Integrating and Querying Semantic Annotations



Luying Chen
St. Catherines College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy
Trinity 2014

Abstract

Semantic annotations are crucial components in turning unstructured text into more meaningful and machine-understandable information. The acquisition of the mass of semantically-enriched information would allow applications that consume the information to gain wide benefits.

At present there are a plethora of commercial and open-source services or tools for enriching documents with semantic annotations. Since there has been limited effort to compare such annotators, this study first surveys and compares them in multiple dimensions, including the techniques, the coverage and the quality of annotations.

The overlap and the diversity in capabilities of annotators motivate the need of semantic annotation integration: middleware that produces a unified annotation with improved quality on top of diverse semantic annotators. The integration of semantic annotations leads to new challenges, both compared to usual data integration scenarios and to standard aggregation of machine learning tools. A set of approaches to these challenges are proposed that perform ontology-aware aggregation, adapting Maximum Entropy Markov models to the setting of ontology-based annotations. These approaches are further compared with the existing ontology-unaware supervised approaches, ontology-aware unsupervised methods and individual annotators, demonstrating their effectiveness by an overall improvement in all the testing scenarios. A middleware system – ROSeAnn and its corresponding APIs have been developed.

In addition, this study also concerns the availability and usability of semantic-rich data. Thus the second focus of this thesis aims to allow users to query text annotated with different annotators by using both explicit and implicit knowledge. We describe our first step towards this, a query language and a prototype system – QUASAR that provides a uniform way to query multiple facets of annotated documents. We will show how integrating semantic annotations and utilizing external knowledge help in increasing the quality of query answers over annotated documents.

Acknowledgements

It is a great honor to study at Oxford University. It's been a long road, but here I am at the end.

First and foremost, I would like to express my gratitude to my superviser, Prof. Michael Benedikt, for his encouragement, patience and supervision during my DPhil study. His deep insights, broad knowledge and valuable suggestions make my research work always on the right track. Without his guidance and persistent help both in research and the procedure of thesis revision, this thesis would not have been possible.

I am also grateful to Dr. Evgeny Kharlamov and Dr. Giorgio Orsi for their great mentorship and valuable assistance in bringing out our projects. I have thoroughly enjoyed working with Stefano Ortona in the ROSeAnn project and I appreciate his invaluable contributions and kind support. Hugh thanks have to go to Dr. Xiaonan Guo for all the proof reading, cups of tea and the friendship extended to me since the day I started back in 2009.

Thank you to the Department of Computer Science and to St Catherine's Colledge for all the support during my study. Within the Department of Computer Science, I would also like to thank Dr. Tim Furche and Dr. Dan Olteanu for helpful discussions and great suggestions for improving both my thesis and my research. In addition, I would like to give my special thanks to Prof. Luke Ong and Mrs. Julie Sheppard for all their help and support.

Thanks to my colleagues: Dr. Clemens Ley, Dr. Huy Vu, Dr. Cristian Riveros and Ben Spencer, for interesting discussions and sharing ideas. To my best friend Vinty Huang, thanks for lots of emotional support and encouragement over the years. Thanks to my good friends accompanying me on this journey, Dr. Wenjia Bai, Dr. Po Wang, Mengying Xie, Dr. Fan Xu, Min Wu and many others for enriching my colorful life in Oxford.

I would like to acknowledge the financial support from the China Scholarship Council.

I dedicate this thesis to my family. As without the love and support from my husband Nathan, from our lovely daughter, Annie, and from my parents I would not be where I am today.

Contents

1 Introduction **1**

1.1 Background 1

1.1.1 Semantic annotations 2

1.1.2 Approaches to semantic annotation 4

1.1.3 Semantic annotation systems 7

1.2 Motivation 7

1.3 Contributions and originality 10

1.4 Organization 12

2 Benchmarking the Accuracy of Annotators **13**

2.1 Semantic annotators 13

2.2 Vocabulary alignment 19

2.3 Evaluation 22

2.3.1 Dataset 22

2.3.2 Evaluation measurement 23

2.3.3 Evaluation results 24

2.4 Conflicts of individual annotators 29

2.5 Summary 30

3 Reconciling Semantic Annotations **32**

3.1 Motivation 32

3.2 Aggregation algorithms 34

3.2.1 Preliminaries and assumptions 35

3.2.2 Supervised aggregation algorithms 36

3.2.3 Unsupervised aggregation algorithms 45

3.3 Evaluation 47

3.3.1 Individual vs. aggregated 47

3.3.2 Comparative evaluation 55

3.3.3 Performance and scalability 55

3.4 Alternatives to MEMM . . . 61
3.4.1 MEMM with multiple answers . . . 62
3.4.2 MEMM with quantitative information . . . 64
3.4.3 MEMM using additional features . . . 66
3.4.4 MEMM using hybrid models . . . 67
3.5 The ROSeAnn system . . . 69
3.5.1 System architecture . . . 70
3.5.2 The ROSeAnn APIs . . . 71
3.5.2.1 The ROSeAnn JAVA APIs . . . 71
3.5.2.2 The ROSeAnn web APIs . . . 73
3.5.3 User interface . . . 73
3.6 Summary . . . 75
3.7 Related work . . . 76

4 Managing and Querying Semantic Annotations 79
4.1 Motivation . . . 79
4.2 Data model . . . 81
4.2.1 The Quasar model . . . 82
4.2.2 Preliminaries and assumptions . . . 84
4.3 The Quasar language . . . 85
4.3.1 Syntax . . . 85
4.3.2 Semantics . . . 86
4.3.3 Example queries . . . 94
4.4 The Quasar system . . . 98
4.4.1 System architecture . . . 98
4.4.2 Implementation . . . 100
4.4.3 The user Interface . . . 103
4.5 Summary . . . 105
4.6 Related work . . . 106

5 Conclusions and Future work 107

References 111

Chapter 1

Introduction

This chapter will introduce semantic annotations, which will be the subject of the entire thesis. We first introduce the background of semantic extraction, then describe the motivations of this study, and in the end summarize the contributions and organization of this thesis.

1.1 Background

With the explosive growth of the World Wide Web, there is a vast wealth of information in semi-structured and unstructured documents (e.g., web pages and full-text documents) available on the Web. Efforts to *automatically* access and distil the information from these sources have thus been ongoing for the past decade. This area is known as Information Extraction (IE). In contrast to Information Retrieval (IR), whose task is to find out and rank relevant documents according to queries representing a user's information need, a broad goal of IE systems is to extract structured data from semi-structured and/or unstructured machine-readable documents, where the specific type and structure of the information to be extracted depend on the need of the particular application.

Web IE tools, also referred to as wrapper induction systems, will not be covered in this study. These systems (e.g. LIXTO [6], ROADRUNNER [17] and WEBTABLE [93]) perform IE on semi-structured online documents such as HTML pages, primarily using the layout structure of the documents. In contrast, the IE tasks we are concentrating on are those which process free text in natural language, since this "unstructured" data is the primary source of human-generated information [78].

To illustrate the distinction between wrapper-induction based and text-based IE, consider the scenario of extracting relational data from the web sites of real estate agents in London (see Figure 1.1). Figure 1.2(a) shows the resulting tuples obtained from the web pages by the wrapper LIXTO, where each record describes the attributes of an individual property such as address, postcode and price. However, there is a *Description* field in the

form of plain text, which includes additional interesting facts. These facts (e.g. "River Thames" is a river and is close to "Timber Pond Road"), as highlighted in Figure 1.2(b), need to be extracted by text-based IE, since a traditional web wrapper is unable to do so.

