approach and execution with respect to identifying earlier work. The subsequent sections summarize existing relevant material in order to provide additional context and insight into the specific problem domain that this paper covers.

We first provide an overview of the different types of applicable ranking algorithms that have been described in earlier works. This is done by means of categorizing the various approaches and a grouping relevant works into related sections. The text then evolves by a critical review of relevant works for each approach. The insight that we gained provided a solid context for our research efforts. A final section concludes the review of related work with a gap analysis and summary of the key findings of the literature study.

2.2 Literature study methodology

In order to answer the theoretical questions from section 1.5 an extensive study of prior research work in related areas of computer science was executed. The overall outline is represented in Figure 9.

![Figure 9: literature study outline](image-url)
The search for relevant material has been performed mainly via search engines from Google (“Scholar”) and scientific paper publishers. A starting set of key search terms has been defined, per theoretical question, to drive the research discovery process. The set of search terms matured further as insights advanced. The search term set is outlined in the appendix (see section 6.2). The search focused on research work written in English in order to guarantee inclusion of high quality renowned work (as interesting domestic work is usually translated into English). Most of the reviewed papers cover a period between 2004 and 2012.

All material referenced in this literature study has been reviewed to the best of judgment to certify that the reference material is appropriate and credible, meaning:

- the research is scientific and peer reviewed (conference paper, journal article);
- the research has been cited in other work (hence, is “authoritative”);
- the research is publically available.

And for key referenced papers:

- the research has been refereed (i.e. peer reviewed);
- the research is published in renowned scientific journals or proceedings.

In addition, some research was done by reviewing information from the US patent database. However, unless such information has been published (i.e. meets the above criteria) no such work is incorporated in this text.

All incorporated literature and references has been managed during the research process via the Zotero reference management suite. This is specialized software that helps researchers manage key aspects of literature study. Most papers have been stored as searchable PDF in a central repository. Each paper has been annotated with additional metadata properties such as the bibliographical details, research notes, relevance and so forth.

The predominant focus in the first stage of the literature study identification was discovery of potentially relevant papers by means of keyword search. These papers were initially scan-read to determine the relevance, if any. Those papers that qualified as relevant were fully read during the second stage of the literature research. Each paper was categorized and augmented with relevant research notes, critiquing were appropriate. This step helped to relate concepts, models, methodologies and core theories into categorized viewpoints. We traced literature references in relevant papers to discover additional interesting related work. This process was reiterated until the point that no further new material work seemed to emerge. Although the volume of covered relevant material can never be fully complete, we feel confident that (at least, to the best of our knowledge) we covered a significant amount of the relevant related works.
Our study of related work was by intent broader than merely the application of ranking within the field of Semantic Web technology. Property ranking is a problem that arises within other contexts than Semantic Web applications. Other similar domains where related property ranking problems can occur are contexts such as object-oriented computing languages (class hierarchies with properties), tables in databases, XML schema complex types, RDF networks and so forth. Within the context of this review we use the generic container term entity to denote all such concepts. We define entity property ranking as the task of creating an ordered list of properties for a given entity, where the position of a particular property ("rank") reflects its specific importance, ideally with respect to the context of the use case. In various related works we find discussions of node properties, RDF properties or subject properties; all these terms are variant terminology for the same concept. In a similar fashion, in addition to ranking other researchers have coined synonyms such as ordering and prioritization. Other terms that are commonly used within this field are salience and prominence. These terms denote the relative importance of some property.

As mentioned in the early chapters, the lack of entity property ranking specifications pose a challenge for Semantic Web browser developers to present users with the most appropriate properties to the users; even though this is a very important aspect of the Semantic Web browsing user experience. A work by [3] exemplifies this: "Users do not want the raw information, but rather they need the information to be restructured and summarized. Restructuring and summarization cannot be done without a deep understanding of the user’s tasks". Entity summarization is a specific field within the Linked Data research community that researches the problem of ‘features’ ranking (with features denoting Linked Data property-value pairs). The goal of entity summarization is to identify those features that define the identity of a subject. Entity summarization deals with selecting features that “unambiguously identify an entity”, where entity property ranking “selects features that are most interesting to present to a user” [8]. Entity summarization, as a research topic, has its origins in the field of data summarization. Data summarization has been researched by different communities, and deals with computing compact representations from the original elements (i.e. databases or graph data). Some interesting work in this area is further discussed in the following sections.
2.3 Approaches to entity property ranking

The absence of directly available entity property ranking information has triggered a fair amount of research into the topic over the last decade in the Linked Data research community. Several of these publications have been studied in this review. A categorized outline of the various identified approaches to property ranking within the context of Semantic Web applications is presented in Table 1. The next sections cover these techniques in more detail with a review of the applicable literature per category.

<table>
<thead>
<tr>
<th>Ranking approach</th>
<th>Overall concept</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schema-driven.</strong></td>
<td>Ranking is deduced from overall structure.</td>
</tr>
<tr>
<td><strong>Triple-driven</strong></td>
<td>Data itself forms the basis for ranking</td>
</tr>
<tr>
<td>2. Property instantiation-based.</td>
<td>Counting occurrences of property instantiations.</td>
</tr>
<tr>
<td>3. Centrality-based.</td>
<td>Extraction of key concepts from RDF data.</td>
</tr>
<tr>
<td><strong>Metadata-driven.</strong></td>
<td>Utilization of external data.</td>
</tr>
<tr>
<td>1. Explicit rankings.</td>
<td>Proprietary metadata specifies importance.</td>
</tr>
<tr>
<td>2. Crowd sourcing-based.</td>
<td>Ranking data gathered from crowd sourcing.</td>
</tr>
<tr>
<td>4. NLP-based</td>
<td>Ranking derived from linguistic analysis.</td>
</tr>
</tbody>
</table>

Table 1: entity property ranking algorithms
The first category encompasses research that utilizes information from a governing schema such as an OWL-based ontology, an object class hierarchy, a XSL schema or a relational database schema. The related work in this category is potentially very relevant to our research. Unfortunately, earlier research with respect to ranking algorithms that utilize this type of information for property ranking purposes is thinly spread.

The “triple-driven” methodologies all base their entity property ranking algorithms on inspecting given RDF graphs. One of the key problems of these approaches is their computational cost as these algorithms tend to visit significant portions of the graph in order to establish ranking. As the main concern of this work does not deal with triple-driven property ranking, a limited selection of methods in this category is discussed in this review.

The approaches in the “metadata-driven” category all require some form of additional external specification to establish relative entity property importance.

2.3.1 Schema-driven approaches: relationship type-based.

As stated, we found very few works that utilize the information from a governing schema to drive property ranking in an interesting non futile manner. A key contribution in this area is an approach where ranking is based on the type of relationships. We found a relevant paper [9] in the field of database research which we discuss here as the method is generic and can also work in a Semantic Web context. In this work the authors describe their approach as follows: “The framework groups and ranks relationships based on their types, and then ranks relationship instances attached to each type. Most users will firstly be attracted by the relationship types, and then move down to the resources belonging to their interested type”.

A resource, be it a database record, of when applied to Linked Data, a subject, contains various relationship types. A relationship type is here a chain of linkage between entities. These relationship types (aka “paths”) are ranked by their relevance via so called “concept hops”. Concept hops indicate how many “concepts” a relationship type contains. As an example: if an entity A is linked via a property to an entity B, which then links to an entity C, then there are 3 Concept hops in that relationship type. This ranking is actually performed at the metadata level, rather than the instance level. A small number of hops in a relationship type indicate a more explicit relationship.
The authors postulate that (in general) people prefer more explicit relationships, and should thus be ranked higher. Although the work is specific for the domain of databases, the presented ideas can be applied to Semantic Web Data as well. One aspect to take into account is that the relationship types are defined via metadata annotations (in the reviewed work methodology). However, it is likely that relationship types can also be established via algorithms from graph theory. The authors do not present a comparison of their results with respect to a baseline, so the validity of their method is difficult to assess. The presented results are somewhat “subjective” in nature. The method is interesting concept and could be a valuable approach in the context of entity property ranking.

2.3.2 Triple-driven approaches: heuristics-based

A very interesting contribution [10] in the category of triple-driven approaches defines eleven heuristics that are used to compute an entity property ranking. The described ranking algorithm computes the ranking using simple rules that are evaluated as a given RDF graph is traversed. The underlying concept of applying heuristics to compute a ranking however could also fit well inside a purely schematic approach. The heuristics presented in the work rely partly on generic patterns; four out of the eleven heuristic rules are however specific to the DBpedia ontology [6]. These rules will not hold up in other (read: ‘generic’) Linked Data ontologies.

The authors conduct a quantitative evaluation in order to find out which heuristic combinations perform best. The results show that some heuristics, such as the Wikilink and Backlink based ones, provide high recall, while Frequency and “Same RDF type”-based heuristics yield high precision. Trials with blending of heuristics have showed that either precision or recall can be kept at a significant high level, but not both at the same time. As an interesting side note, the authors of this work state that summaries (and hence, also ranking) should be considered in a specific context, specific to the search task. Therefore, quantitative measures might not provide the right means to evaluate entity property rankings. This is supported by some of the other findings in this literature study: entity property ranking is subjective and is specific to a particular use case or task.
2.3.3 Triple-driven approaches: property instantiation counting

In a work that covers both finding and ranking Linked Data knowledge [11] properties are ranked by means of counting the numbers of times that a particular property is instantiated. The authors coin the term class-property bond to denote “rdfs:domain” relations between a property and a class. These “c-p bonds” are specified in ontologies in various ways, either via direct associations, or via class-inheritance. The method searches for the two-triple graph pattern (_x , rdf : type , class) ; (_x , property, _) in class instances. Ranking is done over the actual RDF sub-graph that instantiates c-p-bonds, rather than the defining ontology (as the latter does not fully show how well a c-p-bond is adopted in practice).

The authors do not-test their results against a baseline. Hence, evaluating the method is difficult for an outsider. A key problem with this approach is that counting methods will rank properties that are common as very important, even though these might possibly be totally irrelevant. As an example: longitudes and latitudes of cities may be common properties, but humans would rank names and countries of cities as more important properties (more identifying). Counting approaches will rank properties with identical frequency as equally important, but fall short as they do not incorporate the ‘human perspective’ in any way. A second pitfall is the computational complexity. Counting means scanning, and as the number of nodes and properties increases, so does the required scanning effort.

2.3.4 Triple-driven approaches: centrality-based

The concept of centrality is thoroughly researched in the context of Linked Data. One particular interesting research effort is described with respect to ontology summarization by [12]. A given RDF graph is sliced up into a set of sub-graphs, which are further denoted as RDF “sentences”. Sentences are linked based on the common nodes they share. This network is then considered as a separate graph from which centrality measures (e.g. PageRank) are calculated in order to rank properties. The authors state that “generally, a user may think that an RDF sentence about a property is salient if it links to many salient RDF sentences about classes; and similarly, an RDF sentence about a class is salient if it is linked from many salient RDF sentences about properties. It motivates us to analyze the hubness and authority of an RDF sentence. The salience of an RDF sentence is assessed by its hubness if its subject is a property; or the salience is assessed by its authority if its subject is a class."

In a related and newer work by the same authors more focus is given on the topic of entity summarization [13]. The authors present an improved algorithm that computes the ranking from relatedness as well as informativeness (a well-known information-theory concept). They do this by augmenting the centrality-based ranking algorithm with information found in nodes and edges from the original graph data.
The research from [14] is related work that further extends on the above approach. The authors state that features which are shared with its nearby \((k\text{-nearest})\) neighbor entities, are more relevant than features that are shared with entities that are not in the \(k\text{-nearest}\) range. The \(k\) range is computed from an external source (which the authors refer to as “usage” data). Although this is a combinatorial method (as this work actually employs external metadata, i.e. the “usage” data), the concept of centrality-based ranking is still very dominant.

There is a fair amount of other related work that takes the hub/centrality idea to rank RDF data. Some related research in this area is for example found in “TripleRank” [15], which is in turn inspired from earlier work within in the context of database search by [16].

An issue with the centrality approach is that hub-based algorithms typically tend to favor related (“shared”) properties. Not all properties that a human might rank as ‘important’ may satisfy this condition, and may thus be missed.

### 2.3.5 Metadata-driven approaches: explicit ranking

Various authors have proposed the usage of additional metadata to solve the problem of entity property ranking. The authors of Haystack [17] state for example: “We argue that to support appropriate presentation of Semantic Web information to end users, it will be necessary to define an ontology for describing presentation knowledge, such as which are the important properties of a class.”. Next the authors present an ontology for this purpose (the View Ontology Web Language) in an effort to bridge the gap between a user’s display needs and the underlying data model.

In a work that describes methods for mapping ontologies to portals [18] the authors write “Although we already discussed the visual representation of object properties between classes (Visual patterns), none of the proposed solutions (e.g., tabbed interface) provide information about ordering of the visualization.”. In order to mitigate this shortcoming, the authors defined an additional ontology that contains information with respect to the display order of the class properties. Here too external metadata is utilized to solve the problem of entity property ranking.

Another example of utilizing external metadata with explicit ranking information is given in [19]. The authors define additional “subjectivity/objectivity weight modifiers” to modify the subject/object of the property in order to influence ranking.
As a last example of this category Fresnel should be discussed. Fresnel is an external specification of presentation logic for ontology specifications. An interesting contribution that utilizes Fresnel metadata is presented in [20]. They contribute the idea of selecting an appropriate Fresnel lens from a repository of lenses dynamically from on a given RDF graph.

If anything, these methods demonstrate clearly that a lack of entity property ranking information is a problem for Semantic Web browsers. Even though explicit ranking is not an algorithmic method, there is still a very good argument for this approach, and that is that the resultant entity property ranking can be of very high quality.

2.3.6 Metadata-driven approaches: crowd sourcing-based

The research in this category uses crowd sourced data to improve entity property rankings. Most of the described techniques utilize a form of an online quiz-like game where given RDF data is presented as questions using either the subject, the predicate or the predicate of a triple (depending on the game design). The quiz response data is then analyzed, and this information is in turn used to improve ranking of RDF data sets. These techniques are detailed in papers from [21] and [8].

Some important observations with respect to entity property ranking in general are found in two works within this category by [22] and [23]. Properties for seemingly closely related classes (i.e. Politician and Actor or Company and Organization) are actually ranked very differently by humans. In fact, even within the same class, for example within the class Person, entity property ranking is perceived differently with respect to the actual instance (i.e. President Obama versus John Doe).

Furthermore, some of the results seem to indicate that properties that score high for identity by no means also score high in terms of human-based ranking. For example, consider a property such as the date of birth for a person. Humans will generally not rank birth date as a very important property. This problem is explained quite well by the authors of [13]: “some features were ranked high because of their high “informative” and notable relatedness, e.g. features that stand for the longitude and latitude of a city, they were not preferred by most participants because the information they carry were deemed too “domain-specific” to be exploited. That is, these features are highly informative for domain experts that can deal with this particular kind of knowledge, but are not as valuable when presented to average users”.

Hence, (perceived) importance is difficult topic to asses, and is in fact very use case specific. There is also strong evidence that cultural differences are an important factor to consider as well.
2.3.7 Metadata-driven approaches: example-based

An interesting concept is presented in a very recent work by [24] where a given “sample” web page seeds automated interface generation for a set of given RDF data. The essence of the approach is that the DOM of the given web page is analyzed to detect visualization patterns (such as tables or lists). These structures are then analyzed and matched (via labels, property names and so forth) against the given RDF data set. The authors then generate Fresnel lenses from this information as to present the RDF data similar to the example web page. As an approach it is indeed quite novel. An obvious ‘flaw’ being that it cannot deal with properties that are not present in the given example web page.

2.3.8 Metadata-driven approaches: NLP-based

NLP-based approaches may proof very valuable to rank entity properties. For example, there is a wealth of lexical data from sources such as WordNet and Framenet that may be utilized to rank entity properties. WordNet is a large lexical database (of English) that includes nouns, verbs, adjectives and adverbs grouped into sets of cognitive synonyms, each group expressing a distinct concept. Framenet is a lexical database (of English) that contains annotated examples of how words are used in actual texts. Corpus-based analysis techniques could be very useful to rank entity properties.

A fair amount of research within the field of Linked Data exists that employ NLP techniques in various ontology oriented use cases (for example in ontology search and ontology similarity matching methods). However, no specific works could be discovered that use NLP approaches (or lexical data) with the direct intention to rank entity properties. Some hints are given in a W3C draft [25] towards the idea of utilizing WordNet data to rank / categorize RDF concepts: “A related but distinct activity would be to describe the use of WordNet as a basis for RDF/OWL class and/or property hierarchy”. The draft then continues “WordNet's noun term (hypernym) hierarchy captures ‘an X is a kind of Y' relationships between English category terms based on conventional usage”. In addition to the above mentioned “X-is-a-kind-of-Y” relationship, WordNet offers many other interesting relationships that may be utilized to establish entity property ranking. For example, via taxonomic and hierarchical relations such as “part of”, “madeWith”, “Is-a”, “Includes”, “has-part” and so forth.
Another interesting pointer with respect to the possibilities towards ranking entity properties using NLP techniques is given in a work by [26]. This research is inspired on earlier work by [27] in the domain of ontology evaluation. The authors employ NLP techniques to find and rank ontologies from a set of given search keywords. The approach uses data from WordNet to improve the search terms. Next, the first 100 results pages from Google are used as an input to establish a corpus of domain related information. The top 50 terms in the corpus are then employed to score ontology relevance. In the domain of entity property ranking, instead of using a given set of keywords, properties from a given ontology could be used as a starting set of search terms. WordNet and Framenet data may help to improve this set of search terms. Next, rather than just parsing the result pages, the actual webpages could be parsed to obtain a higher corpus quality. This higher quality corpus may then act as the basis for ranking entity properties (using term frequency measures). More ideas in this area (corpus-based entity property ranking) may be derived from [28]. This paper discusses the ranking of ontology concepts. The metrics defined in this context may be applicable for entity property ranking as well.

2.4 Conclusions of related work and gap analysis

The review of known literature has shown that various approaches with respect to ranking entity properties have been described in earlier works. The research study has shown that the field of entity property ranking is diverse. This is partly demonstrated by the variety in approaches found that deal with the entity property ranking problem. It indicates a lively research field. One gap that we identified involves the lack of papers that target the problem of ontology class property ranking through terminological-based techniques\(^\text{18}\). Results from the literature study suggest that such methods may be applicable with some degree of success within the domain of ontology class property ranking.

Various studies have demonstrated that optimal ranking requires knowledge about the use-case and user context. There is a significant amount of evidence that suggests that entity property ranking is subjective, as it is “context sensitive” and specific with respect to the use case scenario, and the type of user participating in a use case (expert versus non expert, locale, language and so forth). A domain expert participating in the same use case as an inexpert user may show different preferences with respect to ranking.

\(^{18}\) The study of a system of terms belonging or peculiar to a science, art, or specialized subject.
Other aspects, such as the native language and nationality of a user group, also influence ranking. Depending on the user type and use case at hand the ideal property ranking for a given entity may be different. We therefore conclude that property ranking is indeed subjective and context specific with respect to the usage scenario and type of user participating in that use case. Hence, it is not possible to create an “optimal” property ranking algorithm that will be fit for all goals and purposes. It is important that this understanding is fed back into our evaluation model. When we use a baseline to measure the distance of a computed ranking with respect to that reference point, and when we use that distance to determine the effectiveness of the algorithm, then it is important that the baseline must be both applicable and well understood, as the baseline itself will be context specific. Judgments made with respect to that baseline are valid only with respect to that same context.

Last, but not least, the study has shown that the fields of entity summarization and entity identification at least partly overlaps with the field of generic entity property ranking, and specifically property ranking for ontology classes.

The research questions have been sufficiently answered through the study of relevant work. The key findings are summarized hereunder.

What does “property ranking” exactly entail (for an entity, such as an ontology class), and which methods and techniques are described in scientific literature with respect to ranking of properties?

The study has clearly shown how the problem of (entity) property ranking has been addressed in prior relevant studies. The different approaches have been evaluated and strengths and weaknesses have been discussed in this review. The study has also yielded insights with respect to what ranking is about. Linkage to related fields of research has been discussed throughout this review.

Which potential ranking techniques may proof useful to further research?

There are gaps in the available literature with respect to utilizing NLP and terminological techniques to rank entity properties in the context of ontology classes. Related works suggest that we could approach the ranking of entity properties through term frequency metrics extracted from a relevant corpus. Such a corpus may be constructed using search terms that are obtained from a given ontology, possibly augmented by WordNet or Framenet information. This is an interesting and novel approach that is further researched in this work. Studies from Brewser et all [27], Jones & Alani [26] and Rospocher et all [28] have yielded strong indications that terminological analysis may be effective to rank the properties of an ontology in the context of generic use cases. Although these works do not directly research the application of ontology property ranking, the methods and results presented suggest that terminological analysis might be an effective approach for ranking properties in ontology classes.
Can we improve on existing ranking algorithms, or alternatively, specify effective new ranking algorithms?

The heuristics-based approach described in this review is a technique that is useful and worthy of further review and development. In particular, we aim to bring the application from an instance data-driven approach to a pure ontological analytical approach. We research this technique further in our research in section 4.6. A search to discover works that specifically target the problem of ontology property ranking through terminological analysis has not yielded any results, which suggests a gap in the literature. This has inspired further study to understand if terminological analysis-based ranking may outperform a naïve ranking algorithm such as an alphabetical ranking of properties. The expected outcome is that this will be the case, which is why some of our experiments will take this approach. We describe NLP-based ranking approaches in sections 4.2, 4.3, 4.4 and 4.5.

What is the validity of the ranking information extracted from Infobox templates for obtaining a target truth set to evaluate ranking algorithms in generic user interface applications?

The assessment of ranking algorithms is extremely complicated if we don’t use some sort of standardized baseline. Research of prior work revealed that the effectiveness of the various ranking algorithms is difficult to compare directly. The study of related work showed clearly that the relative effectiveness of the various entity property ranking algorithms cannot be established. Researchers consistently use different methodologies and target truth sets to evaluate their methods. A major advantage of a standardized baseline is that it enables direct comparison of ranking algorithms from different research strains.

A key assumption in our research is that Infobox templates typically list properties in order of perceived importance, and that (typically) this ranking should be valid for generic reviewing purposes. An Infobox design is the result of a long term collaborative design effort by a large community of users. As such, we feel confident that they can represent a baseline that is appropriate for generic casual data observers. Although the ranking data extracted from Infoboxes may not be ‘perfect’, the data is usable as a valid standard for evaluating entity property ranking research if we clearly indicate the specific context.
How do Infobox template language variants influence the evaluation of ranking algorithms?

Infobox template differences between languages, as well as various other related aspects are researched in more detail by Rinset et al [29]. Even though Infobox templates for the same category of data may differ we did not find any material that prevents the usage of ranking data extracted from English Infobox templates as a ground truth set for the purpose of evaluating ranking algorithms. That is, within the scope of the English language, meaning specifically, we see no arguments why the English ranking should not be used to evaluate rankings of English properties. As argued, Infobox templates have typically matured over a longer periods through various revisions executed by a community of topic enthusiasts. We could not find any principle reasons in the literature to reject this data set as a golden standard for evaluation purposes within this study. The Infobox template rankings have emerged as the result of a consensus-based process involving many contributors within the context of an immensely popular and widely accessible application (Wikipedia). We have not found any reasonable or credible arguments within the literature that a different source of ranking data will have significantly better intrinsic properties for evaluation purposes.

What metrics are useful to assess an ontology property ranking algorithm in generic user interface scenarios?

The literature describes two key metric categories to measure the delta between two alternative rankings (in our study: model rankings versus the ground truth set rankings):

(1) *Rank correlation metrics*: These metrics measure the amount of agreement between two rankings.

(2) *Classification metrics*: These metrics are commonly used in Information Retrieval (IR) applications and are typically based on an understanding (and measure) of relevance.\(^{19}\)

---

\(^{19}\) Relevance denotes how well retrieved data meets the information need of the user. Relevance may include concerns such as timeliness, authority or novelty of the result.
A rank correlation coefficient is a measure of the degree of similarity between two rankings. The evaluation model in this study is concerned only with measuring conjoint\(^{20}\) rankings. The literature describes many similarity measures for conjoint rankings. In fact, over thirty measures are compared and described in works by Tarsitano [30] and Webber et al [31]. Each method takes a different perspective on the association between two permutations. Important factors as to which metric works best need to consider the context in which association analysis is performed, the properties of the items to be ranked and the purpose of the study.

In his comparative study, Tarsitano states that it is unlikely that any single coefficient metric can cope with, or even detect, the many variations in nonlinear relationships between all types of rankings. The Kendall’s Tau (τ) correlation coefficient [32] however remains one of the most popular rank correlation metrics for testing nonlinear correlations. There are specific variants of Kendall’s τ that deal with specific use cases (see [30], [33] and [34]) but standard Kendall’s τ is sufficient in the specific case of this evaluation model (see also the study from Lapata [35]). The Kendall rank coefficient is useable as a test statistic in a statistical hypothesis test. An important property of τ is its limiting probability distribution. Under the null hypothesis of random uniformly distributed rankings an experimenter can infer the significance of the rank correlation coefficient by using the Normal probability distribution tables as is explained by Melucci [33]. This type of testing can demonstrate whether a ranking algorithm produces results that are significantly better than a random ranking (or a semi-random ranking such as alphabetical property ordering).

Some carefulness is necessary when interpreting Kendall’s τ, especially in meta-evaluations. The intuition Kendall’s τ offers may be misleading in specific scenarios. A study from Carterette [36] indicates that a "high" τ does not always necessarily indicate a good ranking, and a "low" τ is far from a guarantee of a bad ranking.

In addition to Kendall’s τ we found various publications that utilized Spearman’s Rho (ρ) and a normalized variant of Spearman’s Footrule distance as ranking metrics.

The research into the application of Information Retrieval related metrics revealed that four specific IR metrics are useful for this study, namely precision, recall, F-score and average precision. Precision is essentially the amount of noise that a model or algorithm generates while recall is the exactness of the result. The IR measures can determine which of a given two given systems outperforms the other when they are compared with respect to some input dataset. The metrics are particularly useful in the context of entity summarization and entity identification applications. In these use cases, precise ranking is of lesser importance, as the predominant goal is to establish the top-\(n\) most relevant properties.

\(^{20}\) Conjoint rankings consider lists where both lists consist of the same items
Chapter 3  Evaluation model

3.1 Introduction

The intent of the evaluation model in this study is to gain a comprehensive understanding of the performance of the studied rankings algorithms. A key objective is to understand if the ranking algorithms can approximate a human designed ranking. In addition, we want to compare the ranking algorithms directly with a naïve ranking algorithm such as alphabetic ranking. Our ranking algorithms are evaluated using a test data set that was introduced in section 1.3. The exact construction is explained in more detail later in this chapter. We have proposed and defended the usage of ranking data from Wikipedia Infobox as a ground truth set for evaluating our ranking algorithms. An important hypothesis in our model is that human nature is such that the most dominant properties will typically be listed before the less relevant or more obscure facts in the majority of the Infobox templates. A key underpinning assumption of the evaluation model is that the properties in an Infobox template are commonly ranked (and grouped) using some sensible intelligent ranking rationale with respect to that particular topic. Before we discuss the full details of our evaluation model we first introduce a high level overview of the involved data structures in Figure 10 (note: some of these structures will be discussed a little later in this chapter).

Wikipedia
- Defines Infobox templates.
- Contains articles that (may) contain one or more Infobox instances.

DBpedia
- Wikipedia infobox templates are mapped to DBpedia ontology classes.
- Extracts property data from Wikipedia article Infobox instances; properties are converted and stored in the DBpedia ontology as RDF triples.

WordNet
- Defines terminology, synonyms, word relationships.

Alchemy
- Computes keywords, categories, concepts, taxonomies from a given input corpus.

Figure 10: key data structures involved in the construction of the test data set
We also introduce a schematic high level overview in Figure 11 of how the research project interacts with the data sets listed in Figure 10.

DBpedia defines their ontology as “a shallow, cross-domain ontology, which has been manually created based on the most commonly used infoboxes within Wikipedia.” The ontology currently covers 529 classes which form a subsumption hierarchy and are described by 2,333 different properties. The ontology data instances are extracted by the DBpedia project from multiple information sources. One particular important data source for DBpedia is the set of Infobox structures attached to Wikipedia articles.

The DBpedia extractors parse the Infobox structures from the Wikipedia pages and transform the property-value pairs into Semantic Web RDF triples. The extracted data is precisely mapped to the correct ontology class. This process is explained in more detail in the next paragraph. Important to this research is the fact that DBpedia publishes the mapping metadata so that we can trace back how DBpedia has sourced the various ontology class properties. The classes that source multiple properties from Wikipedia Infobox templates are usable in our test data set as we can retrieve the rank order information from the linked templates. It is even possible to capture multiple property ranking baselines since DBpedia maps various Wikipedia language variants, as will be explained in section 3.4.

---

21 [http://wiki.DBpedia.org/Ontology](http://wiki.DBpedia.org/Ontology)
As mentioned in section 1.3 we choose the DBpedia ontology as the basis for our test data set. DBpedia extracts information from Wikipedia Infobox structures and publishes this data set as Linked Data triples. We already mentioned that this process is important to our study, so how does it work? With each new release of DBpedia the extractor component [37] traverses over all Wikipedia pages. This module uses manual mapping files that instruct how it must process Infobox instances that it finds.

Consider as an example the DBpedia ontology class Planet\textsuperscript{22}. The DBpedia project maintains a list of class-to-template mappings, and one particular mapping\textsuperscript{23} associates the Infobox template Planet\textsuperscript{24} with the DBpedia ontology class Planet. Whenever this template is found on a Wikipedia page, for example such as on the articles for Mars\textsuperscript{25} and Venus\textsuperscript{26}, it can map the information in the Infobox instance of that article onto the DBpedia ontology class Planet. This structure is high level depicted in Figure 12. Our research utilizes these DBpedia mappings to derive the test data set for the purpose of ranking performance analysis.

![DBpedia maps ontology classes to Wikipedia Infobox templates](image)

Figure 12: DBpedia maps ontology classes to Wikipedia Infobox templates

Our research uses the ranking information embedded in Wikipedia Infobox templates to establish a ground truth to evaluate ranking algorithms. That baseline can then be used to compute a series of performance metrics that measure the effectiveness of ranking algorithms. The metrics are used to quantify how well a ranking algorithm may support automated view generation to target casual data observers in Semantic Web browser applications. These metrics are explained in more detail in section 3.5.

\textsuperscript{22} http://DBpedia.org/ontology/Planet
\textsuperscript{23} http://mappings,DBpedia.org/index.php/Mapping_en:Infobox_planet
\textsuperscript{24} http://en.wikipedia.org/wiki/Template:Infobox_planet
\textsuperscript{25} http://en.wikipedia.org/wiki/Mars
\textsuperscript{26} http://en.wikipedia.org/wiki/Venus
3.2 Mapping DBpedia onto Wikipedia templates

We now explain in full detail how ranking information for DBpedia ontology class properties can be obtained from related Infobox templates. Consider the DBpedia mapping data structure\(^{27}\) shown in Figure 13 that was extracted from the English DBpedia mapping data set\(^{28}\). The mapping relates the ontology class Brain with the Wikipedia Infobox template Infobox_brain\(^{29}\) that is shown in Figure 14. Note that for completeness sake we have also included an abstract of the DBpedia ontology class Brain in Figure 15.

```
{{TemplateMapping
| mapToClass = Brain
| mappings =
  ...
  {{PropertyMapping | templateProperty = Name | ontologyProperty = foaf:name }}
  {{PropertyMapping | templateProperty = GrayPage | ontologyProperty = grayPage }}
  {{PropertyMapping | templateProperty = Artery | ontologyProperty = artery }}
  {{PropertyMapping | templateProperty = Vein | ontologyProperty = vein }}
  {{PropertyMapping | templateProperty = MeshName | ontologyProperty = meshName }}
  ...
}}
```

Figure 13: DBpedia mapping (simplified) for Infobox template "Infobox_brain"

```
{{infobox
...
| label14 = System
| data14 = {{{system}}}
| label15 = Components
| data15 = {{{components}}}
| label16 = <div style="background:pink; width:100%">Artery</div>
| data16 = {{{artery}}}
| label17 = <div style="background:lightblue; width:100%">Vein</div>
| data17 = {{{vein}}}
| ...
}}
```

Figure 14: simplified extract of Infobox template "Infobox_brain"

\(^{27}\) http://mappings.DBpedia.org/index.php/Mapping_en:Infobox_brain
\(^{28}\) http://mappings.DBpedia.org/index.php/Mapping_en
\(^{29}\) http://en.wikipedia.org/wiki/Template:Infobox_brain
Essentially, the DBpedia mapping data structures materializes a linkage between DBpedia ontology classes and their related Infobox templates. It is this very relationship that allows us to extract the DBpedia ontology class property rank order for the application of our ranking algorithm evaluation methodology. Please note that the ranking order in Figure 14 is established through a top-down parse order.