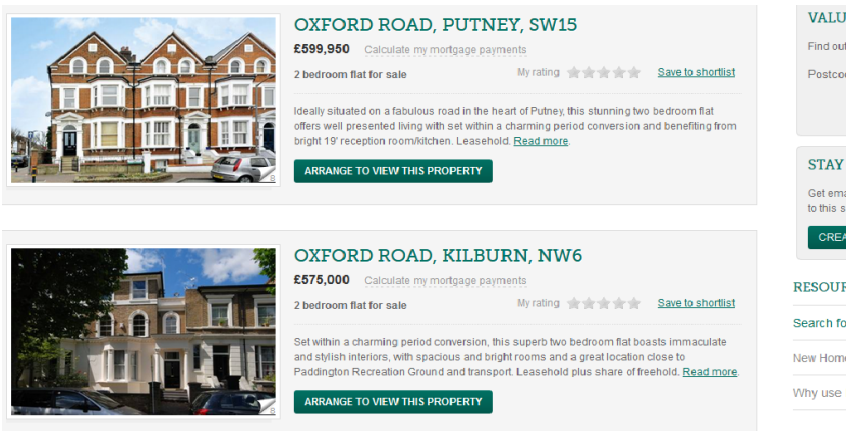
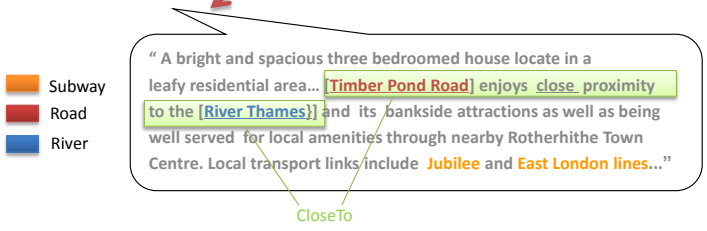


Figure 1.1: An example of a website of London real estate agent

ID	Address	Postcode	Price	Bedroom	Bathroom	Description	Date
1	Oxford Road, Putney	SW15	£599,950	2	N/A	Ideally situated on a fabulous ...	07/04/2014
2	Timber Pond Road, Rotherhithe	SE16	£375,000	3	N/A	A bright and spacious three house...	08/04/2014
...	...	...	...	...	...	...	...

(a) Structural data extracted by Lixto



(b) Text IE on the free text content in the column *Description*

Figure 1.2: Example result from Web IE and text IE

1.1.1 Semantic annotations

Information extraction on text, as we can see in the previous example, extracts factual metadata that conveys the information of the original text. The extracted metadata is assigned to the text, presenting part of the text information content for further processing. This task was initially explored by the Natural Language Processing (NLP) community,

and now has been extensively studied in research communities including IR and web mining. IE has a wide range of applications in domains such as biomedical informatics [34], business intelligence in finance and E-commerce [59, 40], serving as a bridge between raw data and knowledge.

In a nutshell, we treat the metadata extracted by IE tasks for understanding the raw text as the *semantic annotation*, and the text analysis programs tackling IE tasks as *semantic annotators*. To the best of our knowledge, there is no well established definition to determine whether an annotation is "semantic" or not. In this study, the term "semantic" covers a broader scope than what it usually does in the context of NLP, which is not only the meaning within sentences but also the meaning of the text beyond the sentence level, such as the tone and topic. For example, the output from the following text analysis tasks with different purposes are all considered as semantic annotations:

- *Named entity extraction* (NER for short): recognize and classify atomic snippets in text as referring to particular semantic categories (e.g., Person and Company) [18]. In some cases, it also involves a subtask named *entity disambiguation*, which resolves entity identifiers of the entity instance detected [52];
- *Relation extraction*: extract relationships between entities [73];
- *Sentiment annotation*: identify viewpoints or emotions in the underlying text. The attitude or tone can be polarity opinions such as Positive/Negative or comparison opinions such as Better/Worse or simply Subjective/objective [36];
- *Topic annotation*: attach documents with topics or themes [83];
- *Coreference annotation*: catch co-referring mentions of a given entity [89];
- *Keyword annotation*: extract relevant keywords or terminology to the content of the text [54].

Although there are different IE tasks, there is one fundamental research topic – named entity extraction, which by far is the most crucial and mature task. NER has been involved in many applications in addition to being a built-in component for IE. In question answering [85], for example, candidate answer strings are often named entities that must be detected and classified in advance. In entity-oriented search [13, 39], identifying named entities in documents as well as in queries is the first step towards obtaining high relevance of search results.

The development of IE has close relations with benchmarks including the Message Understanding Conference (MUC) [64], the Conference on Natural Language Learning

(CoNLL) [88] and the Automatic Content Extraction (ACE) program [22]. They focus on different IE tasks. For example, ACE focuses on entity, relation and event extraction while CoNLL simply works on entity extraction.

1.1.2 Approaches to semantic annotation

In the rest of this section, we will review IE technologies, in particular work on the NER subtask. A conventional IE system works as a pipeline which is composed of a number of NLP components, such as lexical processing (e.g. tokenization) and syntactic analysis (e.g. Part-of-Speech tagging). Each analysis pipes the output to the next analysis stage which will be consumed as base features. The core upper-layer analysis of NER can be divided into three major categories: *hard-coded methods*, *machine learning methods* and *hybrid methods*.

Hard-coded methods. These methods include:

- *Gazetteer annotators.* These annotators annotate a arbitrary text string by simply finding a match from gazetteers or dictionaries, where the gazetteers are maintained as a set of static files with lists of NE instances [84]. The matching process can be implemented by methods such as hash-based matching and finite state machines.

  In particular, gazetteer annotators manifest their superiority when dealing with some "rare" and static concepts such as Country and Holiday. However, gazetteer annotators suffer from several inherent limitations:(i) it is time-consuming to manually enumerate entity items for large volumes of lists; (ii) it can be an issue for precision because utilizing a dictionary intuitively can force the recognition of certain strings to be specific NEs regardless of the ambiguity of the text to be annotated (e.g., "Nottingham" can be a City or a SportTeam); (iii) it can also be problematic for the recall of those NE types which cover a broader domain of instances (e.g., names of people who are not celebrities).

- *Regular expression annotators.* These annotators apply regular expressions to capture the patterns of character strings. Thus they are particularly useful if those entities instances share a common pattern, such as well-formed email addresses, phone numbers and URLs.

- *Complex rule-based annotators.* Such annotators consist of a set of more complex hand-crafted rules such as GATE [18]. It composes annotations based on those base annotations produced by low layer extractors. The composition rules implicitly construct a "bottom-up" hierarchy among annotators. Composed annotators consume the text along with the annotations attached by base annotators.

Since each annotator in the hierarchy can be treated as a fine-grained "blackbox", it is: (i) easy to construct and maintain in many cases; (ii) easy to reuse them to make more complicated annotators. However, it is obvious that the quality of rule-based annotators depends on the proficiency of the rule designer. Moreover, if the manual rules are highly specific, then precision will be high but recall might be poor.

Machine learning methods. More recent work on NER uses statistical machine learning methods, which have become a popular alternative due to their capability to automatically infer patterns hidden behind large volumes of data. They can further be categorized into supervised, semi-supervised and unsupervised.

- *Supervised annotators.* These approaches address the NER problem by learning disambiguation rules from labelled examples based on discriminative features (e.g. the linguistic analysing result from the IE pipeline). Hidden Markov models (HMM) are an early attempt to address NER [99]. Other well-known learning models such as Maximum Entropy models [14], Maximum Entropy Markov models [60, 19], Support Vector Machines [24] and Conditional Random Fields [51] have also been widely applied as NER solutions.

  Supervised methods can work well when the training data is plentiful and easy to create. In addition, they are more likely to capture complex patterns which may be difficult to observe or difficult to encode with hand-crafted rules. Nevertheless, one major disadvantage is the reliance on human annotation of a training set, especially when porting the system to a new domain.