<table>
<thead>
<tr>
<th>DBpedia ontology, class Brain</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdf:type = DBpedia-owl:Brain</td>
</tr>
<tr>
<td>Label (English) = “brain”</td>
</tr>
<tr>
<td>Super classes = AnatomicalStructure</td>
</tr>
</tbody>
</table>

Key properties for class Brain (name, label, domain, range)

- artery, “artery”, AnatomicalStructure, Artery
- brainInfoNumber, “brain info number”, Brain, xsd:string
- brainInfoType, “brain info type”, Brain, xsd:string
- branchFrom, “branch from”, AnatomicalStructure, AnatomicalStructure
- branchTo, “branch to”, AnatomicalStructure, AnatomicalStructure
- component, “component”, Brain, AnatomicalStructure
- nerve, “nerve”, AnatomicalStructure, Nerve
- precursor, “precursor”, AnatomicalStructure, Embryology
- vein, “vein”, AnatomicalStructure, Vein

Figure 15: summary of the DBpedia ontology class “Brain”
3.3 Test data set construction (English)

The test data is sampled from a wide range of DBpedia ontology classes to guarantee coverage of a wide span of topics. A total of 417 DBpedia ontology classes were reviewed for potential inclusion in our test data set. In order to calculate meaningful and statistically relevant evaluation metrics we only considered classes where a minimum of eight properties have been successfully mapped. This threshold effectively halves the input set, but it also decreases the chances for false positive ranking results. The final cleansed data set compromised of 218 ontology classes. We enriched and extended the test data set with related terminology data so that we didn’t had to re-compute commonly used support data during our experiments over and over again. Although principally merely a convenience step, it also ensured that the enrichment data itself was stable. The entire high level test data preparation process is illustrated in Figure 16.

We largely explained the first two steps of Figure 16 already in section 3.2. Essentially, each DBpedia mapping links an Infobox template with a DBpedia ontology class. This mapping allows for the creation of a series of tuples \((C, I)\) where each included ontology class \(C\) has a 1:1 relationship with an (English) Infobox template \(I\). We parse the Infobox template \(I\) and capture the contained property ranking. We then discard all properties that fail to interlink. The high level mapping process is schematically depicted in Table 2. Here we depict the fact that an Infobox template may contain properties that are not mapped in DBpedia, and furthermore, that the linked ontology class may also contain properties that do not originate from Wikipedia. The final mapping only consists of those properties that could be successfully identified in both sets (completing the third step of Figure 16).

<table>
<thead>
<tr>
<th>Infobox template</th>
<th>Ontology class</th>
<th>Mapped ontology class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property D</td>
<td>Property A (not mapped)</td>
<td>Property D (@ rank 1)</td>
</tr>
<tr>
<td>Unmapped property P</td>
<td>Property B</td>
<td>Property E (@ rank 2)</td>
</tr>
<tr>
<td>Property E</td>
<td>Property C (not mapped)</td>
<td>Property B (@ rank 3)</td>
</tr>
<tr>
<td>Unmapped property Q</td>
<td>Property D</td>
<td>...</td>
</tr>
<tr>
<td>Property B</td>
<td>Property E</td>
<td>Property Z (@ rank n)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Property Z</td>
<td>Property Z</td>
<td>Unmapped properties discarded</td>
</tr>
</tbody>
</table>

Table 2: sample mapping of Infobox properties to ontology class properties
The actual rank capture process logic is complex since the parser has to deal with various Wikipedia legacy Infobox formats and a complicated multifaceted Wikipedia template data structure. We used the Sweble\textsuperscript{30} Java library to deal with the core parsing of Wiki mark-up (‘Wikitext’)\textsuperscript{31}. Sweble produces an Abstract Syntax Tree\textsuperscript{32} for an input Wikitext consisting of text nodes, redirect nodes, transclusion nodes and tag extension nodes. Although Sweble significantly eased the task of navigating the templates we still spend a considerable amount of time and effort in tuning the parsers as to obtain the most accurate mapping data possible within the scope of our research. We had to build two independent AST processors, one for parsing Wikipedia Infobox templates, and one for parsing the DBpedia mapping templates.

The Infobox template parser can process regular Infobox templates (consisting of the now standard header/label/data format) and various legacy (i.e. vCard\textsuperscript{33}) formats. The mapping template parser has full support for processing template-mapping and propertymapping constructs, and limited support for dealing with conditional mapping tags. There is also support for the processing of if and switch templates at a structural level, by which we mean that we do not execute full tag expansion which implies that Wikitext magic words and tag extensions are processed ‘as is’. We extensively tested the Infobox template parser on some of the most complicated Infobox templates. Although there are a few limits as to the completeness of our parsing process the resultant data set is sufficient rich for our needs and purposes.

The final stage in our test data preparation process (as depicted in Figure 16) involves an enrichment step where we extend our extracted data with various NLP data structures that we used commonly in our experiments. We initially obtained additional these data structures during specific experiments, but this slowed down the experiment run time so we pre-computed most of our commonly used structures instead. It also ensured that we used exactly the same data structures during the course of our experiments. As an example, the Alchemy API that we used had multiple version upgrades during our study. If we would not have pre-computed these data structures, then experiment data that spawned multiple versions of the Alchemy API might not have been directly comparable for that reason.

\textsuperscript{30} http://en.wikipedia.org/wiki/Sweble
\textsuperscript{31} http://en.wikipedia.org/wiki/Wiki_markup
\textsuperscript{32} A tree representation of the abstract syntactic structure of the Wikitext template code; each node in the tree represents a construct within the code structure
\textsuperscript{33} http://www.w3.org/TR/vcard-rdf/
The first type of enrichment data stems from WordNet\textsuperscript{34} [38]. WordNet is a large lexical database of English. The database contains nouns, verbs, adjectives and adverbs grouped into sets of cognitive synonyms (so called “synsets” or “senses”). Each such group is expressing a distinct concept. Various semantic relationships between two WordNet senses may exist; the most frequently encoded relation among senses is the super-subordinate relation, also called hypernymy, hyponymy or ISA relation. Each ontology class property in our test data set was extended with relevant WordNet synsets that were deduced based on the property label.

In addition, we sourced NLP data from the NLP online service Alchemy API\textsuperscript{35}. Alchemy API is a web-based text analysis service that is backed by a linguistic processing engine based on statistical and symbolic NLP techniques. There are various comparable NLP engines such as Open Calais\textsuperscript{36} and TextWise\textsuperscript{37}, but we preferred Alchemy API for its ease of use, feature completeness and academic licensing. For each ontology class property we queried Alchemy API to source related keywords, categories, concepts and taxonomy information. Note that Alchemy API offers these services in several languages, but our evaluation context, and hence, enrichment, was predominantly focused towards the English language. We used the Alchemy API keyword extraction service to source the most dominant terms from a class property label. Alchemy API may find multiple relevant keywords, and each keyword is scored through a relevance score that Alchemy API computes using statistical analysis. We stored all extracted keywords for a specific property inclusive of the relevance scores.

In addition, we utilized Alchemy API to compute a high level category for class properties from the property label\textsuperscript{38}. Alchemy API can assign one of the following groups for a given textual phrase it is asked to analyse: Arts & Entertainment, Business, Computers & Internet, Culture & Politics, Gaming, Health, Law & Crime, Religion, Recreation, Science & Technology, Sports and Weather. Alchemy API can compute concepts from phrases and even identify concepts that may not necessarily be directly referenced in a text. For example, if an article mentions “CERN” and “Higgs boson” then Alchemy API may compute “Large Hadron Collider” as a related concept, without an explicit mention of the term in the article.

\textsuperscript{34} http://WordNet.princeton.edu/
\textsuperscript{35} http://www.alchemyapi.com/
\textsuperscript{36} http://www.opencalais.com/
\textsuperscript{37} http://www.textwise.com/
\textsuperscript{38} We do not compute keywords from the property comments as this added a high amount of keyword ‘noise’.
Unfortunately, Alchemy API typically requires more textual context than the class property labels and comments can offer to enable the accurate computation of relevant concepts. Alchemy API could compute a concept for seven out of 39 properties in the example of the DBpedia Person class. We found that the assignment of categories suffered from the same issue. In addition, the assigned concepts sometimes had flaws typically seen with the statistical approaches associated with natural language processing, reflecting that this computational approach to language has its limitations.

Finally, the taxonomy mapping service from Alchemy API was used to link properties to its rich bespoke taxonomy. In our example of the Person class we could assign a taxonomy entry point for nearly all properties based on their labels and associated comments (36 out of 39). The property ‘birth place’ for example linked to ‘/family and parenting/babies and toddlers’ (with confidence 0.745602), ‘/family and parenting/children’ (with confidence 0.427014) and finally ‘/family and parenting/motherhood/pregnancy’ at a confidence level of 0.372944.
3.4 Test data construction (multi-lingual experiment validations)

The multi-language experiments compared the influence of alternative human property rankings and the performance consistency of the top ranking algorithms. Typically one Infobox template is defined for a particular language per topic category. However, each Wikipedia language variant tends to localize Infobox templates (see also Table 3). These variants have different degrees of similarity with respect to the comparable English Infobox template. We can map a DBpedia ontology class $C$ both to an Infobox template $I_L$ for language $L$ and to an Infobox template $I_M$ for language $M$. Each such mapping yields a specific tuple $(C, I_L)$ where $I_L$ is an Infobox template for one of the mapped languages. These mappings can thus connect two related Infobox templates $I_L$ and $I_M$ (via $C$). This connection allows us to examine property rankings across languages.

<table>
<thead>
<tr>
<th>Template for topic $I_L$ (i.e. English)</th>
<th>Template for topic $I_M$ (i.e. French)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property D (@ rank 1)</td>
<td>Property B (@ rank 1)</td>
</tr>
<tr>
<td>Property E (@ rank 2)</td>
<td>Property D (@ rank 2)</td>
</tr>
<tr>
<td>Property B (@ rank 3)</td>
<td>Property P (@ rank 3)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Property Z (@ rank $n$)</td>
<td>Property Z (@ rank $n$)</td>
</tr>
</tbody>
</table>

Table 3: Infobox templates for the same topic may rank properties differently

We extracted several other language mappings from DBpedia in addition to English to understand how the top ranking algorithms (for English ranking data) would perform with respect to non-English ranking data. We created intersections of the English mappings with the non-English mappings. For example if class $C$ is mapped in both English and French then the intersection English-French will contains class $C$. The numbers are listed in Table 4.

<table>
<thead>
<tr>
<th>Language</th>
<th>Total collected</th>
<th>Intersection with English</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dutch</td>
<td>545</td>
<td>104</td>
</tr>
<tr>
<td>German</td>
<td>353</td>
<td>85</td>
</tr>
<tr>
<td>Portuguese</td>
<td>323</td>
<td>81</td>
</tr>
<tr>
<td>French</td>
<td>328</td>
<td>73</td>
</tr>
<tr>
<td>Spanish</td>
<td>221</td>
<td>70</td>
</tr>
<tr>
<td>Turkish</td>
<td>178</td>
<td>54</td>
</tr>
<tr>
<td>Polish</td>
<td>134</td>
<td>49</td>
</tr>
</tbody>
</table>

Table 4: multi-lingual data sets
Note that due to various reasons only a much smaller fraction of these numbers is actually usable, typically as little as half of the collected data, but sometimes even much more.

To enable direct comparisons between ranking variants an amount of further data cleansing was required on the intersection test data sets, as neither the DBpedia mappings, nor the Infobox templates are complete with respect to the mapped properties across all the languages in scope. An English mapping $E$ may map property $A$ for ontology class $C$, where the French mapping $F$ may not map property $A$ at all for $C$. Furthermore, Wikipedia Infobox editors may have removed a property from a template, although DBpedia may still actually map the property. Hence, crosschecks and clean-ups are a vital pre-processing step. This is explained in more detail in Table 5.

<table>
<thead>
<tr>
<th>English ranking for $C$</th>
<th>French ranking for $C$</th>
<th>Averaged ranking for $C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property D (@ rank 1)</td>
<td>Property B (@ rank 1)</td>
<td>Property D (@ rank 1)</td>
</tr>
<tr>
<td>Property E (@ rank 2)</td>
<td>Property D (@ rank 2)</td>
<td>Property B (@ rank 2)</td>
</tr>
<tr>
<td>Property B (@ rank 3)</td>
<td>Property P (@ rank 3)</td>
<td>Property P (@ rank 3)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Table 5: available ranking variants for a given DBpedia class
3.5 Evaluating ranking algorithms

All conducted experiments were reviewed using a bespoke experiment evaluation framework that we specifically developed for this research. The total code base, not including libraries, consisted of over 13,000 lines of code. This framework enabled repeated experiment execution and in-depth result observation, which assisted us in the calibration of the ranking algorithms under research. The experimentation framework computed all Key Performance Indicators in a fully automated fashion which eased the evaluation of ranking algorithms. We thoroughly segregated the metric computational logic from the experiment logic by means of decoupling the KPI computation code and ranking algorithm computation code in the framework (using a Chinese wall implemented through software).

We now explain how a typical experiment cycle was conducted. For each ranking algorithm under research we iterated over the data set and computed the rank order for each mapped class in scope using the algorithm logic. Each computed rank order is then evaluated using a set of evaluation metrics that judge the ranking against the English rank order baseline (as extracted from the English Infobox templates).

The described process yields a set of comparative metrics (per mapped class) for each ranking algorithm under review. Note here that the metrics computed for class A may not be compared directly to the metrics for class B, since A and B are in no way related (fully independent samples). However, we may compute the arithmetic means per metric for the entire data set of mapped classes. The average KPI values may then be compared between two algorithms to understand the relative performance of two algorithms. If one algorithm seems to outperform another, a statistical paired sample t-test is performed to validate if the differences are significant. This validation process is detailed further in paragraph 3.9. The next section will first detail the different metrics that we have computed in our research.

3.6 Rank correlation metrics

We have selected three key rank correlation measurements that allow for objective measurements: the basic Kendall’s Tau (τ) metric, the Spearman Rho (ρ) metric and a normalized version of Spearman’s Footrule. The following paragraphs outline these metrics in more detail.

39 A barrier that prevents the exchange of information that could cause conflicts of interest
3.6.1 Kendall’s Tau (τ) metric

An intuitive explanation of Kendall’s τ metric is that it measures the number of exchanges of two items necessary to transform the observed ranking into the reference ranking. The count of exchanges in Kendall τ is similar to that in the bubble-sort algorithm. The more exchanges required to transform the observed ranking to the reference ranking, the smaller the correlation between the two rankings. A more formal description of Kendall’s τ is as follows. Given two different rankings of the same n items, count the number of pairs that are concordant (in the same order) in both rankings, and discordant (in reverse order). If P is the concordance-count and Q the discordance-count, then:

$$\tau = \frac{P - Q}{P + Q}$$

The coefficient ranges from -1 to 1. The interpretation of Kendall τ is as follows: an increasing rank correlation implies increasing agreement between rankings. A coefficient of 1 denotes a perfect agreement (identical rankings). A coefficient of 0 indicates that the rankings are completely independent. A coefficient of –1 indicates a complete ‘disagreement’ between the two rankings, or with other words, one ranking is the reverse of the other. Note that we have re-mapped the KPI to a range between 0 and 1 in order to unify all quantities to a common range:

$$\tau_{\text{normalized}} = \frac{\tau + 1}{2}$$

3.6.2 Spearman’s Rho (ρ) metric

The second correlation measure that we use is Spearman’s Rho (ρ) metric. The metric is a modified version of the Karl Pearson coefficient (calculated on the rank numbers). Note that the Spearman ρ produces identical values as Kendall’s W metric (in the case of two judgment sets). Suppose there are n data pairs and their respective rank numbers are indicated by rank ri and si. We first define the difference as:

$$d_i = r_i - s_i$$

Assuming no duplicates in the data exists (true in our study) then we can define ρ using the following definition:

$$\rho = \frac{6 \sum d_i^2}{n(n^2 - 1)}$$
The natural function range is from -1 to 1. The extreme value -1 indicates that we review two rankings that are opposite of each other. A value of 0 means that we have two completely different rankings and the extreme 1 signifies two identical rankings.

Please note that we also scale this metric to the range of 0 to 1 so that there is full range unification for all our metrics. The scale computation is similar to the range mapping for the τ metric in section 3.6.1:

\[ \rho_{\text{normalized}} = \frac{\rho + 1}{2} \]

We found that computed Spearman ρ metrics tended to indicate slightly stronger relationships when there was positive concordance for two compared two rankings than the respective computed Kendall τ metrics.

### 3.6.3 Spearman’s Footrule.

The third and final correlation metric that we compute is a normalized version of Spearman’s Footrule. The Footrule distance is a well-known indicator of disarray for ranked data that has found applications in many areas of research. We compute a variant so that we can use the distance as a metric. The natural range for our metric is 0 (opposite rankings) to 1 (identical rankings), so there is no need for final scaling.

In order to compute the normalized metric we reuse some of the definitions from the Spearman ρ metric, and define than \( S \) as being the sum of the absolute differences:

\[ S = \sum_{i=1}^{n} |d_i| \]

Depending on whether \( n \) is even or odd we next compute the maximum value \( M \) for Spearman’s Footrule distance:

\[ M \mid n \text{ is even} = \frac{n^2}{2} \text{ or } M \mid n \text{ is odd} = \frac{(n-1)(n+1)}{2} \]

This allows us to compute the normalized Footrule metric with the following formula:

\[ F = 1 - \frac{S}{M} \]
3.7 Proximity and group stability metrics

Many Infobox templates tend to bundle related properties in partitions. Typically, related properties, such as for example area code and postal code, are found together. It may actually not matter all that much if either postal code or area code is ranked first, as long as the two properties are in close proximity. If a ranking algorithm orders the properties as [A,B], whereas the ground truth has ranked the properties in order [B, A], then the proximity is still identical, even though the ordering is actually reversed. It may even be argued that ranking [A, x, B] or [B, x, A] are still quite good orderings since the properties A and B are still in close proximity of each other. Reversals result in lower Kendall τ scores, so this metric cannot be used to measure proximity.

The metric Proximity Error (PE) described here is a count of the number of properties that fall outside a given proximity tolerance:

\[ PE = \sum_{i=1}^{n-1} I_{i,i+1} \]

With \( I_{i,j} \to \{0,1\} \) as an indicator function defined as:

\[ I_{i,j} = (D_{i,j} > PT) \lor (D_{i,j} < NT) \]

Where:

- \( D \) is a distance matrix. The rows and columns in \( D \) correspond with the properties in ground truth set order \([A, B, \ldots]\). Each pair in \( D \) contains the distance of the two properties with respect to the ranking under investigation. As an example, if a ranking has ranked properties in the sequence \([\ldots, A, x, B, \ldots]\), then \( D \) contains 2 for the pair (A,B) and -2 for the pair (B,A).
- \( PT \) is a given positive tolerance (default: 2).
- \( NT \) is a given negative tolerance (default: -2).

The range of \( PE \) is between 0 and \( n - 1 \). The metric \( PR \) below outputs a range between 0 (all properties are out of proximity) and 1 (all properties in proximity):

\[ PR = 1 - \left( \frac{PE}{n - 1} \right) \]
A small proportion of the Infobox templates group their properties using headings and/or subheadings. Ranking algorithms that can group related properties are more valuable than those that cannot. Users may care less about the exact sequential order of two related properties, as long as they are within reasonable proximity of each other. An interesting metric to compute therefore is the degree to which these groups remain intact when they are ordered by a ranking algorithm. This group stability statistic is only computable for DBpedia ontologies mapped from Infobox templates that define property groups. The metric Group Fragmentation (GF) presented hereunder measures group fragmentation:

\[
GF = \sum_{i=1}^{g} FRAG(R_i)
\]

Where:

- \( n \) is the number of properties.
- \( D \) is a \( g \times n \) matrix with:
  
  \( g \) being the total number of groups

  and

  \( D_{g,r} = 1 \) if the property at rank \( r \) is part of group \( g \) within the original ranking, and 0 otherwise.

- \( R_i \) is the \( i^{th} \) row vector of \( D \).
- \( CONSEC(R) \) is a function that returns the maximum length of a consecutive non-zero sequence in the row vector \( R \). If the maximum consecutive length < 2, then 0 is returned.
- \( SUM(R) \) is a function that returns the sum of the row vector \( R \).
- \( FRAG(R) \) is defined as follows:

\[
\begin{align*}
SUM(R) \geq 1 & \quad FRAG(R) = CONSEC(R) \\
SUM(R) < 1 & \quad FRAG(R) = SUM(R)
\end{align*}
\]

The range of \( GF \) is between 0 and \( n \). A metric \( GR \) with range 0 (all group ordering lost) to 1 (all groups intact) is defined as follows:

\[
GR = \left( \frac{GF}{n} \right)
\]
3.8 Classification / IR metrics

This section describes how we compute recall and precision metrics with respect to specific cut-off points.

3.8.1 Precision @ n

We compute the metric Precision @ n metric to denote the proportion of the top-n properties that are relevant with respect to the ground truth set. If r relevant properties have been retrieved at rank n, then:

\[ \text{Precision @ n} = \frac{r}{n} \]

Consider a ground truth set of 10 (ranked) properties for which we denote the first 5 properties as relevant. If the top 5 properties in a ranking algorithm result list are all marked as relevant in the ground truth set, and the next 5 are all non-relevant, then there is a 100% precision at a cut-off point 5. The precision at cut-off point 10 would be 50% for the same data points (indicating ‘head or tails’).

3.8.2 Recall @ n

The metrics recall @ n refers to the actual number of relevant properties selected up to the cut-off point in the result-list:

\[ \text{Recall @ n} = n \times \text{Precision @ n} \]

If t denotes the total number of relevant properties, then the recall proportion @ n signifies the relevance selection frequency at rank n:

\[ \text{Recall proportion @ n} = \frac{r}{t} \]
3.8.3 Balanced F-score

Another traditional metric in IR is the F-measure or balanced F-score ($F_1$ score). The $F_1$ metric is the harmonic mean of precision and recall, and can be interpreted as the weighted average of precision and recall. $F_1$ reaches its best value at 1 and worst score at 0:

$$F_1 = 2 \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

3.8.4 Average Precision

Finally, the metric Average Precision (AP) combines precision, relevance ranking and overall recall over the entire result list:

1. Let $n$ be the number of properties in a ranking algorithm result list
2. Let $RP$ be the total number of relevant properties
3. Let $p[i]$ be the $i^{th}$ property in the in a ranking algorithm result list
4. Let $r[i]$ be 1 if $p[i]$ is relevant and 0 otherwise.

$$\text{Precision}[j] = \sum_{k=1}^{j} \frac{r[k]}{j}$$

$$\text{Average precision} = \frac{\sum_{j=1}^{n} (\text{Precision}[j] \ast r[j])}{RP}$$

The AP metric is the sum of the precision at each relevant property in a ranking algorithm result list, divided by the total number of relevant properties in the ontology. Notice that non-relevant properties do not add anything to the numerator, but they reduce the precision at all points below, and as such reduce the contribution of each relevant property in a ranking algorithm result list. Average precision is a very good IR measure, albeit not a ‘perfect’ measure as it does not account any cost for non-relevant properties that appear below all relevant properties.
3.9 Comparing ranking algorithms

We compare the ranking algorithms in our research to the performance of alphabetic ranking using a paired sample \( t\)-test, specifically, a paired differences test. We computed the significance using this test for both the Kendall \( \tau \) and Spearman \( \rho \) metrics.

The paired sample test evaluates the average value from two measurement sets. More formally, a paired sample \( t\)-test is used to determine whether there is a significant difference between the average values of the same measurement made in two different conditions for the same subject. The test is based on the paired differences between these two values.

Two measurement sets are paired when they come from the same observational unit. Observations are considered paired if there is a natural link between an observation in one set of measurements, and a particular observation in the other set of measurements, irrespective of their actual values. Our observations are indeed paired since we rank a mapped class \( C \) using ranking algorithm \( A \) (for example, alphabetic ranking) and then rank the exact same mapped class using ranking algorithms \( B \). Hence, there is a clear pairing between the measurements sets.

The null hypothesis is that the difference in the mean values is. The null hypothesis \( H_0 \) is \( \mu = 0 \), meaning that any differences are due to chance. We can establish a significant statistical difference between two algorithms if we can reject \( H_0 \). We have used a confidence level of 95\% to calculate the critical values.

Note that that in all cases where a significant Kendall \( \tau \) result was found we also computed a significant Spearman \( \rho \) result (see paragraph 3.6.2 for an explanation why this is the case).
3.10 Benchmarks / extremes

We designed several benchmarks to test the proper working of the KPI computation software module. An important goal for these benchmarks is to validate the KPI computations via a set of predicted metrics. If the actual observed values deviate from the predicted values then this will typically indicate KPI implementation flaws, or possibly KPI model flaws. Note that we allowed the control benchmarks to use information that normal ranking algorithms could not access. As an example, the benchmarks could access the Infobox-ranking data directly. A second major benefit from these benchmarks is that they give the reader of this paper a perspective, but also guidance, towards the interpretation of the observed experiment metrics. The following sections describe the various benchmarks.

3.10.1 Benchmark: perfect ranking

This benchmark inspects the properties from an ontology class C and then orders the properties identically as the ranking data extracted from the respective Infobox template ranking I for a given language L. This procedure yields a ranking that exactly replicates the ordering from I, or with other words, creates a ‘perfect’ ordering. All properties are ordered exactly according to the input Infobox template ranking. This is also explained in schematic form in Table 6.

<table>
<thead>
<tr>
<th>Infobox template ranking</th>
<th>Perfect ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property A</td>
<td>Property A</td>
</tr>
<tr>
<td>Property B</td>
<td>Property B</td>
</tr>
<tr>
<td>Property C</td>
<td>Property C</td>
</tr>
<tr>
<td>Property D</td>
<td>Property D</td>
</tr>
<tr>
<td>Property E</td>
<td>Property E</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Property Z</td>
<td>Property Z</td>
</tr>
</tbody>
</table>

Table 6: schematic form for perfect ranking

The expected normalized Kendall τ output (see Figure 17) is indeed the observed value of 1.0 given the perfect ranking concordance with the baseline ranking. The other computed metrics are also in line with their expected values.
3.10.2 Benchmark: reverse ranking

This benchmark takes an ontology class $C$ and orders the properties reverse with respect to the ranking data extracted from the respective Infobox template ranking $I$ for a given language $L$. This is explained in schematic form in Table 7. The ranking output is thus a negated version of the Infobox template source.

<table>
<thead>
<tr>
<th>Infobox template ranking</th>
<th>Reverse ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property A</td>
<td>Property Z</td>
</tr>
<tr>
<td>Property B</td>
<td>…</td>
</tr>
<tr>
<td>Property C</td>
<td>Property E</td>
</tr>
<tr>
<td>Property D</td>
<td>Property D</td>
</tr>
<tr>
<td>Property E</td>
<td>Property C</td>
</tr>
<tr>
<td>…</td>
<td>Property B</td>
</tr>
<tr>
<td>Property Z</td>
<td>Property A</td>
</tr>
</tbody>
</table>

Table 7: schematic form for reverse ranking

The observed normalized Kendall $\tau$ output is 0 (see Figure 18), which is expected, as there is a complete reverse ranking.

Figure 17: KPI metrics for perfect ranking benchmark

Figure 18: KPI metrics for reverse ranking benchmark
3.10.3 Benchmark: random ranking

This benchmark considers an ontology class $C$ and produces a scrambled random order of all the properties in $C$, as is shown in Table 8.

<table>
<thead>
<tr>
<th>Infobox template ranking</th>
<th>Random ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property A</td>
<td>Property E</td>
</tr>
<tr>
<td>Property B</td>
<td>…</td>
</tr>
<tr>
<td>Property C</td>
<td>Property R</td>
</tr>
<tr>
<td>Property D</td>
<td>Property C</td>
</tr>
<tr>
<td>Property E</td>
<td>Property M</td>
</tr>
<tr>
<td>…</td>
<td>Property Z</td>
</tr>
<tr>
<td>Property Z</td>
<td>Property L</td>
</tr>
</tbody>
</table>

Table 8: schematic form for random ranking

The expected normalized Kendall τ output (see Figure 19) is almost perfectly aligned with the expected value of 0.5 (meaning: no correlation to the Infobox ranking). The other correlation metrics are close enough to their expected values. A perfect match is extremely unlikely due to the involved randomness.

Figure 19: KPI metrics for random ranking benchmark
### 3.10.4 Benchmark: half-perfect ranking

This benchmark takes an ontology class $C$ and utilizes the ranking data extracted from the respective Infobox template $I$ for a given language $L$ to compute a “half-perfect” rank order. To understand, consider a function $I(n)$ that yields the $n^{th}$ property of ranking $I$. The half-perfect ranking creates two groups of properties: $G1$ and $G2$. Group $G1$ contains the first $n/2$ properties from $I$, so $I(1), I(2)$ and so forth. Group $G2$ contains the remaining properties. Group $G1$ is therefore in perfect Infobox template order, whereas we sort the properties in group $G2$ in reverse Infobox template order. The final ranking is the concatenation of the properties from $G1$ and $G2$, as is depicted in schematic form in Table 9.

<table>
<thead>
<tr>
<th>Infobox template ranking</th>
<th>Half-perfect ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property A</td>
<td>Property A</td>
</tr>
<tr>
<td>Property B</td>
<td>Property B</td>
</tr>
<tr>
<td>Property C</td>
<td>Property C</td>
</tr>
<tr>
<td>Property D</td>
<td>…</td>
</tr>
<tr>
<td>Property E</td>
<td>Property Z</td>
</tr>
<tr>
<td>…</td>
<td>Property Y</td>
</tr>
<tr>
<td>Property Z</td>
<td>Property X</td>
</tr>
</tbody>
</table>

Table 9: schematic form for half-perfect ranking

The expected normalized Kendall τ output is 0.75 as half of the properties are in correspondence with the Infobox template ranking. The graph in Figure 20 shows that the metrics are in line with the predicted model values. Note the slightly odd-looking proximity metrics. The rationale is explainable by observing the above table. In the Infobox template ranking, property $Z$ has neighbour property $Y$, and no second neighbour. In the benchmark ranking, property $Z$ is still in proximity with its original neighbour property $Y$. Hence, the only property that gets a new neighbour is at the switch point where group $G1$ transitions into group $G2$. The proximity KPI’s are identical regardless of the scanning range.

![Figure 20: KPI metrics for half-perfect ranking benchmark](image-url)
3.10.5 Benchmark: half-reverse ranking

This benchmark is identical in form and shape with respect to half-perfect ranking, but the properties in the first group $G1$ are ordered in reverse Infobox template order, and the second group $G2$ is ranked in perfect Infobox template order (see Table 10 for the schematic form).

<table>
<thead>
<tr>
<th>Infobox template ranking</th>
<th>Half-reverse ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property A</td>
<td>Property Z</td>
</tr>
<tr>
<td>Property B</td>
<td>Property Y</td>
</tr>
<tr>
<td>Property C</td>
<td>Property X</td>
</tr>
<tr>
<td>Property D</td>
<td>…</td>
</tr>
<tr>
<td>Property E</td>
<td>Property A</td>
</tr>
<tr>
<td>…</td>
<td>Property B</td>
</tr>
<tr>
<td>Property Z</td>
<td>Property C</td>
</tr>
</tbody>
</table>

Table 10: schematic form for half-reverse ranking

The expected normalized Kendall $\tau$ output is 0.25 as half of the properties are in negative correspondence with the Infobox template ranking. The graph in Figure 21 shows KPI’s that meet the prediction values.

![Figure 21: KPI metrics for half-reverse ranking benchmark](image-url)
3.10.6 Benchmark: alphabetical ranking

Alphabetical ranking is an important baseline reference point throughout this research. We sort the properties using the ontology property label that is associated with the respective language under research. Many Semantic Web applications utilize this form of ranking. Hence, the performance indicators for alphabetical ranking are key metrics and we use these throughout our experiments as a baseline. We consider any ranking algorithm that can statistically outperform alphabetical ranking as a good method. The graph in Figure 22 outlines the key performance metrics for alphabetical ranking.

An interesting comparison to make is the performance of alphabetical ranking versus random ranking. One of the key observations is that the key ranking metrics (such as Kendal Tau and friend functions) demonstrate that alphabetical ranking produces a slightly better property ranking than random ranking (that is, for English). This implies that human rankings tend to include some form of alphabetical ranking, at least to some extent. Little other rationale can explain the higher level of concordance.