- *Unsupervised annotators.* For these annotators:(i) the vocabulary of target NE types (or favoured predicates) can either be input by users (as in [26]) or taken from external resources. For example, [2] makes use of the ontology of WordNet [63]; (ii) with regard to the NE identification process, one common way is to obtain "supervision" from some external sources. For example, a corpus can be retrieved by applying keyword queries to search engines. Keywords can be determined by heuristic patterns or terms from a synset in WordNet, etc., where a synset is a set of synonyms for a concept.

  Unsupervised methods save the cost of labelling large amounts of training data. However, they rely on external resources which may not be so "well-tailored" to support the learning process. This tends to make these approaches lose accuracy.

- *Weakly supervised annotators.* In order to bridge the gap between the time-consuming manual labelling required by supervised learners and the possible degradation of accuracy in unsupervised systems, the technology of "bootstrapping" is used where both labelled data and unlabelled data are used to build predictors [97].

  Although weakly supervised methods can create more accurate classifiers from less training data, the main disadvantage is that early prediction mistakes in the "bootstrapping" procedure could reinforce themselves during subsequent iterations. Further strategies may be required to control the noise brought by non-original labels (generated in bootstrapping), such as defining confidence and thresholds to filter unreliable labels.

Hybrid methods. The goal of hybrid-based methods is to combine both handcrafted-based and ML-based methods. [86] builds a NER model which incorporates rule-based methods, HMMs and Maximum Entropy methods. [95, 32] combine several comparatively strong individual ML-based classifiers such as a bootstrapping-based approach and SVM in different ways. For example, they train multiple classifiers sequentially by making each successive classifier consume the results of the previous one, taking advantage of the strongest point of each method.

Intuitively, hybrid methods tend to obtain more "convincing" results especially when an outstanding annotator is currently not available. This is at a cost of longer computation, compared with the case in which only one classifier is involved. The issues of choosing appropriate base-level classification approaches and how to combine them optimally may need to be concerned.

In addition to these prevalent methods, alternative solutions are emerging and contribute to modern IE. Efforts on investigating *Semantic Web-driven* (or *Knowledge-driven*) approaches [16, 15, 49] recently have been made by the inter-domain collaboration of researchers from the fields of IE and Semantic Web who turn their attention towards how Semantic Web technologies can benefit the traditional IE community. On the one hand, IE helps to mine knowledge from the text that enriches domain-specific repositories. On the other hand, the Semantic Web community makes these resources available, e.g., through Linked Open Data cloud (LOD) [55], which are ready to be utilized for IE. LOD describes a method to publish and share structured data, and interlinks the data from different sources [55]. In general, these approaches use unstructured, i.e., gazetted, or structured, i.e., ontological, background knowledge to locate the entities of interest in a text document. In detail, ontologies can be used to:(i) drive the construction of meaningful linguistic-patterns, as in [16, 46, 57]; (ii) refine the extraction rules, as in [15, 5],

or (iii) logically validate the result of the extraction, as in [7, 58] or, simply, (iv) link the extracted entities to a reference ontology, as in [61, 49].

Query language-driven approaches have also been explored to make use of declarative languages such as SQL and DATALOG. For example, [81] creates a DATALOG-based language – XLOG for relation extraction, where the body of each XLOG rule is embedded with procedural predicates or functions. [100] proposes to express relationships among entities by employing an extended-SQL query language, named *Content Query Language*, which allows users to specify context patterns as constraints. [41] uses the *Object Query Language* to represent rules with queries where annotations are treated as objects. [91] combines run-time IE and relational queries over a standard probabilistic database to query the most likely extractions.

1.1.3 Semantic annotation systems

Having examined the general methodologies for semantic annotation, we now turn to the state-of-the-art tools which provide the ability to semantically annotate documents. Most of the earlier IE systems are standalone software which is freely available. LINGPIPE NER [53] is a NLP library providing NER solutions using a HMM-based classifier; ANNIE [18] is a rule-based NER system which is embedded in the GATE framework; STANFORDNER [31] is a ML-based NER tool using CRFs; KNOWITALL [26] provides NER using an unsupervised learning approach; TEXTRUNNER [4] is an unsupervised IE system for relation extraction without any NE annotation. Systems such as LEARNING-PINOCCHIO [16] , CROSSMARC [46], SPOTLIGHT [61] and KIM [49] are all Semantic Web-driven NER tools. Examples of query language-driven systems include DOCQS [100] and AVATAR [41].

More recently, web services to semantically annotate documents have gained growing popularity. These services are characterized by their simplicity and suitability for the Web. Representative APIs include OPENCALAIS [66], EXTRACTIV [28], and ALCHEMY [1]. These web APIs usually perform online computation to generate useful annotations such as sentiment and topic annotations besides named entities. The web services could either be freely or commercially available. A more detailed introduction to a collection of selected systems will be presented in Chapter 2.

1.2 Motivation

The ecosystem of modern semantic annotators has been evolving dramatically. As mentioned in the end of Section 1.1, the appearance of web service APIs for online semantic extraction has drawn much attention. These APIs have their strengths and shortcomings,

but to the best of our knowledge there exist few comprehensive studies which investigate and compare these annotators on issues important to a semantic annotation system, such as accuracy and coverage.

Thus as the first part of this study, we benchmark the accuracy of these annotators in order to give a clear view of the state-of-the-art, focusing on web service APIs. The empirical evaluation results show that conflicting opinions of semantic annotations occur frequently among these annotators. In addition, these widely-used annotators suffer from serious accuracy deficiencies.

In order to make use of multiple annotators with varying annotation capabilities, we need to investigate integrating annotators, which leads to new challenges, both compared to usual data integration scenarios and to standard aggregation of machine learning tools. Take NER systems as an example. They initially focused on a few general classes of annotations (people, places), but nowadays annotators are becoming increasingly diverse. There are high- and low-level concepts that are supported by multiple annotators, in addition to a range of annotators with highly distinct vocabularies, but having many semantic interconnections. There are annotators that deal with very common entity classes, and those that focus on very specialized domains [8]. Although annotators aim to associate snippets with "semantic" labels, the meaning of annotations is often not formalised, and needs to be discovered. Annotators may use the same concept name with a different meaning (e.g. Organization is used with different meanings in OpenCalais and Extractiv) and can also use distinct concept names with the same meaning (e.g. OpenCalais:Movie vs. Spotlight:Film). Thus seeing the output of annotators as living in a single semantic space is not straightforward.

Furthermore, instead of being custom built "white-boxes" that can be directly tuned by developers such as GATE and UIMA [30], annotation is often available as a service (e.g. OpenCalais, Zemanta [98]), where the user does not have to implement his own annotator and concept classes can be added or refined by the service host transparently to clients.

The discussion above motivates the work in the second part of the thesis – namely, ROSeAnn (Reconciling Opinions of Semantic Annotators), which concerns an approach to address the problem of integrating opinions from diverse annotators with varying reliability. This approach to integration will be influenced by several distinctive aspects of semantic annotation. First, semantic annotations do not form a flat space of tags for document snippets, but rather associate snippets with a meaning, with the space of meanings being highly structured – two annotations with different concepts may thus still be compatible (e.g. if the concepts are in a subclass relation). In many cases, Web-based annotators provide mappings of instances and concepts to a publicly-available formal

ontology – for example SPOTLIGHT links to the well-known DBpedia ontology. Even in cases where such mappings are not provided, a relationship to a formal ontology can be inferred, and then utilized as a precise means of capturing compatibility or incompatibility of the judgements given by different annotators. This problem requires us to look for integration methods that are *ontology-aware* – making as much use as possible of semantic inter-relationships that are available. Another key factor is that many annotators are "black-boxes", making an approach based on hand-crafted "integration rules" problematic: a particular Web-based annotator may exhibit a certain pattern of errors at one point, and later change as the underlying rule engine is adapted. It thus requires to explore aggregation methods that are as *domain-independent* as possible, and do not rely on hand-rolled heuristics, focusing completely on the case of "black-box annotator aggregation".