The proximity metrics, however, do not show an improvement, which is a second interesting observation. It means that post alphabetical ranking the properties are overall less in proximity as compared to a random ranking. This indicates that alphabetical ranking does not maintain proximity very well. Recall and precision are better after alphabetical ranking than in random ranking. This indicates that the key properties of an Infobox are better selected using alphabetical sorting. All this makes sense given the human tendency to order things; even it means alphabetical ordering rather than some more ‘natural’ ranking that is based on topic content.

![Figure 22: KPI metrics for alphabetical ranking benchmark](image-url)

40 <rdfs:label xml:lang="iso">
Chapter 4  Conducted experiments

4.1 Introduction

This chapter outlines several experiments that test various ranking algorithms through empirical evaluation. We describe four approaches based on the terminological analysis of a given ontology, and one algorithm that employs ontology structural analysis via a heuristics-based method. The ranking performance is evaluated through a series of computed performance metrics that measure the effectiveness of a ranking algorithm. Each performance indicator quantifies an aspect of the ranking algorithms value.

4.2 Terminological-based ranking: word frequency matching

In this experiment we computed the ranking of class properties using external word frequency lists. More specifically, we examine the terminology of a property, determine its most specific term, then lookup the frequency of that term in the word list and finally sort the properties in decreasing frequency order. Table 11 demonstrates this logic in schematic form. The numbers between parentheses denote the derived property term frequency.

<table>
<thead>
<tr>
<th>Infobox template ranking</th>
<th>Experiment ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property A (100)</td>
<td>Property E (8000)</td>
</tr>
<tr>
<td>Property B (2000)</td>
<td>Property Y (6000)</td>
</tr>
<tr>
<td>Property D (500)</td>
<td>…</td>
</tr>
<tr>
<td>Property E (8000)</td>
<td>Property A (100)</td>
</tr>
<tr>
<td>…</td>
<td>Property C (30)</td>
</tr>
<tr>
<td>Property Z (400)</td>
<td>Property R (2)</td>
</tr>
</tbody>
</table>

Table 11: word frequency ranking in schematic form

Compound phrase processing is one of the key challenges that we faced throughout this research. Multi word terms (such as “birth date”) are typically not included in word frequency lists. There is no fail-safe way to select the most dominant word from a given compound term. For a term such as “birth date” it is very debatable whether “birth” or “date” is the more important word; the semantic meaning is typically found only in the entire phrase.
As we still have to deal with compound terms we use a simplification procedure to derive a single term from a compound term. We first inspect the keywords that we extracted from Alchemy API in the pre-processing step (see page 47). If Alchemy API was able to determine a single keyword for a given phrase as the most dominant term then we select this keyword for further frequency analysis. As an example, suppose we are computing the rank for a term such as “processing date”. Alchemy API indicated “date” as the most dominant keyword with a relevance score of 0.91 for this compound term. We can then lookup the frequency for “date” and use that derived frequency to score “birth date”.

Unfortunately, we found that Alchemy API could not determine the most specific keyword from a short phrase compound term. Statistical keyword extraction simply works better for longer phrases. We would in such cases fall back to alternative Alchemy API category data such as derived concepts or the derived taxonomy classification (see also page 47). However, even these might not always be available. Furthermore, bear in mind that the conversion of a compound term towards category data is per definition a simplification of the original terminology. As a consequence, an amount of noise is included in this processing step. For example, the Alchemy API category for “birth date” is “astrology”. It is possible to establish the word frequency for “astrology”, but the relevancy of the derived frequency for the compound term “birth date” is obviously debatable.

When Alchemy API would not offer any form of category data we turned to WordNet to see if it contained synonyms, derived words or hypernyms41 for a given compound term. A compound term like “fiscal year” does occur in WordNet. A WordNet hypernym for “fiscal year” is “year”, so we can use that relationship to lookup the frequency, and thus determine a derived frequency for the compound term “fiscal year”. However, this fall-back does not always work. In a return to the example of “birth date” we find that WordNet does not include that term, and hence, even this strategy will fail.

If neither Alchemy API nor WordNet has related data we cannot establish a term frequency, and the frequency is then established as zero occurrences.

41A word with a broad meaning constituting a category into which words with more specific meanings fall. For example, color is a hypernym of red.
The full ranking logic is defined next in a more formal form:

1. Single word term?
   Use frequency from word list

2. Alchemy API reports (single word) keywords.
   Establish frequency from most relevant (single word) keyword

3. Alchemy API reports (single word) concepts.
   Establish frequency from most relevant (single word) concept with correction factor 0.75

4. Alchemy API reports (single word) categories.
   Establish frequency from most relevant (single word) category with correction factor 0.50

5. Multi Word term is not available in WordNet?
   Establish frequency as zero occurrences.

6. WordNet defines (single word) synonyms?
   Establish frequency from most relevant (single word) synonym with correction factor 0.25

7. WordNet defines (single word) derived words?
   Establish frequency from most relevant (single word) derived word with correction factor 0.25

8. WordNet defines (single word) hypernyms.
   Establish frequency from most relevant (single word) hypernym with correction factor 0.25

9. Establish frequency as zero occurrences.

We used correcting factors to offset the frequency computations for compound terms to reflect the uncertainty of compound term simplification. In addition, please note that Alchemy API reports relevance metrics for most operations. We factor these relevance scores, together with the before mentioned correction factors, into a relative relevance.
As an example, assume that Alchemy API has denoted “date” as a keyword with relevance 0.91. Also, assume that we have a correction factor of 0.75 to offset an amount of uncertainty, and that our word list defines “date” as having a corpus frequency of 33858. The relative relevance then becomes $0.91 \cdot 0.75 \cdot 33858 = 23108$. As so often in optimization problems, finding optimal correction values is a challenge, partly due to the vast amount of processing required in finding optimum values. We do however believe that offsetting is sensible to reflect the uncertainty when processing simplified or derived terminology.

We used three different lists for our experiments as we found that different word frequency lists resulted into measurable ranking differences. The first list is composed of the 21K most frequent words from the Corpus of Contemporary American English (COCA)\textsuperscript{42}. The second list is the Bing 100K word list\textsuperscript{43} of most frequently indexed terms (English web pages as of April 2013). The third frequency list reflects the word frequency in English Wikipedia articles (+/- 800K words)\textsuperscript{44}.

4.2.1 Result discussion

The best results that were obtained have been computed using the Wikipedia word frequency list and are listed in Figure 23.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure23.png}
\caption{KPI metrics for word frequency ranking (Wikipedia frequency list)}
\end{figure}

\textsuperscript{42} http://corpus.byu.edu/coca/
\textsuperscript{43} http://datahub.io/dataset/microsoft-web-n-gram-service
\textsuperscript{44} http://goo.gl/NFWo94
The comparison to alphabetical ranking reveals that the ranking results for all examined frequency lists are better for most metrics (see Figure 24). A key improvement in frequency-based ranking is the improved precision.

![Figure 24: KPI improvements for frequency ranking over alphabetic ranking](image)

Word frequency ranking is in fact a statistically significant improvement over alphabetic ranking, as can be observed from the details of a paired samples t-test in Figure 25. The results are significant both for the normalized Kendall τ metrics and the normalized Spearman ρ metrics (note: Spearman ρ data not shown here).

![Figure 25: paired samples t-test for word frequency ranking](image)
A paired samples t-test is only valid if the difference measures are normally distributed. The box-plot in Figure 26 indicates that the normalized Kendall τ differences distribution is sufficient symmetric.

![Box Plot](image)

Figure 26: Kendall τ differences distribution for word frequency ranking

As mentioned the Wikipedia frequency list (Figure 23) performed best overall. Bing is a close follow-up (Figure 27) with the COCO data coming last (Figure 28). The Wikipedia word frequency list contains eight times more terms as the Bing list, and forty times more than COCA. Whether or not there is a direct relation to the word list size remains to be seen though. We can however safely conclude that the input word list is a very relevant factor for the overall performance.

![Figure 27](image)

Figure 27: KPI metrics for word frequency ranking (Bing frequency list)

![Figure 28](image)

Figure 28: KPI metrics for word frequency ranking (COCA frequency list)
Intuitively, one might aspect that ranking would be more efficient if we sort properties in order of least frequent occurring terms first, and the most frequently occurring terms last. This would match with the expectation that a rare term is more important than a very common term, and therefore should appear earlier. Consider a term such as “equalization” (frequency 280 in COCA). It seems sensible that this term holds more useful information (aka Shannon entropy) than a term such as “system” (frequency 215748 in COCA), and thus should obtain a higher rank. However, experiments with inverse ranking showed that this is actually not the case, and in fact, the results are contrary to our expectations, as can be seen in a reverse frequency ordered experiment based on the Wikipedia frequency list (see Figure 29).

The performance of reverse ranking is nearly 15% worse for the Spearman $\rho$ KPI and 9% worse for the Kendall $\tau$ metric. We also conducted some experiments where we ranked less-frequent terms first and also ‘boosted’ the 25% most rare terms. This approach did not improve the results. Clearly, terms that are more frequent should appear first.

![Figure 29: KPI metrics for reverse ranking (Wikipedia frequency list)](image)

Finally, we present figures for an experiment variant where we did not simplify multi-word phrases. When confronted with compound phrase properties we simply took the average frequency of all recognized words in the phrase. We used the Wikipedia list again, and the results are significantly worse than those that we observed in the more complex compound term-to-word simplification approach (see Figure 30). Note the remarkable difference in precision when compared to the regular (Wikipedia) frequency rank computation from Figure 23. The metrics reveal that an averaging approach is not efficient and that alphabetic ranking even outperforms this ranking method.
Overall, whilst reviewing term frequency ranking, we found that the ranking performance outperforms alphabetical ranking, although there is some dependency on the specific word frequency list in use. The method is fast but does require interaction with a NLP service such as Alchemy API to provide fall-back data in order to process multi-word phrases effectively. Without this improvement step, frequency ranking is not significantly better than alphabetic ranking. Hence, disambiguation for compound term processing is a key factor with respect to the performance characteristics of this ranking algorithm.

Figure 30: KPI metrics for average term frequency (Wikipedia frequency list)
4.3 Terminological-based ranking: N-gram probability

An N-gram is commonly defined as a contiguous sequence of \(n\) items from a given sequence of text. Consider a (three word) phrase like “he walked home”. Such phrases are referred to within literature as 3-grams (or trigrams). More generally, an N-gram is defined as the probability of a specific N-word phrase occurring within some source corpus. Natural language processing applications frequently use N-grams as a basis for text analysis. Applications range from software that correct spelling to compression algorithms to generative text. In this research we used 5-grams to predict property order ranks.

We sourced our 5-gram data from the Microsoft Web N-gram service\(^{45}\). This service provides N-grams from Microsoft Bing search indexed documents for the English-American market. We used the updated 2013 data sets from Bing for our experiments. The service has various smoothed N-gram models that reflects content extracted from document bodies, document titles and anchor texts. There are unigram, bigram, trigram, 4-gram and 5-gram sets available. For our research we opted for the 5-grams as they clearly demonstrated the best performance overall in our experiments. These experiments also showed that the 5-grams extracted from document bodies performed slightly better than the 5-grams from document titles or document anchors (this is shown later in the KPI metric discussion).

The applied ranking algorithm is now described in full detail. For each property in a DBpedia ontology class we request the Microsoft N-gram service to find the joined probability of the words in the respective property label. Specifically, it computes the base-10 log of the joined probability of the word sequence. If a given word sequence is denoted as \([w_1^, w_2^, \ldots, w_n^]\) than the probability is computed as \(P(w_1^) \cdot P(w_2^|w_1^) \cdot \ldots \cdot P(w_n^|w_{n-m+1}^ \ldots w_{n-1})\) where \(n=5\) as we use the 5-gram model. We then sort the properties in order of most probable phrases first. The schematic form of this ranking algorithm is shown in Table 12; the numbers between parentheses indicate the 5-gram probability.

<table>
<thead>
<tr>
<th>Infobox template ranking</th>
<th>Experiment ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property A (0.1)</td>
<td>Property E (0.8)</td>
</tr>
<tr>
<td>Property B (0.02)</td>
<td>Property Y (0.6)</td>
</tr>
<tr>
<td>Property C (0.03)</td>
<td>Property D (0.5)</td>
</tr>
<tr>
<td>Property D (0.5)</td>
<td>...</td>
</tr>
<tr>
<td>Property E (0.8)</td>
<td>Property A (0.1)</td>
</tr>
<tr>
<td>...</td>
<td>Property C (0.03)</td>
</tr>
<tr>
<td>Property Z (0.4)</td>
<td>Property B (0.02)</td>
</tr>
</tbody>
</table>

Table 12: N-gram ranking in schematic form

4.3.1 Result discussion

The KPI metrics for N-gram ranking are shown in Figure 31 (5-grams, 2013-Body model).

![Figure 31: KPI metrics for metrics joined probability ranking (body)](image1)

In Figure 32 we review individual KPI improvements relative to alphabetic ranking. As can be observed this ranking algorithm outperforms alphabetical ranking by about 8%. Proximity is even better, up to 9%. Precision is nearly 20% better, and recall and group stability show a modest improvement. We may conclude from these metrics that N-gram-based ranking offers reasonable performance. In addition, this method requires no pre-computations whatsoever, is relatively simple to implement and executes reasonably fast.\(^{46}\)

![Figure 32: KPI improvements for N-gram ranking over alphabetic ranking](image2)

---

\(^{46}\) The execution is obviously bound to the speed of the N-gram (web) service
The performance of the N-gram ranking algorithm is a statistically significant improvement over alphabetic ranking, as can be observed from the paired samples \( t \)-test data that is summarized for the normalized Kendall \( \tau \) metric in Figure 33. The normalized Spearman \( \rho \) metric results are also significant (but not shown here).

![Statistical t-test: two paired samples Kendall-Tau](image)

<table>
<thead>
<tr>
<th>Group</th>
<th>Count</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Std Err</th>
<th>t</th>
<th>df</th>
<th>Cohen d</th>
<th>Effect r</th>
</tr>
</thead>
<tbody>
<tr>
<td>N-gram</td>
<td>210</td>
<td>0.61341</td>
<td>0.16384</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alphabetical</td>
<td>210</td>
<td>0.52343</td>
<td>0.11643</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Difference</td>
<td>210</td>
<td>0.08099</td>
<td>0.21358</td>
<td>0.01447</td>
<td>5.59676</td>
<td>217</td>
<td>0.37920</td>
<td>0.36527</td>
</tr>
</tbody>
</table>

Results for \( t \)-test:

<table>
<thead>
<tr>
<th>p-value</th>
<th>t-crit</th>
<th>lower</th>
<th>upper</th>
<th>sig</th>
</tr>
</thead>
<tbody>
<tr>
<td>One Tail</td>
<td>0.00000</td>
<td>1.65191</td>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>Two Tail</td>
<td>0.00000</td>
<td>1.97096</td>
<td>0.00248</td>
<td>0.16950</td>
</tr>
</tbody>
</table>

Figure 33: paired samples \( t \)-test for N-gram ranking

We need to make sure that the assumptions for the \( t \)-test hold, namely that the difference measures are normally distributed or at least reasonably symmetric. From Figure 34 we see that this is the case.

![Distribution box plot](image)

Figure 34: normalized Kendall \( \tau \) differences distribution for N-gram ranking
Bing offers multiple N-gram models. The body model represents N-grams extracted from HTML body sections, but there are also models that have been extracted from HTML titles and HTML anchors. We tested these models to review if they made a significant difference in terms of performance. The results are visualized in Figure 35 (5-grams, 2013-Title model) and Figure 36 (5-grams, 2013-Anchor model). The differences are not substantial, but the title model performs slightly worse than the body model. The anchor model performed worst, but again, the differences are minimal.

Figure 35: KPI metrics for joined probability ranking (titles)

Figure 36: KPI metrics for joined probability ranking (anchors)
The Microsoft N-gram service also offers a conditional probability model that is computed as $P(w^n | w^{n-m+1} \ldots w^{n-1})$. This model however performed significantly worse as may be seen in Figure 37 (5-grams, 2013-Body model).

![Figure 37: KPI metrics for conditional joined probability ranking (body)](image)

We also examined if it made sense to reverse sort on the 5-gram probability, or in other words, least probable properties first. The KPI graph in Figure 38 (5-grams, 2013-body model) shows clearly that reverse sorting is a very bad idea indeed, with up to 20% performance loss.

![Figure 38: KPI metrics for reverse-sorted joined probability ranking (body)](image)
4.4 Terminological-based ranking: term similarities

This experiment investigates the notion that people tend to group related concepts together when they order information. Conceptual relatedness is difficult to specify well. There is a wealth of research into this topic within the realm of natural language processing. In this experiment, we choose to use four metrics that deal with term relatedness in order to determine the level of property correspondence.