Targeting the above problem, a set of supervised aggregation methods are proposed based on a Maximal Entropy Markov model, adapted to the setting of semantic annotation. An unsupervised approach based on logical constraints is overviewed as related joint work. These approaches are compared through quantitative experiments both with respect to each other and to individual annotators. The experimental results show clear benefits for aggregation of annotation, while also giving new insight into the behaviour of Web-based annotators.

In addition to the problems of accuracy and integration, there are also issues about usability and programmability on semantic annotations, in other words about how applications and users can easily obtain a collection of annotations they are interested in. As mentioned before, benefiting from the additional information from semantic annotations, many recent applications have been established on top of IE technologies, such as question-answering [85], entity retrieval [13, 9] and semantic-enhanced information retrieval [49, 45, 29]. Srihari et. al [85] make use of IE to populate semantic annotations (e.g. named entities and relation facts) and then focus on querying over the annotation collection, leaving out the original text. Entity retrieval systems extract named entities and provide queries for relevant entities combining the context keywords. Semantics-aware IR systems, such as [49, 45], work as follows: they take annotated documents as input, and search them on the presence of desired NEs and/or entity relations, without considering the document structure.

However, more advanced systems are required that are capable of integrating the above heterogeneous results of semantic extraction with diverse goals, and/or querying the filtering specification to take advantage of the multiple kinds of relationships within an annotated document. The relationships that one wants to explore should include:

Document structure: a query language should be able to ask for annotations that lie in a certain position within a document, within a certain paragraph, etc. E.g. "return all

annotations appearing in the first few paragraphs of the document".

Explicit annotation structure: an annotation may only identify an entity, or may distinguish the particular class or concept to which it belongs. An annotation may even identify the particular entity instance, relating it to a named entity in an ontology. An annotation may be a relationship or role, with its arguments likewise being known or unknown instances. A query language should be able to ask for all annotations given by a certain annotator, having a certain entity or relationship class, or containing a particular instance. E.g. "return all entities annotated by OpenCalais as a city".

Implicit knowledge: an important feature of annotations is that they have a well-defined semantics, encoded in the rules of an ontology. A query language should be able to make use of the ontology and exploit *implicit properties* of annotations accessible via reasoning, rather than syntactic matching. For example, one should be able to ask for all annotations of entities that the ontology infers are politicians – such a query would include entities labelled as presidents, senators, etc. One should be able to ask for all annotations with a person who is known to be married to a US politician – such a query would include all classifications returning Hilary Clinton, Michele Obama, etc. Implementing such queries requires integrating *reasoning* with structural querying.

Hence, the third part of this study concerns a querying system for semantically-annotated documents with the ability to filter query results based on the presence of annotations in diverse annotation vocabularies, and the relationships of those semantics to:**(i)** the text with the innate structure from which annotations emerge, **(ii)** the external knowledge resources such as the DBpedia ontology. We will describe QUASAR (QUerying Annotation, Structure And Reasoning), a system for structured querying of annotated documents that deals with all of the above structural aspects. We will show how such systems can allow users to specify their information needs with greater precision.

1.3 Contributions and originality

Contributions

1. By benchmarking the prevalent semantic annotators especially the web service APIs, we provided a clear view of the state-of-the-art focusing on critical aspects to a semantic annotation system such as accuracy and coverage.

2. Focusing on 11 widely-used semantic annotators over the NER task, a merged ontology was built manually based on the taxonomies of annotators which was able to capture the semantic interconnections between entity classes of various annotators. The aligned ontology now contains around 280 concepts with constraints

of *rdfs:subclassOf*, *owl:disjointWith* and *owl:equivalentClass* specified. In addition, quantitative evaluations were conducted for all the 11 individual annotators on four benchmark datasets, giving an overview of the performance variations among these individual annotators.

3. Aiming at integration over NE annotators, methods appropriate for an off-line scenario was proposed in which training data is available. They make use of ideas from supervised learning, and are able to capture global patterns of the behaviours of individual annotators. The experimental results show clear benefits for aggregation of annotations.

4. A system ROSeAnn was implemented for the management and the reconciliation of semantic annotations. ROSeAnn not only provides end-users and programmers with a unified view over the results of multiple online and standalone annotators, but also serves as a federated annotator that allows a uniform means to access diverse source semantic annotators. The corresponding JAVA APIs and web service APIs were developed as well for users and applications to integrate and utilize the functionality of ROSeAnn.

5. A query language was proposed allowing users to access documents with diverse semantic annotations and external ontologies. A prototype system QUASAR was implemented. QUASAR shows how integrating semantic annotations and utilizing external knowledge help in increasing the quality of query answers over annotated documents by both filtering out irrelevant answers and obtaining extra answers that are not explicitly available in annotated documents.

Statement of originality

The material of Chapter 2 and Chapter 3 (except Section 3.4 and Section 3.5) has been published by Luying Chen, Stefano Ortona, Giorgio Orsi and Michael Benedikt in VLDB2013 [11]. A demonstration paper [12] presenting the ROSeAnn system in Section 3.4 has been published by Luying Chen, Stefano Ortona, Giorgio Orsi and Michael Benedikt in VLDB2013. The JAVA APIs and web service APIs of ROSeAnn have been developed and released by Stefano Ortona, Luying Chen and Giorgio Orsi in WWW2014 [69]. The material of Section 3.5 is joint work with Michael Benedikt and has not been published so far. The material of Chapter 4 has been published by Luying Chen, Michael Benedikt and Evgeny Kharlamov in EDBT2012 as a demonstration paper [10].

1.4 Organization

In Chapter 2, the state-of-the-art semantic annotators are reviewed and benchmarked with quantitative evaluation over a set of real datasets. Chapter 3 concerns the annotation reconciliation system ROSEANN, which discusses the details of our core reconciliation algorithms and gives experimental evaluation which compares our algorithms against both individual annotators and existing aggregation systems. The chapter finally overviews our prototype system. In Chapter 4, the system for integrating and querying semantic annotations – QUASAR is proposed. We introduce the data model and the QUASAR language in both levels of syntax and semantics, describe the prototype architecture and its implementation and demonstrate the main GUI of the system. At the end of the thesis, conclusions and future work are discussed in Chapters 5.

Chapter 2

Benchmarking the Accuracy of Annotators

The state-of-the-art named entity annotators can be mainly categorized into software tools and web service APIs. Although these annotators have the same annotation goal, their annotation capability varies in many aspects such as the coverage of vocabularies and the quality of annotation results. In this chapter, we present an overall understanding of the ecosystem of the existing semantic annotators. This chapter first gives the background on annotators in general and then introduces the state-of-the-art named entity annotators along with their mapping to a global ontology. We then compare the accuracy of these annotators, making use of a scoring model which is tied to the ontology.

2.1 Semantic annotators

Recall from the introduction that *semantic annotators* usually refer to text analysis programs tackling text-based IE problems. In general, these programs take as input text documents and output semantic annotations in a variety of forms, transforming data into knowledge. The most common forms are *concept* labels (e.g., for named entity recognition which labels document snippets into pre-defined semantic categories), *entity* referencing (e.g., for named-entity disambiguation which associates snippets with unique entity identifiers) and *relation* (or role) labels (e.g., for relation extraction which captures the relationship between entities). Of these, the first is by far the most mature and fundamental. On the one hand, NER has been thriving for more than twenty years which is the most common task focused by IE benchmarks. On the other hand, NER is crucial to assist other IE tasks. For example, most solutions to entity disambiguation and relation extraction simply assume NEs (or at least NE spans) have been correctly detected. Hence, concept labelling is the focus of in this study.