The derived relatedness in this experiment stems solely from WordNet word relationship structures. The semantic relatedness is expressed through the four measures, but as so often, each metric has its shares of pros and cons. What follows next is a short generic description of four WordNet-based semantic relatedness metrics [39] that we selected for this experiment.

- **Path** computes the semantic relatedness of two terms by counting the number of nodes along the shortest path between two WordNet terms (in an ISA hierarchy).

- **Resnik** defines the similarity based on the lowest superordinate of two terms, also known as the most specific common subsumer. A subsumer is defined here as an ancestor node of both terms. For example, *animal* and *mammal* are both subsumers of *cat* and *dog*, but *mammal* is a lower subsumer than *animal* (which is the most specific common subsumer). Resnik includes probabilistic information derived from a corpus to enhance the metric.

- **Lin** is an extension on Resnik and takes the differences between two terms into account, rather than just similarities, using the lowest common subsumers of the two compared concepts.

- **WU-Palmer** calculates relatedness by considering the depths of the two terms in the WordNet taxonomies, along with the depth of the least common subsumer.

These algorithms were selected because they cover a range of relatedness computation mechanisms, are reasonably fast to compute and yield values in a comfortable range for further processing (0 = not related, 1 = most related).
One problem that had to be addressed for this approach is the issue of disambiguation. Consider an example for the English word ‘model’ in Table 13.

<table>
<thead>
<tr>
<th>Noun</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>theoretical account, framework (a hypothetical description of a complex entity or process)</td>
<td>model (a type of product)</td>
</tr>
<tr>
<td>poser (a person who poses for a photographer or painter or sculptor)</td>
<td>simulation (representation of something (sometimes on a smaller scale))</td>
</tr>
<tr>
<td>exemplar, example, model, good example (something to be imitated)</td>
<td>role model (someone worthy of imitation)</td>
</tr>
<tr>
<td>example (a representative form or pattern)</td>
<td>mannequin, manikin, fashion model, model (a woman who wears clothes to display fashions)</td>
</tr>
<tr>
<td>modelling, modeling (the act of representing something (usually on a smaller scale))</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Verb</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>pattern (plan or create according to a model or models)</td>
<td>mold, mould (form in clay, wax, etc.)</td>
</tr>
<tr>
<td>pose, sit, posture (assume a posture as for artistic purposes)</td>
<td>model (display (clothes) as a mannequin)</td>
</tr>
<tr>
<td>simulate (create a representation or model of)</td>
<td>mock up (construct a model of)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Adjective</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>exemplary, model (worthy of imitation)</td>
<td></td>
</tr>
</tbody>
</table>

Table 13: different meanings for the term “model”

As can be observed from Table 13 is that the word ‘model’ alone already has 16 senses in WordNet. Which sense of model in WordNet is the right model that is actually intended from the DBpedia ontology class perspective? There is no way to tell, and it is worth mentioning that disambiguation is an unsolved and extremely difficult problem to resolve. To deal with this uncertainty we computed all sense combinations. We illustrate this approach with an example. Assume we want to compute the relatedness of the term “model” with respect to the term “shape” (which has 11 senses). Our algorithm will compute all 176 variants (16 x 11). This ultimately yields 704 relatedness metrics as we compute four relatedness metrics per iteration. The entire approach is explained in full detail in the next section.

---

47 Data extracted from WordNet senses available for the term “model”
We now outline the experiment in full detail. The input is an ontology class $C$ from the DBpedia ontology. We then iterate over all properties in no particular order. We first compute a non-compound term for each property $p$ if the label of $p$ is compound. For each iterated property $p$ a term $t$ is extracted which is computed as follows:

1. If the ontology defines a non-compound label for $p$ then this is term $t$.

2. If the (highest ranking) Alchemy API keyword for $p$ is non-compound then use this as term $t$.

3. If the (highest ranking) Alchemy API concept for $p$ is non-compound then use this as the term $t$.

4. Assign an empty string to term $t$.

This step does incur a considerable amount of noise as the generalization from compound terms to non-compound terms results in loss of term accuracy. Unfortunately, we do require non-compound terms as WordNet does not contain many compound terms.

---

48 A compound term has more than one word, for example “birth date”.
The next step in the experiment involves the computation of the relatedness of all term pairs. We computed the relatedness metrics using the WordNet Similarity for Java\textsuperscript{49} library. The relatedness metrics are derived in the following manner for any given two (non-identical) terms $t$ and $u$:

1. Extract all WordNet senses for $t$ into sense list $r$.

2. Extract all WordNet senses for $u$ into sense list $s$.

3. Compute the sum of all four metrics for all sense pairs $(r^i, s^i), \ldots, (r^n, s^n)$

4. Compute the similarity score (sum of metrics divided by the number of sense pairs). The resultant score determines how two terms are related.

In the last preparation step of the algorithm we map each property $p^x$ to another property $p^y$ based on the similarity score as to yield a set of related pairs $(p^x, p^y)$.

We then finally order the properties in order of relatedness. If a relationship between $(p^x, p^y)$ is stronger than between $(p^a, p^b)$ then the rank order becomes $p^x, p^y, p^a, p^b$.

Furthermore, if $p^y$ lies alphabetically before $p^x$, than the ordering for $(p^x, p^y)$ would be $p^y, p^x$. Bear in mind that a relationship of $(p^x, p^y)$ does not imply symmetry $(p^y, p^x)$. Symmetry is not possible since any ontology class with a non-even amount of mapped properties would violate this assumption, but furthermore the metrics simply do not always yield symmetric pairs.

\footnote{https://code.google.com/p/ws4j/}
4.4.1 Result discussion

Slightly surprising, term similarity ranking is not a technique that performs very well on any evaluation dimension. All metrics in Figure 39 clearly demonstrate a performance that is under alphabetic ranking. In fact, property ordering is very similar to the performance of random ranking. The proximity metrics are twice as good as random ranking though. This is an indication that the properties tend to better remain in proximity than in a random ordering approach. However, the proximity values are not a major improvement over alphabetic ranking. Recall and precision are also underperforming with respect to alphabetic ranking. Furthermore, the computation performance of the algorithm is overall on the slow slide due to the many computations typically required to compute the similarity metrics. Ultimately, word similarity ranking is not a very useful method, and offers no benefit over alphabetical ranking.

A closing remark that we want to make is that we also crafted several variants where we computed the rank order using additional aspects than similarity score alone. One particular variant that we ran created weighted graphs with vertices denoting properties and the edges denoting relationships (with the edge weights set to reflect the strength of the relatedness). We then examined this graph to find an optimum Hamiltonian cycle (aka ‘Traveling Salesman’ problem). The result of that approach was a slightly better ordering (one to two percent improvements observed in the Kendall $\tau$ based metrics), but the proximity marks would actually decrease a bit.
4.5 Terminological-based ranking: dynamic corpus

We now describe the most complicated ranking algorithm that we considered in this research. The governing idea that underlies the dynamic terminological ranking algorithm is the assumption that ontology property ranking is effective through terminology analysis of a dynamically generated relevant corpus. The approach involves the dynamic creation of a corpus which is relevant to the ontology class that we want to rank. For example, if we want to rank the ontology class ‘Planet’ than the resultant corpus should contain documents that are relevant to the domain of planetary objects.

A ranking algorithm constructs a corpus by searching the internet for domain-specific documents that are relevant with respect to the given DBpedia class. A ranked terminology list is then extracted from the corpus which is matched to the ontology properties in order to establish the property ranking. Figure 40 graphically illustrates the high level processing steps of the algorithm.

The ranking approach is inspired by a recent work from Rospocher et al [28]. Their work is stimulated by earlier research from Jones & Alani [26] and also Brewster et al [27]. These studies focus on the domains of ontology comparison and ontology evaluation via terminological analysis but do not study the application of ontology property ranking. The paper from Rospocher et al does however hint towards the application of property ranking using terminological analysis. Our implementation borrows concepts from the previously mentioned works but is novel and unique on several aspects. We have expanded on the ideas from the earlier works in order to apply terminological analysis to the domain of this study. The earlier works are different on many aspects and also lack a full evolutonal study with respect to property ranking.
A ranking algorithm contains the following key processing steps:

1. **Extraction of terminology from the ontology structure:** The given ontology structure is analyzed in order to extract the core terminology.

2. **Search term generation:** The algorithm transforms the core terminology from the ontology into search terms. This is a key step as we will delegate an internet search engine to retrieve the most relevant documents from its index given our search query. If the search terms are inappropriate or not optimized then this will severely impact the selection of appropriate documents.

3. **Corpus generation:** In this step, the algorithm submits the search terms to a search engine such as Google. The top $n$ (ranked) search result Unique Resource Locators (URL’s) are then crawled and analyzed in order to append the core textual content to the corpus.

4. **Corpus terminology classifier:** The system analyzes the constructed corpus using terminological analysis in order to generate a ranked list of corpus terms.

5. **Property ranker:** Finally, the algorithm matches the ranked corpus term list to the properties from the ontology as to establish relative property importance.

The first stage involves the generation of relevant search terms to select documents from which a corpus may be generated. The generation of good search terms is one of the trickiest steps in the entire process. A search for a term such as ‘planet’ demonstrates the involved complexity quite well. All search engines that we tested yielded top ten search results that have nothing to do with planetary objects for this query. Search results ranged from links to ‘planet fashion’ to ‘animal planet’ and about anything in between. Bear in mind that in the specific case of the term ‘planet’ we actually have a very limited amount of ambiguity to deal with. Many terms however have different semantic senses that depend on the context. An example was already given on page 79 for the term ‘model’ for which we listed 16 semantically different interpretations.

To circumvent the ambiguity problem, or at least, to some extent, we include additional terms from the ontology class hierarchy. We traverse the class hierarchy bottom up to yields additional class labels that are incorporated as additional search terms. As an example, consider the DBpedia ontology class Planet, which is a subclass of the CelestialBody class. The extra class hierarchy traversal generates an extra search term ‘celestial body’. Additional context helps search engines to retrieve more relevant search results.
An additional improvement to further increase search result accuracy involves the inclusion of additional class properties as supporting search terms. However, search engines have limits on the amount of query terms that they accept (in the case of Google for example, a maximum of 32 terms). For classes with a large amount of properties, these limitations do actually manifest, as we must select properties to include or exclude. There may be other limitations too, for example with respect to the total number of characters in a query string.

We discovered that Google (as an important example) quickly fails to deliver any search results if the search query becomes too specialized. As an example of this phenomena, consider that we want to rank the DBpedia ontology class Planet. If we take a semi-random sample of the properties from the Planet class in order to construct a search query that is specific to the topic context we may construct a query that looks like this:

- `+planet`
- `+"celestial body"`
- `+perihelion`
- `+temperature`
- `+magnitude`
- `+discoverer`
- `+albedo`
- `+aphelion`
- `+periastron`
- `+apoapsis`
- `+"orbital period"`
- `+"surface area"`
- `+satellite`
- `-dictionary`
- `-glossary`
- `-wikipedia`
- `-dbpedia`
- `-infobox`
- `-WordNet`

This query includes only 11 out of 22 properties from the Planet class; however, Google would not return any search results for this query.

---

50 The mandatory (+) term operator is used on all property terms; exclusion (-) terms were added to remove “predictable sources of search result noise”.

---

85
We discovered that including mandatory operators restricted the ability of Google to produce an appropriate amount of search results. A workaround for this problem is obviously to skip the mandatory operator for all property terms. In fact, we found that it worked best to even exclude the mandatory operator from key terms such as the class label terms. Most search engines we tested will then revert to a mode where a type of AND/OR operator is assumed for the given terms. Typically, any of the given terms in an indexed resource will suffice for result classification. The top results will usually be the resources with the most terms included\(^5\). A side effect of this approach is that it becomes somewhat unpredictable which properties are favored as this depends on internal search engine logic. One search engine may favor term frequency as the most dominant factor, where another may use other weighting criteria. In general, we aim to rank the most “obscure” properties lowest since casual data observers probably do not care too much about these. We assume that this goal is aligned with typical search engine strategies.

With all the mandatory operators removed, Google still only yielded a result set of seven URL’s. When we take into consideration that our search query included only 11 out of 22 properties from the DBpedia Planet class, a further level of constraint alleviation is required. To increase the number of search results we opted for a mechanism where a search query is limited to three property search terms. In order to decide which properties are converted to search terms we used a term frequency list to select the most common properties only (based on term usage frequency). Obviously, this generates bias towards the query results since this will tend to select indexed documents where such properties are omnipresent. Note that common or over-generic (and therefore search-disruptive) properties are excluded; examples being properties such as “homepage”, “name” and so forth.

Unfortunately, we could think of few ways that are better to resolve this problem. An alternative solution to this problem is to select a random number of properties for the search query, but this would ruin any chances for repeatable results. Non-repeatable results are problematic as they hinder the evaluation; this aspect is further elaborated in the result discussion paragraph. By selecting the most frequently used terms we are in some way imitating the way a typical casual data observer would think, which is in fact aligned to one of the research goals for this work. A second benefit is that we have installed a repeatable mechanism to construct query terms. Future work in this area could quite possibly improve this area, but the problem does demonstrate the issues with terminology-based approaches.

\(^5\) Although this is not required, and is in fact search engine dependent.
Post the search term construction the terms are submitted to a search engine as to obtain hopefully appropriate and accurate search results. We developed three interfaces to popular search engines, namely Google, Google scholar and Microsoft Bing, as to research how different search engines influence the results. Post the submittal of search terms to a search engine we obtain a set of search result pages. Some search engines have an API for this task (i.e. Bing). Unfortunately, Google does not at the time of writing. This meant that we had to parse the raw result page HTML in order to obtain the result data. This is a non-trivial and actually rather complicated software engineering task as there is a significant amount of result ‘garbage’ to weed through while parsing the results. We found that Google changes the HTML structure very frequently, which required code changes to accommodate for the changed format, just to complicate matter further.

Once the search results are extracted we extract the key properties Uniform Resource Locator, title and abstract to drive the corpus construction phase. Initially we constructed the corpus through a full download of the top five URL resources inclusive of extraction, parsing and text processing. A comprehensive document processing engine was build covering features such as download management and topic text extraction from HTML, Portable Document Format, Microsoft Word and various other document types. However, we established that the average time required for reading and processing full documents was too time-consuming. The entire process lagged so much that the method became of impractical use. But also testing required an entire night of test runs just to compute the statistical effect of even small change, and this severely slowed down the progress of our research. We decided to construct a corpus from the actual the search result data instead, that is, through the top 150 indexed documents titles and abstracts. The corpus construction easily ran a factor thousand faster using this approach. The obvious downside being that the quality of the corpus is diminished in terms of scope, breadth and depth. Bear in mind that summaries are computed by the search engine, which might use HTML Meta tags but could also employ NLP related techniques for this function.

With the corpus constructed we can start to relate the corpus terminology to the property terminology in order to establish a property ranking. The overall idea is that if the corpus refers more frequently to some property $p$ than to some other property $q$, than $p$ is a more important property than $q$, and hence, should rank higher from the perspective of a casual data observer. The rank computation (which is derived from a computed corpus score) is somewhat complex and is explained in more detail next. To do so, we must first define more rigorously what we mean with a ‘corpus reference’. Obviously, if a property $p$ has a label $l$, and we find that this label is directly mentioned in the corpus, than this is indeed a very strong linkage. Weaker references (secondary/ternary linkage) are established when a derivative form of $l$ is found in the corpus, such as derived keywords, concepts, synonyms, derived words or hypernyms.
We will use the next section to explain with more detail how we computed the corpus scores. For any DBpedia class $C$ for which we want to rank the properties, the ranking algorithm iterates over all the properties from $C$. For any property $p$ from $C$ that we find we test if we can establish a corpus reference. Since a direct label match is the strongest linkage, any such corpus match is fully accounted as the relevance of that reference is 100%.