An *annotation* consists of a contiguous segment of a document – namely, a *span* – along with a concept (also referred to as *class* or *entity type*: we use these interchangeably throughout the thesis), such as Person, Organization, or Artist. Figure 2.1 shows a real-life example of entity class annotations provided by multiple annotators. In this piece of newswire text, the snippet "Middlesbrough" is tagged with different opinions by individual annotators, e.g., SoccerClub by one annotator, and Person by another.

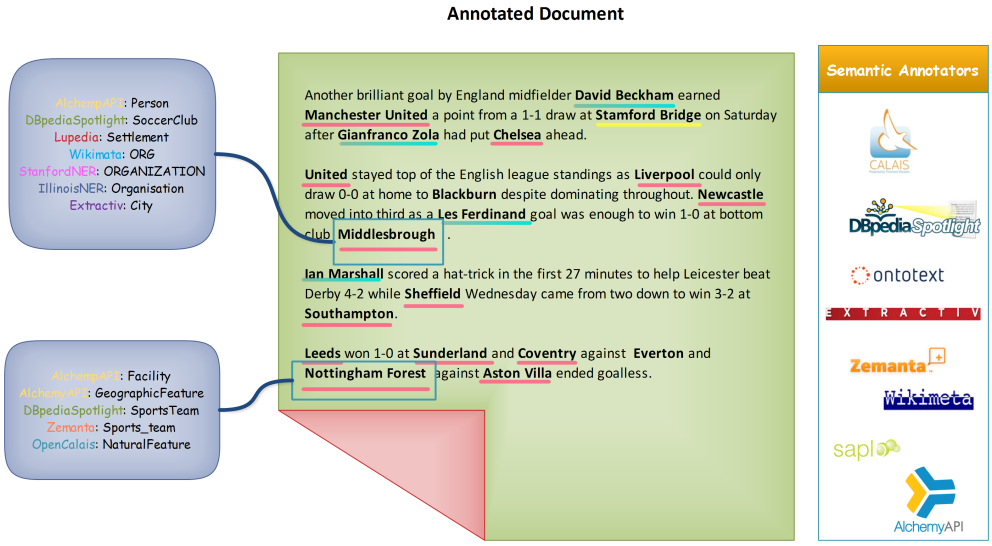


Figure 2.1: Example of semantic annotators

The state-of-the-art semantic annotators include software frameworks such as StanfordNer, IllinoisNer [72] and Gate, and web service APIs such as OpenCalais, Alchemy, Extractiv, Spotlight and Zemanta. The former ones are mostly provided as "white-boxes" by research communities, while the latter are often available as "black-box" services for commercial use. While originally focused on a few general entity classes by research-based systems, the vocabulary has been expanded in a wider range by commercial annotators with particular interest from the perspective of applications. Each annotator may focus on a particular subset of the concepts. For example, StanfordNer and IllinoisNer recognize only high-level concepts such as Person and Organisation, while Spotlight is able to support more specific concepts such as SoccerClub. With respect to the techniques utilized by these annotators, the extraction algorithms vary from simple gazetteers to complex hand-crafted rules. As mentioned in the introduction, there are also Machine Learning-based approaches, and even hybrid methods.

In order to give a deeper insight on the current ecosystem of semantic annotators, the first part of this study is to investigate the individual annotators. Before starting this, we describe the scenarios that are concerned, and the aspects of systems that we will

evaluate. We focus on three aspects of annotation systems: the supported languages, the domain, and the desired annotation types. For this study, we will only focus on systems dealing with English. In addition, systems which have been tailored to highly specific domains will not be covered. For instance, we will not consider systems (e.g., ABNER [80]) that are specially designed to annotate bioinformatics-related entities such as "protein", "DNA" and "RNA". In terms of the types of named entities, we consider a comparatively generic domain. In other words, we do not emphasize any particular vocabulary.

Given the above scope, a certain number of representative and prevalent annotators are targeted for the investigation: two software-based systems – STANFORDNER and ILLINOISNER, and nine web-based APIs: ALCHEMY, SPOTLIGHT, EXTRACTIV, LUPEDIA [56], SAPLO [77], OPENCALAIS, WIKIMETA [94], YAHOOQL [96] and ZEMANTA. Notice that Web-based annotators draw particular attention in this study. This is due to the fact that they pose specific challenges to integrators – the performance of the annotators and even the vocabularies may be highly dynamic. In the rest of this section, we introduce all the eleven tools above in detail. We overview the characteristics of the freely-available version of each annotator as of the time of the writing of this thesis.

ILLINOISNER is a standalone NER system developed by University of Illinois for research purposes. It utilizes hybrid approaches which inject gazetteer matches as features in machine-learning based methods. There are four pre-compiled models trained on the CoNLL03 dataset by utilizing different sets of features. The system recognizes classic entity types – Person, Organisation, Location and Misc. Given plain text documents as input, it returns annotated documents in which snippets are enclosed by NE tags.

STANFORDNER is a NE annotator from University of Stanford using the Conditional Random Field machine learning model. The software package provides four "already-baked" CRF models: the first two were trained on the mixed NE corpora of CoNLL03, MUC-6, MUC-7 and ACE in versions with and without additional Distributional Similarity Feature [31], while the other two were models trained on CoNLL03 training data. Users could either load one of the existing classifiers or customize a new one with other training corpus. The NER tagger with basic trained model is able to label PERSON, ORGANIZATION, and LOCATION entities. The input is a document and the output can be in formats of XML, inlineXML or slash-tags by default, where the slash-tags format assigns a label for each input token.

ALCHEMY is a web service for text analysis, based on NLP and ML techniques. Besides NE extraction, it offers overall 11 semantic extraction services such as sentiment annotation, keyword tagging, topic extraction, and relation extraction. ALCHEMY's named entity extraction supports 29 major classes of NE and hundreds of their subclasses. All the functions of ALCHEMY are accessible via a REST endpoint. The services are open

to input forms including URLs, plain text documents and HTML pages. The response can also be in various formats, such as XML and RDF. Instead of giving the exact span boundary of an entity, the output annotation is only returned with a *text anchor* that does not directly link to the original document. A free API key is limited to 100 calls per day.

Spotlight is an open source application which is able to recognize entities of DBpedia's concepts. In addition, it provides a disambiguation service for linking entity mentions in the text to DBpedia's URIs. A DBpedia's URI uniquely identifies an entity instance as a resource (e.g., the URI "http://dbpedia.org/resource/france" represents the country *France*). The system attempts to annotate an input document with DBpedia resources using at least 272 classes in the DBpedia ontology. Spotlight provides a free Scala/JAVA API and a REST endpoint, where a SPARQL query can be configured in the request to narrow down the output annotations. The Spotlight API accepts plain text and URLs as input and supports multiple output formats including XML, RDF and JSON. There is a size limitation for input text/html, for instance, the maximum size for a plain text input is 490KB.

Lupedia text enrichment service is a gazetteer-based entity extractor which uses the Large Knowledge Base(LKB) gazetteer to lookup words against DBpedia and Linked-MDB (Linked Movie Database) entities. The LKB gazetteer is provided by the Ontotext KIM team [47], consisting of a set of lists about entities such as persons, locations and organisations. In particular, w.r.t. DBpedia concepts, Lupedia focuses on a subset of the top-level classes in DBpedia including Event, Person, Place, Work and Organisation, as well as their subclasses. Users can call Lupedia using either POST or GET request methods, where the user submits a text document and receives annotations in the output format specified with the request, such as JSON and RDF.

Extractiv is a web service providing entity extraction, entity linking, relation extraction, sentiment extraction and topic detection. The vocabulary of entity annotation contains around 200 types, of which the depth of the subclass hierarchy is usually no more than 2. The entities may also be resolved and linked to DBpedia URIs in the response. Texts and URLs can be submitted through its REST API and the result formats are in XML or JSON. There is a daily limit of 1000 calls for free users.