The indirect corpus references are based on derived terminology that we introduced on page 47 (keywords, concepts and categories). This type of derived terminology has an Alchemy API assigned confidence score. These confidence levels are included as a factor in the calculation of the corpus score. Derived terminology with a confidence level under 0.60 is not considered. WordNet does not offer confidence levels, and hence, are fixed at a constant value (1.0). If we establish a derived corpus match then we reduce the value of that match in order to correct for the level of uncertainty by using a relevance offsetting factor. The corpus score computation process is outlined below:

- Is the label of $p$ mentioned fully in the corpus? If so, we rate each such corpus mention (aka 'hit') at relevance 1.0.
- Are any of the associated Alchemy API keywords for $p$ found in the corpus? If so, hits are scored at relevance 0.80.
- Are associated Alchemy API taxonomy categories for $p$ found in the corpus? If so, hits are scored at relevance 0.60.
- Are any of the associated Alchemy API concepts for $p$ found in the corpus? If so, hits are scored at relevance 0.40.
- Are any of the associated Alchemy API categories for $p$ found in the corpus? If so, hits are scored at relevance 0.20.
All these scores are summarized in order find a final corpus score for $p$. We now give a small example below; assume that property $p$ has:

- A label $l$ which is found 3 times in the corpus.
  
  Corpus score = $3 \times 1.0 \times 1.0 = 3.0$.

- A keyword $k^1$ with confidence 0.9, found 2 times in the corpus.
  
  Corpus score = $2 \times 0.9 \times 0.80 = 1.44$.

- A keyword $k^2$ with confidence 0.7, found 4 times in the corpus.
  
  Corpus score = $4 \times 0.7 \times 0.80 = 2.24$.

- A concept $c^1$ with confidence 0.85, found 5 times in the corpus.
  
  Corpus score = $5 \times 0.85 \times 0.40 = 1.70$.

The total corpus score for $p$ is 8.38. We can then derive the final ranking by ordering the properties of $C$ on their corpus score, with the highest scoring properties to come first.

4.5.1 Result discussion

Google, nor any of the other search engines that we tested, generated repeatable results for the same search query, which hindered the evaluation process. The exact same search query yielded completely different results over consecutive test runs, with the possible exception of the top five or so results. The top results would also differ on a day-to-day basis, but even near sequential queries in the same browsing session presented different top-ten results in Google. Hence, two experiment runs to rank an identical DBpedia class showed different ranking results. As we used a large set of test cases in our evaluation we consider the obtained performance metrics as reliable indicators.

We did not establish major differences between the various search engines during our initial research. However, scraping results from Google proofed very difficult for longer session runs as they have very aggressive ‘anti-robot’ measures that are extremely difficult to counter. As the Microsoft Bing API was more convenient to use we opted to optimize and refine our methods using Bing in the final experiment rounds.
We discovered through the empiric phase and algorithm evaluation that the quality of resultant corpus is linked to quality of the search results. The search results quality in turn depends on the quality of the search terms, and the ability of a search engine to serve relevant results. With so many variables into play, controlling the quality of a dynamically constructed corpus turned out to be extremely difficult.

Although the overall concept for this ranking approach is promising on paper, the observed performance is mediocre at best, as can be seen in Figure 41. In fact, this method can barely outperform alphabetical ranking. There is no statistical significant difference with alphabetical ranking. One of the root causes is the frequent inclusion of corpus text that is irrelevant with respect to the topic being ranked.

A final consideration that we want to bring forward is the codebase extensiveness, in terms of lines of code and the involved logic complexity. The implementation required a lot of work and tweaking. We doubt if Semantic Web browser tool developers will invest so much time and effort in a subsystem to rank properties. This is not a research question as such, but we did feel that this ranking approach became over-academic and had little practical applications.

Figure 41: KPI metrics for dynamic terminological ranking

We saw no significant result differences with respect to the search engine that we used.
4.6 Ontology structure analysis ranking: heuristic approach

Heuristic (or ‘rules-of-thumb’)-based ranking is an approach where a set of simple but efficient processing rules are applied to compute the rank positions. Earlier research [10] applied a heuristics-based approach to rank RDF instances. The work from Waitelonis et al utilized specific pre-knowledge of the DBpedia ontology to tune their algorithms. Our research applies heuristics to rank class properties solely based on ontology structural data and we generalized the rule set so that no optimization is required towards any particular ontology.

The results were obtained through an exhaustive range of development iterations to test heuristically-based ranking improvement ideas. The results of each development cycle were reviewed using the evaluation model metrics. Over time the ranking matured, up to the point where we could no longer identify any further optimizations. We will explain how our approach is implemented in the following paragraphs.

The underpinning mechanism in the method is to categorize all properties into ‘priority’ buckets based on specific property characteristics. The properties in bucket A come first, with properties from bucket B following and so forth. Each bucket can contain further levels for secondary or even ternary ordering based on relevant property characteristics. We will now describe how we promote properties to specific buckets. Please note that the bucket prioritization rules are evaluated in the exact order given below. Furthermore, if we at any point assign a bucket for a property, then that assignment is final.

1. Any property which contains one of the following strings in its label is promoted to bucket A:
   - ‘name’
   - ‘code’
   - ‘identifying’
   - ‘identifier’
   - ‘label’

2. Any properties from the following ontologies are promoted to bucket E:
   - foaf:name
   - foaf:nick
   - foaf:givenName
   - foaf:givenname
   - foaf:familyName
   - foaf:family_name
   - foaf:firstName
   - foaf:lastName
3. Properties from the following ontologies will be promoted to bucket $F$:

- foaf:surname
- foaf:topic
- http://schema.org/alternateName
- http://schema.org/name
- dc:title
- dc:subject
- dc:identifier

4. If the property is any of the following XSD date data types then bucket $B$ is assigned:

- XSDdate
- XSDdateTime
- XSDtime
- XSDgYearMonth
- XSDgYear
- XSDgMonthDay
- XSDgMonth
- XSDgDay
5. We then look at the range of the property and compute a metric that we call the range rank.

A zero range rank is assigned to most of the default XSD data types:

- XSDbyte
- XSDdouble
- XSDfloat
- XSDdecimal
- XSDboolean
- XSDint
- XSDshort
- XSDstring
- XSDbyte
- XSDunsignedByte
- XSDunsignedInt
- XSDunsignedLong
- XSDunsignedShort
- XSDnonNegativeInteger
- XSDnegativeInteger
- XSDnonPositiveInteger
- XSDpositiveInteger
- XSDduration

For ranges that are object types (rather than data types) we determine the rank with respect to its usage throughout the entire ontology. To do so, we first compute the most popular ranges within the ontology. We do this by iterating over all properties that exist in the given ontology, counting each occurrence of that class as a range of a property. This procedure yields an ordered list of classes, ranked on 'popularity', meaning that as classes are used more frequently as a range, they will rank higher. We denote this list as \( L \).

We can now compute the range rank as follows. For a property \( p \) that is being ranked we examine the range of \( p \) which we denote as \( R \). From \( R \) we determine all super classes as to obtain the entire class hierarchy \( H \). We then iterate over the classes in \( L \), in order of most popular to least popular. For each class that we visit, we test if it matches any of the classes in the hierarchy \( H \). If there is a match, then we assign the position of the match to the range rank.

If a range rank has been assigned than the property is promoted to bucket \( C \). The actual range rank becomes the immediate next level of ordering within bucket \( C \) (popular ranges are ranked higher).
An interesting observation to make at this point is that we also tested an algorithm where we computed a range popularity list based solely on the class being ranked (rather than the entire ontology). This resulted in a less precise ranking performance, roughly four basis points on Kendall τ and related KPI’s. The advantage of this approach is that it does not require a full structural ontology analysis, and hence, is faster to compute.

6. Properties that are either functional or inverse functional are assigned to bucket $D$.

   Note: We haven’t found any positive effects from promoting symmetric or reflexive properties to high ranking buckets, but this is something that may be researched further in future work.

7. Any other properties are assigned to bucket $Z$ (these will be the lowest ranking properties).

Once the main bucket for a property is determined much of the heuristic ranking work is done. To improve the ranking results further we examine the properties in each bucket and create an additional intra-bucket ranking level from information within the property label that hint towards time ordering. Properties such as “start date” should obviously be ranked before properties such as “end date”. To create this distinction we create three additional sub-levels within each bucket:

- The highest priority ($a$) is assigned to properties that have labels containing any of the following words: “start”, “begin”, “birth”, “former”, “previous”, “opening” or “predecessor”.

- The second priority ($b$) is set for those properties where the label contains either “stop”, “end”, “death”, “future”, “subsequent”, “closing” or “successor”.

- The lowest priority ($c$) is designated to all properties that do not fall in one of the first two categories.

The effect of this step is that we can rank each bucket further based on time frame information extracted from property labels. This is an important step as it establishes a logical ordering between related properties whenever there is a clear ‘first’ and ‘second’ property. Obviously, we can extend the word lists above significantly, but even this limited list of terminology already performs rather well. Finally, all levels obtain a last ordering step that is based on term frequency. We rank two properties that have the exact same bucket and secondary / ternary level using the respective term frequencies with more frequent terms to come first. In case the frequencies are identical we apply alphabetical sorting to resolve any ties.
We also experimented with so-called ‘out & in’ properties to see if we could find a means of ranking such properties in a meaningful and ranking-effective manner. Out & in properties have ranges that have a back-reference to the originating class. More formally, an out & in property \( p \) of class \( P \) has a range \( R \) where \( R \) has at least one property \( r \) with range either \( P \) or some superclass of \( P \). In short, \( p \) manifests a circular route: \( P \rightarrow R \rightarrow P \). Such properties exist in our dataset, and Waitelonis et al. already described the incorporation of such properties in their ranking mechanisms. However, we could (somewhat surprisingly) not establish a ranking that considered out & in properties which outperformed approaches that did not consider these properties.

4.6.1 Result discussion

In Figure 42 we have graphed the computed ranking performance metrics.

![Figure 42: KPI metrics for heuristics-based ranking](image)

The concordance metrics are up to 14% better than alphabetic ranking as can be seen from Figure 43. Precision is very high and recall outperforms alphabetical ranking with at least 10% at any level. Proximity is somewhat better than alphabetical ranking, but it’s not as impressive as some of the other KPI’s. Group stability is on par with alphabetical ranking.

![Figure 43: KPI improvements for heuristic ranking over alphabetic ranking](image)
Clearly, the ranking performance is good in terms of ranking accuracy, but also in terms of performance, ease of implementation, and in addition, the method does not depend on any external features. In fact, heuristic ranking outperforms any other ranking algorithm examined in this research.

The outperformance of heuristic ranking with respect to alphabetic ranking, based on a paired samples t-test computed both for the normalized Kendall τ and the normalized Spearman ρ metric, is statistically significant. The computed results are shown in Figure 44, with a supporting box plot shown in Figure 45 that shows appropriate normally distributed difference measures to support the t-test. The t-test data for the normalized Spearman ρ metric are not shown here.

![Figure 44: paired samples t-test for heuristic ranking](image1)

![Figure 45: normalized Kendall τ differences distribution for heuristic ranking](image2)
4.7 Multi-lingual evaluation for ranking algorithms

In the research described so far we used English Infobox ranking data exclusively as the baseline against which we conducted our evaluation of ranking algorithms. This was done in line with our research goal to review ranking algorithms within the scope of the English language (as stated in paragraph 1.4). The process by which we obtained the English ranking data baseline was explained fully in section Chapter 3.

As our research progressed we felt that a review of ranking algorithms would be stronger if we would also incorporate ranking baselines from non-English Infobox templates. Infobox templates vary per language, and the ranking order that can be extracted from these templates is indeed different to some degree. Ranking algorithms that consistently produces strong ranking results ‘regardless’ of the ranking baseline in use are obviously far more useful that those that cannot.

Before we continue our discussion on the multi-lingual evaluation aspects we want to highlight that our ranking algorithms internally operate only in “English mode”. The multi-lingual experiments take the ranking output from these algorithms and compare these rank orders with non-English ranking baselines. To illustrate this aspect in a somewhat more concrete manner, take for example our heuristic algorithm that we described in section 4.6. The algorithm will at some point search for specific English phrases to improve the rank order for properties. This algorithm was thus not re-written for this experiment to search for other (non-English) phrases. Hence, we used the exact same algorithms for our multi-lingual experiments. The word frequency lists that we used, the WordNet database and the NLP API interaction all remained ‘as is’ (English). The input fed into the ranking algorithms also indeed all English.

There were some data set factors that influenced the design of our cross-language ranking baseline experiments. First, one has to consider that two given Wikipedia languages $L_1$ and $L_2$ will typically not share the same set of Infobox templates. As an example, consider the fact that the English DBpedia at time of writing defines 417 mappings where the German set mapped 353 classes. The number of classes that are mapped both in German and English constituted of a mere 85 classes. The number of usable classes for which we can actually compute rankings is even smaller as we exclude for example classes that have fewer than 8 properties. The number of shared classes diminishes further very quickly for each additional language included to the point that the statistical validity becomes questionable.
To circumvent this issue we designed a meaningful experiment in the following manner. We created a data set that consisted of 244 non English mapped Infobox templates. The set compromised of Infobox templates that were mapped by French, German, Spanish, Dutch, Polish, Portuguese and Turkish DBpedia mappings. Each included mapping at least maps 8 class properties. Please note that the construction of the test data was to a large extend already discussed in sections 3.3, 3.4 and 3.5.

We ran the different ranking algorithms repeatedly over all the classes and computed average KPI metric values. The computed average KPI values could then be compared and evaluated for significance using paired sample t-tests. The experiment design is shown schematically in Table 14.

<table>
<thead>
<tr>
<th>Alphabetic</th>
<th>Word frequency</th>
<th>N-gram</th>
<th>Heuristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class A (NL)</td>
<td>Class A (NL)</td>
<td>Class A (NL)</td>
<td>Class A (NL)</td>
</tr>
<tr>
<td>Class B (NL)</td>
<td>Class B (NL)</td>
<td>Class B (NL)</td>
<td>Class B (NL)</td>
</tr>
<tr>
<td>Class Q (DE)</td>
<td>Class Q (DE)</td>
<td>Class Q (DE)</td>
<td>Class Q (DE)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Class A (TR)</td>
<td>Class A (TR)</td>
<td>Class A (TR)</td>
<td>Class A (TR)</td>
</tr>
</tbody>
</table>

Compute average Compute average Compute average Compute average

Table 14: multi-lingual experiment approach

By including various languages in our test data set we obtain a blended average that covers a wide array of different ranking baselines. The experiment output is used to evaluate if the ranking algorithms reviewed in our research also outperform alphabetic ranking in a non-English context. The logic rationale is shown in Figure 46.

Figure 46: governing logic for multi-lingual experiment
We started with a run to review the alphabetic ranking algorithm in a multi-lingual context. The results of this run are shown in Figure 47. Please note however that the group stability metrics have been excluded in the multi-lingual metrics set as these are meaningless in this context.

![Figure 47: metrics for alphabetic ranking in a multi-lingual ranking baseline](image1)

We have included the English metrics again as a reference point in Figure 48 although we cannot compare the above figures directly due to the different input data set. Note that his graph is the same as was already presented in section 3.10.6. The two metric diagrams are very close in terms of the comparative KPI metrics.

![Figure 48: metrics for alphabetic ranking in an English ranking baseline](image2)

With the above reference in mind as a starting point we can discuss the performance of the three ranking algorithms that we researched in this paper. We only considered the ranking algorithms for which we already proofed that they outperformed alphabetical ranking when evaluated with respect to an English ranking baseline.
First, we discuss the word frequency ranking algorithm that we described fully in section 4.2. In Figure 49 the deltas for the KPI metrics with respect to alphabetical ranking are presented (please also refer to Figure 47). As all these metrics have been computed from the same multi-lingual ranking baseline data set we have paired data, which allowed us to compute a paired samples t-test. We do not present the t-test data here, but the outperformance is statistically significant (with a 95% confidence level).

The exact same procedure was executed to review the N-gram ranking algorithm with non-English ranking baselines. The N-gram ranking algorithm was fully described in section 4.3. Again, we only present the delta metrics, and note that the outperformance is statistically significant in a paired sample t-test for differences.
Finally, we present the performance metrics for the heuristic ranking algorithm in a multi-lingual ranking baseline context in Figure 51. The paired samples $t$-tests for both the normalized Kendall $\tau$ and normalized Spearman $\rho$ metrics show that the results are statistically significant. We have included the supporting data for the $t$-test (for the normalized Kendall $\tau$) in Figure 52 and Figure 53. The $t$-test data for the normalized Spearman $\rho$ metrics are not shown.