OpenCalais is a web API developed by Reuters based on NLP and ML approaches. It not only performs NER, but also provides semantic metadata about entity relationships and document topics. Plenty of annotation types are covered by the entity tagger of OpenCalais– currently 39 named entities in total, no pair of which are in a subclass relationship. When an entity is extracted, it is associated with a dereferenceable URI within OpenCalais's domain, which can be further explored to obtain extra information

such as URIs of external Web assets (e.g., DBpedia, Freebase). The main format of the output-annotated text is RDF. The web APIs can be accessed through REST or SOAP under a usage restriction of 50,000 calls per day.

Zemanta web service enriches submitted content with "suggested" information. The suggestions include related images, articles, in-text links and keywords, where in-text links suggest named entities in the content. The taxonomy of entity types in Zemanta is composed of a subset of Freebase's hierarchical schema, which involves 78 classes. Similar to Alchemy, Zemanta returns entities with the anchor text instead of its position, and in some cases an empty type could be returned if it is only known as a general entity (e.g., Thing). Possible destination links of suggestions for a specific anchor text are connected to external repositories (e.g., Wikipedia, Youtube). Zemanta API can be invoked via a REST endpoint or be integrated into a web publishing platform as a widget by using their JavaScript SDK. The API accepts HTML documents or plain text documents as input and outputs result in multiple formats. The maximum length of an input text is 8KB.

Saplo is a web API providing NE tagging, suggestion of related text, and sentiment detection. Its NE tagging service returns entities categorized as Person, Organisation, Location, Url and unknown. The input could be a single text or a collection of texts which can be processed in parallel, and the main format for responses is JSON.

Wikimeta is a web API whose functionality includes semantic annotation tasks (e.g., NE recognition, sentiment annotation) and NLP-related tasks (e.g., Part of Speech Tagging). The API of NER deals with 7 common general classes (e.g., locations, persons and time) and their subclasses. Users are able to submit documents through a REST interface. The returned output could either be XML or JSON.

YahooQl Content Analysis web service extracts entities, document categories and relationships within unstructured content and ranks them by relevance. The entities returned are resolved into Wikipedia pages if possible. The APIs are accessible via a REST request where users can pass a query of Yahoo Query Language (Yql). The language allows users to query, filter and combine data from different datasources on the Web. Taking the entity extraction service for example, a sample Yql query to harvest entities from given text is as follows:

```
SELECT * FROM contentanalysis.analyze WHERE text= "Italian
   sculptors and painters of the renaissance favored the Virgin
   Mary for inspiration"
```

Since there is no full list of NE types in the online documentation, the vocabulary had to be experimentally estimated. By using the testing corpora in Section 2.3.1, 59 types

Table 2.1: Instances of semantic annotators

	Annotator	Algorithm	Taxonomy	NE types	Linked Open Data	Input format	Output format	Key Required
Software	StanfordNer	ML	Flat	3	None	Text	XML inlineXML	No
	IllinoisNer	Hybrid	Flat	3	None	Text	Inner format	No
Web API	Alchemy	ML	Hierarchical	≥280	Freebase YAGO GeoNames	Text URL	XML JSON RDF	Yes
	Spotlight	ML Gazetteer	Hierarchical	≥220	DBpedia	Text URL	XML JSON RDF	No
	Lupedia	Gazetteer	Hierarchical	195	DBpedia LinkedMDB	Text	XML JSON RDF	No
	Extractiv	N/A	Hierarchical	225	DBpedia	Text URL	JSON RDF	Yes
	OpenCalais	N/A	Flat	39	DBpedia Freebase GeoNames IMDB LinkedMDB	Text XML HTML	XML JSON RDF	Yes
	Saplo	N/A	Flat	4	None	Text	JSON	Yes
	Wikimeta	ML	Hierarchical	52	None	Text	XML JSON	Yes
	YahooQl	N/A	Flat	≥59	Wikipedia	Text URL	XML JSON	No
	Zemanta	N/A	Hierarchical	78	Wikipedia IMDB	Text HTML	XML JSON RDF	Yes

are returned by the API. These types are limited to several high-level concepts along with a small collection of their subclasses. The supported output formats are XML and JSON. With respect to usage limitations, users are capped at a maximum of 2,000 calls using the public endpoint.

Table 2.1 summarizes the observations of the above annotators, taking into account the following aspects: availability, user interface, methodology, major input/output format and the accessibility to external ontologies.

In Table 2.1, we denote *N/A* as unknown , *ML* as Machine Learning-based approaches. The value of *Linked Open Data* shows the external ontologies they connect entities to via dereferenceable URIs if entity disambiguation is applicable. Focusing on the most widely-supported task – NE recognition, StanfordNer and IllinoisNer simply return text with entity type markups embedded in. Among those web APIs, Alchemy, Wikimeta, Saplo and Zemanta return text anchors with entity types, while all the rest are able to return entities along with their positions and types. More importantly, the taxonomies of named entity types they support are quite distinct. For example, OpenCalais deals with 39 types, most of which are general concepts such as Person and Organization. Spotlight, Lupedia and Zemanta adopt existing ontology resources (DBpedia and Freebase

respectively) as labelling vocabularies. The taxonomies of ALCHEMY, EXTRACTIV and WIKIMETA are comparatively richer both in depth and breadth, but the deepest subclass hierarchy is quite short (usually no more than 3), including both abstract and specialised concepts. STANFORDNER, ILLINOISNER and SAPLO only recognize a few classic top-level concepts.

2.2 Vocabulary alignment

One obvious disparity among these annotators is that the annotators do not have a unified vocabulary. Annotations that are syntactically distinct may have the same meaning, e.g., OPENCALAIS:NaturalFeature and ALCHEMY:GeographicFeature in Figure 2.1. In some cases, although annotators use terms which are lexically the same, their semantic definitions are different. For example, SAPLO uses ORGANIZATION to represent general organisations while EXTRACTIV refers ORGANIZATION to organisations which are not captured by other predefined specific organisation types. Some annotators, such as STANFORDNER, have mostly high-level concepts in their vocabulary – e.g. Organization; others such as EXTRACTIV, have many much finer classifications – e.g. FinancialOrg and CommercialOrg.

In order to capture the interrelationships of concepts in different vocabularies, we will make use of an ontology Ω which states the relationships between types of entities. In this study, the ontology Ω consists of a finite set of concepts (classes, types), including the special type $\bot$ (the empty concept) and $\top$ (the universal concept), along with constraints restricting or enforcing the co-occurrence of concepts. The most common constraints supported for concepts are:

- Subclass (subsumption) constraints $C \sqsubseteq D$, stating that an entity of type C is also an entity of type D.
- Disjointness constraints $C \sqcap D \sqsubseteq \bot$, stating that C and D have an empty intersection.

The above constraints are supported by the major languages for describing Web semantics, such as the OWL2-QL fragment of OWL. As mentioned above, SPOTLIGHT and ZEMANTA directly use an external ontology. The other annotators do not come with any mapping, and so the meaning of the concepts has to be recovered from the documentation. In some cases the concepts of the other annotators could be mapped into Freebase or DBpedia concepts, but in other cases new concepts need to be added. We believe that as these annotators grow mature, the use of standardized vocabularies will increase, making the use of rules as "hard facts", as we do throughout this study, appropriate.

```
<Class rdf:about="global:GeographicFeature ">
    <owl:equivalentClass rdf:resource="alchemyAPI:GeographicFeature"/>
    <owl:equivalentClass rdf:resource="OpenCalais:NaturalFeature"/>
    <rdfs:subClassOf rdf:resource="extractiv:LOCATION"/>
    <owl:disjointWith rdf:resource="OpenCalais:Facility"/>
    ...
</Class>
```

Figure 2.2: Example snippet of merged ontology

Here standardized vocabularies are referred to as uniform taxonomies which local vocabularies can be aligned to, and the "hard facts" are referred to as the interrelationship of the vocabularies such as Actor is a subclass of Person. An effort in this direction is the SCHEMA.ORG initiative. We are focusing in this study on *reconciling* annotator outputs using *schema information*, not on the very different issue of mapping/aligning concepts and schemas (in other terms, "integrating the ABox", not "integrating the TBox").

We have thus created a merged ontology manually, aligning the relevant fragments of Freebase and DBpedia within it, focusing on rules that are clear from the documentation. The two external ontologies focus on subsumption relations, while disjointness relationships are crucial for gauging whether two annotations are compatible. We thus manually added disjointness relationships that were obvious from the documentation. Figure 2.2 gives a snippet of our merged ontology that maps the concepts GeographicFeature from ALCHEMY and NaturalFeature from OPENCALAIS as equivalent classes of a global concept GeographicFeature, while the concept Location from EXTRACTIV is mapped as a superclass, and the concept Facility from OPENCALAIS as disjoint class. The merged ontology used in this study is accessible from `http://diadem.cs.ox.ac.uk/roseann/MergedOntology.owl` and can be queried through a SPARQL endpoint available at URL: `http://163.1.88.38:8081/openrdf-sesame`. Since each entity type from local vocabularies now has been mapped to an unique global class, we will simply use the global class names in all the examples throughout the rest of the thesis. We denote by "global vocabulary" the set of global classes.

Given the integrated taxonomy, the coverage of vocabularies supported by individual annotators can be easily figured out. Figure 2.3 shows a selected view on both overlap and diversity of NE types supported by different annotators. Some concepts are wildly covered such as City, Country and Person; some are shared by a subset of annotators – e.g. PhoneNumber and Building; and some are only recognized by a particular annotator – e.g. DayName. Figure 2.4 depicts percentages of the coverage over the global vocabulary. Blue bars indicate the portion of the exact coverage against the global vocabulary (denoted as Cov_{voc}), while red bars reveal the fraction of the potential named entity coverage by

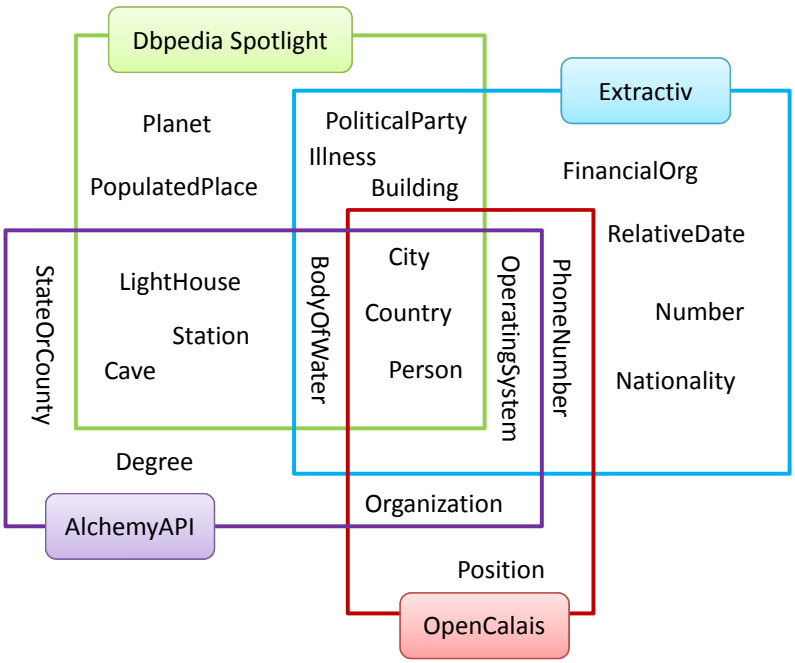


Figure 2.3: Example of overlapping vocabularies

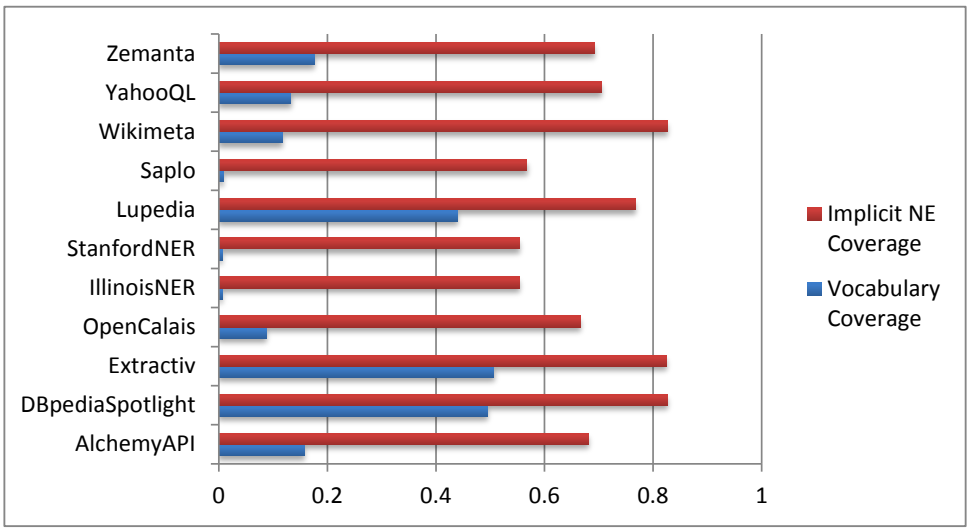


Figure 2.4: Percentage of coverage over global vocabulary

adopting the subsumption constraints (denoted as Cov_{ent}). More specifically, Cov_{voc} is the number of exact concepts in the local vocabulary divided by the number of concepts in the global vocabulary. Cov_{ent} is the number of concepts in the local vocabulary along with their subclasses divided by the total number of the global vocabulary, where we simply assume the instances of each concepts are uniformly distributed. The measurement of Cov_{ent} implies the potential recall which annotators are supposed to obtain. We note that annotators with a low Cov_{voc} could still be supposed to recognize a high portion of entities

if their vocabularies consist of high-level concepts, e.g. SAPLO and STANFORDNER.

2.3 Evaluation

In addition to the aspects such as vocabularies, the quality of annotation results could also be essential for end-users and annotation-consuming applications. In the following section, quantitative evaluations on real datasets are conducted to assess the annotation accuracy of individual annotators.

2.3.1 Dataset

We examine the performance of the individual annotators by "black-box" testing. For STANFORDNER and ILLINOISNER, we use their default "already-baked" models in the testing. The experiments are conducted on the following four benchmark corpora:

MUC7 dataset. The MUC7 dataset [64] contains 300 newswire articles from New York Times Service for evaluating named entity recognition. The articles of the dataset are selected from two different domains: aircraft accidents and missile launch events. They are annotated in SGML format with standard high-level concepts including Person, Organization, Location, Currency, Date, Time and Percentage.

Reuters corpus (RCV1). This corpus is released by Reuters. It consists of Reuters News stories as a standard benchmarking corpus for the IR/IE community. All the documents of the corpus are marked up in XML, subdivided into topics. Since the original Reuters corpus is very large (810,000 articles), we looked at five of the most common Reuters topics – *Entertainment/Sports*, *Financial/Economics*, *Healthcare/Social*, *Products* and *Tourism/Travel* – and randomly sampled 250 documents, distributing evenly over the topics. For this sub-corpus of Reuters, we manually tagged each documents by using the most specific concepts from the global vocabulary of the aggregated ontology discussed in Section 2.2.

Fox corpus. The corpus is used by the FOX [33] entity extractor and consists of 100 snippets of text from news headlines annotated with three concept types, namely Person, Location, and Organisation.