![Figure 51: multi-lingual heuristic ranking (Δ compared to alpha ranking)](image)

<table>
<thead>
<tr>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>Heuristic</td>
</tr>
<tr>
<td>Alphabetic</td>
</tr>
<tr>
<td>Difference</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Results for $t$-test:</th>
</tr>
</thead>
<tbody>
<tr>
<td>p-value</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>One Tail</td>
</tr>
<tr>
<td>Two Tail</td>
</tr>
</tbody>
</table>

![Figure 52: heuristic ranking $t$-test details (multi-lingual context)](image)

![Figure 53: differences distribution for heuristic ranking (multi-lingual context)](image)
Chapter 5  Research summary

5.1 Conclusions

This section presents our conclusions from the conducted research and experiments. We researched five proprietary property ranking algorithms in this study that utilize different approaches to ranking that vary from relatively simple term frequency counting up to complex ontology-based terminological analysis. Several ranking algorithms employed novel approaches to the ranking of properties from ontology class structures not studied before in this context. The conducted ranking algorithm experiments were described in full detail in this paper and we concluded each section with a comprehensive review of the respective ranking approach performance.

The performance characteristics of these ranking algorithms were measured objectively using an appropriate human designed ranking baseline extracted from Inbox templates. The usage of this ground truth set was defended in paragraph 2.4. We described how we created a test data set from this baseline for the purpose of ranking evaluation in Chapter 3.

A total of 19 ranking metrics were described in paragraphs 3.6, 3.7 and 3.8 that assisted us in measuring the performance of ranking algorithms using the test data set. The paired t-test used to compare ranking algorithms described in detail in paragraph 3.9 is a scientifically sound statistical approach. In Table 15 we present an outline of the overall results for the various conducted experiments. We have restricted the summary to the key rank correlation metrics as the evaluation across these dimensions was central in our research.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Kendall $\tau$</th>
<th>Spearman $\rho$</th>
<th>Significant with respect to alphabetical?</th>
<th>Cohen correlation classification for $\rho$ (rounded to 0.1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alphabetical</td>
<td>0.53</td>
<td>0.54</td>
<td>N/A</td>
<td>Small</td>
</tr>
<tr>
<td>Word-frequency</td>
<td>0.61</td>
<td>0.65</td>
<td>Yes</td>
<td>Moderate</td>
</tr>
<tr>
<td>Term similarity</td>
<td>0.51</td>
<td>0.51</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>N-gram</td>
<td>0.61</td>
<td>0.64</td>
<td>Yes</td>
<td>Moderate</td>
</tr>
<tr>
<td><strong>Heuristic</strong></td>
<td><strong>0.65</strong></td>
<td><strong>0.68</strong></td>
<td>Yes</td>
<td><strong>Moderate</strong></td>
</tr>
<tr>
<td>Dynamic terminological</td>
<td>0.55</td>
<td>0.55</td>
<td>No</td>
<td>Small</td>
</tr>
</tbody>
</table>

Table 15: comparison of ranking algorithms
A review of Table 15 shows that the largest rank correlation metric has a value of 0.68 (the average normalized Spearman $\rho$ metric for heuristic ranking). The regular, that is, the non-normalized average Spearman $\rho$ value is 0.36, computed using the simple conversion formula \((\rho_{\text{normal}} * 2) - 1\). Note that the regular Spearman $\rho$ metric has a range from -1 up to 1. The top boundary (1) denotes a perfect matching of ranks; the low boundary (-1) indicates complete reverse rankings. A score of 0 denotes a complete abundance of association, that is, rankings that share no commonality. A value between 0.30 and 0.50 is viewed by Cohen [40] as a “moderate” correlation. This classification from Cohen remains to hold even in the somewhat weaker case of the averaged Kendall $\tau$ score (0.65 normalized, 0.30 regular). Although the correlation is categorized as moderate, it must be noted that the amount of possible rank permutations is substantial. Hence, the term ‘moderate approximation’ must be viewed in perspective. We now have come to the point to derive a conclusive answer for the main research problem that we defined in paragraph 1.2:

**To what extent can an ontology property ranking algorithm approximate a human-designed ranking to support automated view generation in Semantic Web browsers?**

Based on our research the answer to the overall research question is that three of the studied ranking algorithms approximated a human designed ranking up to a moderate extent. Two of these methods are terminology-driven and intrinsically NLP-based. The discussed disambiguation issues are both challenging and limiting aspects of NLP-based approaches. However, these ranking algorithms are potentially applicable to domains other than the currently discussed Semantic Web context. Furthermore, we established that the three ranking approaches outperformed alphabetical ranking in a statistically significant manner (see also Table 15). The techniques surpass alphabetical ranking significantly based on the averaged Kendall $\tau$ metrics and the averaged Spearman $\rho$ metrics, both in English and in the multi-lingual evaluation context.

Rather than re-iterating over the individual experiment conclusions we give our view with respect to the most useful ranking based on the key lessons learned and the ranking algorithm evaluation data. The review of the key ranking algorithm performance metrics with respect to ranking ontology properties has led to the overall conclusion that the heuristics-based algorithm is the best performing approach to ranking that we studied. In addition to the key performance characteristics we also looked at secondary dimensions such as the ranking algorithm implementation complexity, the speed of ranking and the lack of external dependencies. A holistic evaluation leads to the conclusion that the heuristics-based ranking approach is clearly the overall “winning” ranking method. The heuristics-based approach does not require a large amount of code to implement and the involved logic is quite modest. Furthermore, ranking speed is fast and predictable.
5.2 Future work

Our research reviewed several ranking algorithms that we developed through an empiric and holistic approach. The algorithms were inspired partly by past work, partly by gaps in the literature and partly by evolving insights. Future work may utilize machine learning techniques to discover new approaches to ranking. In particular, we speculate that it may be possible to predict the rank of specific properties based on the rank of comparable properties in other classes. It might be possible to find property interrelationships that can ‘predict’ the ranking position of other properties. This type of research would reverse engineer ranking patterns using techniques such as data mining and possibly also feature learning to discover recurring ranking patterns. The heuristic ranking algorithm in particular will benefit from this type of research. Another interesting future work research direction would involve a further strengthening of the presented methods through inclusion of instance data sampling.

The evaluation framework, as well as the constructed test data set, may be used by future researchers to review alternative ranking algorithms. Our study results can act a solid foundation for future reference comparisons. The evaluation framework itself may be extended further with additional metrics to support the review of other aspects of rankings, as is seen fit. Future work may also involve a subjective (‘human’) review of our ranking algorithms to provide a different perspective on the performance of the studied ranking algorithms.

Another obvious way to further complement this research would involve the application of our algorithms within a semantic web browser to test-drive our algorithms in “real life” scenarios. An implementation based on the Fresnel Display Vocabulary for RDF53 is already being considered within the current institution’s research program.

Last, but not least, we wondered how the ranking information in Infoboxes evolved over time. Wikipedia is fully versioned which enables a thorough research of historical ranking information. Are properties stable over time in terms of their rank position? What types of properties are typically reordered? The central research question would involve the identification of patterns that can be utilized to improve the heuristic ranking approach. Data mining techniques may again prove useful for this type of research.

53 http://www.w3.org/2005/04/fresnel-info/
Chapter 6  Appendixes

6.1 Appendix 1: references

A. Balmin, V. Hristidis, and Y. Papakonstantinou, “Objectrank: Authority-based key-
word search in databases,” in Proceedings of the Thirtieth international conference on

D. A. Quan and R. Karger, “How to make a semantic web browser,” in Proceedingsof

Information Base for Domain Oriented Portal Solutions,” in Proc. of 15th Int. Conf. on

B. Bamba and S. Mukherjea, “Utilizing resource importance for ranking semantic web

Tiýek, Michal and Grlický, Vladimir, “Ontology-based Information Presentation,” Fac-
ulty of Informatics and Information Technologies, Slovak University of Technology in
Bratislava, Slovakia.

J. Hees, T. Roth-Berghofer, R. Biedert, B. Adrian, and A. Dengel, “BetterRelations:
Detailed Evaluation of a Game to Rate Linked Data Triples,” in First International
Workshop on Orderig and Reasoning, Koblenz, Germany, 2011.

heuristics with a quiz that cleans up dbpedia,” Interact. Technol. Smart Educ., vol. 8, no.

L. Wolf, M. Knuth, J. Osterhoff, and H. Sack, “RISQ! Renowned Individuals Semantic
Quiz: a Jeopardy like quiz game for ranking facts,” in Proceedings of the 7th Interna-

M. T. Pazienza and A. S. Noemi Scarpato, “Application of a Semantic Search Algorithm
to Semi-Automatic GUI Generation,” in Proceedings of the International Conference on


C. Brewster, H. Alani, S. Dasmahapatra, and Y. Wilks, “Data driven ontology evalua-

M. Rospocher, S. Tonelli, L. Serafini, and E. Pianta, “Corpus-based terminological

D. Rinser, D. Lange, and F. Naumann, “Cross-lingual entity matching and infobox

### 6.2 Appendix 2: typical search terms per research question

<table>
<thead>
<tr>
<th>Research question</th>
<th>Related search terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. What does “property ranking” exactly entail (for an entity, such as an ontology class), and which methods and techniques are described in scientific literature with respect to ranking of properties?</td>
<td>“(entity</td>
</tr>
<tr>
<td></td>
<td>Additional search terms are listed below to drive specific search focus for locating:</td>
</tr>
<tr>
<td></td>
<td>• Semantic Web research:</td>
</tr>
<tr>
<td></td>
<td>Specific extra keywords: “Semantic Web”, “rfds”, “rdfs”, “ontology”, “owl”, “class(es)”, “taxonomy”.</td>
</tr>
<tr>
<td></td>
<td>• Model Driven Architectures research:</td>
</tr>
<tr>
<td></td>
<td>Specific extra keywords: “mda”, “model”, “model-driven”, “model-based”</td>
</tr>
<tr>
<td></td>
<td>• XML schema processing research:</td>
</tr>
<tr>
<td></td>
<td>Specific extra keywords: “XML schema”, XMLS, “complex type”</td>
</tr>
<tr>
<td></td>
<td>• Database processing research:</td>
</tr>
<tr>
<td></td>
<td>Specific extra keywords: “Database”, “RDBMS”, “relational data”</td>
</tr>
<tr>
<td></td>
<td>In addition to the above key search terms, various searches were performed to find relevant papers that deal with Semantic Web browsing and visualization. Key search terms here were “Semantic Web (browsing</td>
</tr>
<tr>
<td>Question</td>
<td>Answer</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2. Which potential ranking techniques may proof useful to further research? Which potential ranking techniques may proof useful to further research?</td>
<td>Mostly overlapping with the terms mentioned above and below (although there is obviously a different context of interpretation).</td>
</tr>
</tbody>
</table>
Glossary

Alchemy API
A web-based (http://www.alchemyapi.com) multi-lingual NLP terminology processing engine with services accessible via an Application Programming Interface (API).

Attribute (object)
A named characteristic or property of an object.

Class
A class is a general specification or description for a set of objects and defines aspects such as common data structures and relationships.

DBpedia
A project that extracts structured content from Wikipedia and publishes that data available as Linked Data. DBpedia is one of the largest Linked Data hubs on the Web with millions of characterized entities.

Holonym
A linguistic term for the name of the whole of which the meronym is a part. Formally, Y is a holonym of X if X is a part of Y. As an example, Apple tree is a holonym of apple. See also meronyms.

Hypernym
A linguistic term for a word whose meaning includes the meanings of other words. Hypernyms (also called superordinates) are thus general words. In formal notation, Y is a hypernym of X if X is a (kind of) Y. For example, dog is a hypernym, while Collie and Chihuahua are more specific.
Hyponym
A linguistic term for subdivisions of more general words. Hyponyms designate members of a class. Formally speaking, X is a hyponym of Y if X is a (kind of) Y. As an example, a footstep is a kind of step, or, in more technical terms, footstep is a hyponym, or subtype, of step, and step is a hypernym, or supertype, of footstep.

Individual
See instance.

Inferencing
Inference is the act or process of deriving logical conclusions from premises known or assumed to be true. The logic within and between statements in an ontology is the basis for inferring new conclusions from it. An inference engine is also known as a reasoner. Inference commonly proceeds by forward chaining or backward chaining. See also reasoners.

Inheritance
Inheritance is a relationship between classes where one class is a parent of another and implements “is-a” relationships between classes.

Instance
Instances are basic “ground level” components of ontologies. An instance is an individual member of a class. Synonymous terms are entity, individual or member. The instances in ontologies may include real world objects such as people and planets as well as abstract instances such as numbers and words.

Linked data
Linked data is a set of best practices for publishing and deploying instance and class data using the RDF data model, and uses Uniform Resource Identifiers to name the data objects. The approach exposes the data for access via the HTTP protocol. There is great emphasis on data interconnections, interrelationships and context.

Member
See instance.
Meronym
A linguistic term for a word that denotes a constituent part or a member of something. Formally, \( X \) is a meronym of \( Y \) if \( X \) is a part of \( Y \). For example, apple is a meronym of apple tree.

Natural language processing (NLP)
NLP is the process of a computer extracting meaningful information from natural language input and/or producing natural language output. NLP is one method for assigning structured data characterizations to text content for use in semantic technologies. (Hand assignment is another method.) Some of the specific NLP techniques and applications relevant to semantic technologies include automatic summarization, co-reference resolution, machine translation, named entity recognition, relationship extraction, topic segmentation and recognition, word segmentation, and word sense disambiguation, among others.

Object
An object is an abstraction or simulation of physical things such as people and machines or intangible things such as events and processes that captures their characteristics and behavior.

Ontology
A data model that defines the types, properties and interrelationships of the entities that really or fundamentally exist for a particular domain in a closely resembling manner.

OWL
The Web Ontology Language (OWL) is a family of dialects designed for defining and instantiating formal Web ontologies. An OWL-based ontology may include descriptions of classes along with their related properties and instances.

Property (triples)
Properties are the ways in which classes and instances can be related to one another. They are also known as predicates. Properties are used to define attribute relations for instances. Properties are fundamentally relationships.

Property (object)
A property is a named characteristic or attribute of an object.
Resource Description Framework (RDF)

The Resource Description Framework (RDF) is a family of World Wide Web Consor-
tium (W3C) specifications. RDF was originally designed as a metadata model, but it
has come to be used as a general method of modeling information through a variety of
syntax formats. The RDF metadata model is based upon the idea of making state-
ments about resources in the form of subject-predicate-object expressions, called tri-
ipples in RDF terminology. The subject denotes the resource, and the predicate denotes
traits or aspects of the resource and expresses a relationship between the subject and
the object.

RDFS / RDF Schema

RDFS or RDF Schema is an extensible knowledge representation language providing
basic elements for the description of ontologies, otherwise called RDF vocabularies,
intended to structure RDF resources.

Reasoner

A semantic reasoner, also known as a reasoning engine, rules engine or simply a rea-
soner, is a piece of software able to infer logical consequences from a set of asserted
facts or axioms. Many reasoners use first-order predicate logic to perform reasoning.
See also inferencing.

Semantic Web

The Semantic Web is a collaborative movement led by the World Wide Web Consor-
tium (W3C) that promotes common formats for data on the World Wide Web. By
encouraging the inclusion of semantic content in web pages, the Semantic Web aims
at converting the current web of unstructured documents into a “web of data”. It
builds on the W3C’s Resource Description Framework (RDF).

Semantic Web browser

A browser used for navigating the Semantic Web (data). The Semantic Web architec-
ture does not involve HTML. Semantic Web browsers specialize in processing RDF
data from Web servers. A Semantic Web browser renders information that it can find
on the Semantic Web about a specific resource. The views may contain hyperlinks for
users to navigate between the found resources. Semantic Web browsers are also
known as hyperdata browsers.

Statement

A statement consists of a subject, a predicate and an object. Statements are also
known as S-P-O assertions. Statements are by definition the “facts” (or axioms) with-
in ontologies.
Subject
A subject is a reference (or definition) to a particular object, thing or topic, or groups of such items. Subjects are also often referred to as concepts or topics.

Superordinate
See hypernym.

Synset
A Wordnet synonym set, a set of words that are interchangeable in some context without changing the truth value of the preposition in which they are embedded.

Terminological
The system of terms belonging or peculiar to a science, art, or specialized subject; nomenclature; the science of terms. Example usage: “the terminology of botany”.

Triple
A basic statement in the RDF language that compromises a subject, a property and an object, with the subject and property (and object optionally) referenced through a Uniform Resource Identifier (URI). See also statement.

WordNet
WordNet is a lexical database for the English language. It groups English words into sets of synonyms called “synsets”, provides short, general definitions, and records the various semantic relations between these synonym sets. The purpose is twofold: to produce a combination of dictionary and thesaurus that is more intuitively usable, and to support automatic text analysis and artificial intelligence applications. The database and software tools can be downloaded and used freely. Multiple language versions exist, and WordNet is a frequent reference structure for semantic applications.