Illinois corpus. The corpus is used by the ILLINOIS entity extractor and consists of text sourced from 20 web pages mostly about personal, academic and computer-science conference homepages and annotated with the same three concepts above plus a MISC concept which is excluded from the evaluation due to the ambiguity in the annotated instances. E.g., the string *Computer Science* is annotated as MISC, but other similar

Table 2.2: Description of testing corpora

Test Corpus	Docs	Covered Types	Entities (≈)
MUC7	300	7	18,700
Reuters	250	215	51,100
Fox	100	3	395
Illinois	20	3	624

topics e.g., *Biomolecular Computation* are not. This corpus also allows us to see the performance of our own annotators in a very different domain from Reuters and MUC7.

Note that except the Reuters corpus, we directly use the original gold standard provided by the benchmarks. Details on the coverage of each test set are shown in Table 2.2. Our gold standard annotation of the Reuters corpus is available online from [76].

2.3.2 Evaluation measurement

In NER competitions, the NER algorithms are judged according to standard scoring functions based on how their output compares with the gold-standard. The scoring functions vary from the simplicity of CoNLL to the complexity of ACE. In general, a system is evaluated on two axes: its ability to assign the correct type and its ability to find the exact snippet. In this study, the constraint of exact boundary match will not be applied. On the one hand, in many applications, the goal is to determine whether or not a particular entity type is mentioned in the text without requiring exact boundaries. On the other hand, the diversity of the response from individual annotators shows that exact NE boundaries is unnecessarily stringent. For instance, one annotator annotates the snippet "the Subic Naval Base" while another one annotates "Subic Naval Base" without covering the token "the", but both of them are considered to be correct. Hence in this study, a correct annotation is credited if an entity is assigned the correct type, regardless of boundaries as long as there is an overlap.

After the clarification of the boundary issue in our scoring manner, we then present the performance indicators. Precision and Recall are defined in a way that is *ontology-aware*: for example, given an ontology Ω, if an annotator declares a given span to be a Person while our gold standard indicates that it is an Artist, then this annotator will be eventually penalized in recall for Artist (since it had a miss), but not in precision for Person (since it can be inferred via Artist). There is no standard way to do this. Euzenat [27] has defined semantic precision and recall measures in the context of ontology alignment

– but their goal is to assign static measures to an alignment, which approximates the precision and recall one would get from an "ideal" extension of the concepts via sets of instances. In our case, we can use concrete cardinality estimates since we are interested only in the precision and recall for an algorithm on a particular dataset.

More precisely, we define the precision of an annotator AN for a concept C as:

$$\text{PRECISION}_{\Omega}(C) = \frac{|Inst_{\text{AN}}(C^+) \cap Inst_{GS}(C^+)|}{|Inst_{\text{AN}}(C^+)|}$$

where $Inst_{\text{AN}}(C^+)$ denotes all instances annotated as (a subclass of) C by AN, and $Inst_{GS}(C^+)$ denotes all instances determined to be (a subclass of) C in the gold standard. In computing the intersection, we use a "relaxed" span matching, which requires only that the spans overlap. Here and throughout the remainder of the paper, when we say that the concept A is a subclass of a concept B we mean that the rules of the ontology derive that B is a subclass, possibly improper (that is, A may be equivalent to B). Similarly, when we talk about superclasses of a concept A, we include those derivable via transitivity, and include A as a superclass of itself.

We define the recall of an annotator AN for a concept C in an analogous way, again using the "relaxed" notion of matching for the intersection:

$$\text{RECALL}_{\Omega}(C) = \frac{|Inst_{\text{AN}}(C^+) \cap Inst_{GS}(C^+)|}{|Inst_{GS}(C^+)|}$$

Based on the extended definitions of precision and recall, the F-score for concept C is defined in the standard way:

$$F - score_{\Omega}(C) = \frac{2 \times \text{PRECISION}_{\Omega}(C) \times \text{RECALL}_{\Omega}(C)}{\text{PRECISION}_{\Omega}(C) + \text{RECALL}_{\Omega}(C)}$$

2.3.3 Evaluation results

We tested individual annotators over the datasets using the entire merged ontology as a reference schema. We implemented ontology-aware scorers in order to calculate the precision and recall defined above. The full list of types with score for each individual annotator, as well as the source code of the scores are available online from [76].

Selected results are presented in terms of PRECISION and RECALL from Figure 2.5 to Figure 2.12, along with the corresponding F-score in Table 2.4, where highlighted values represent the best/worst performance. For each annotator AN, we display results for all

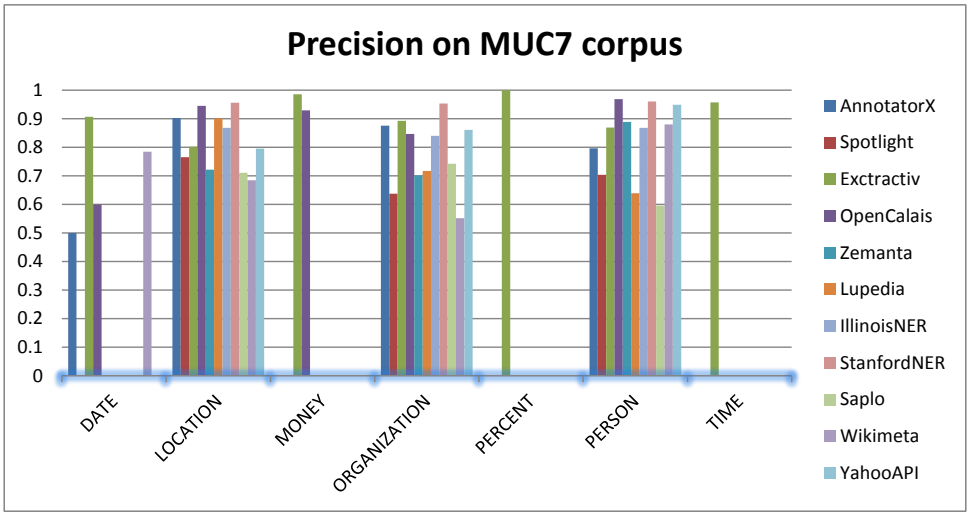


Figure 2.5: Precisions of individual annotators on MUC7 corpus

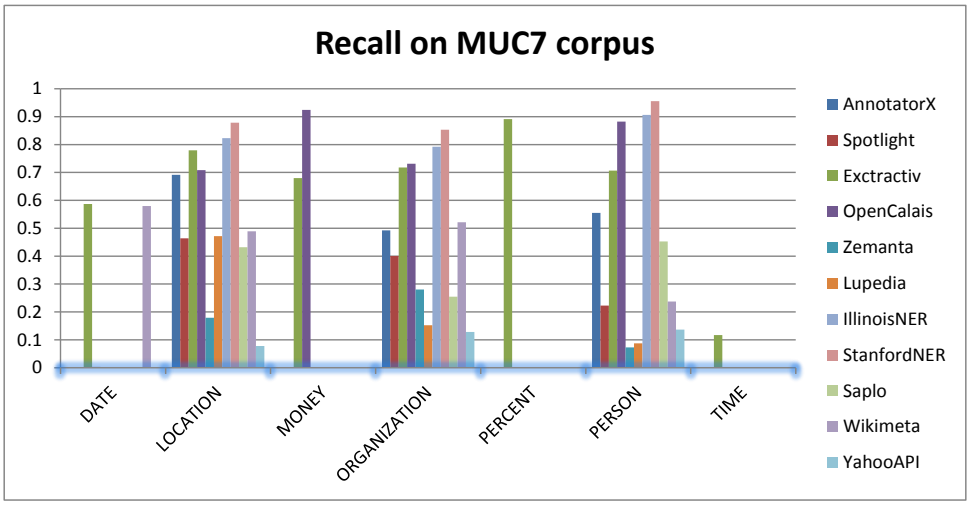


Figure 2.6: Recalls of individual annotators on MUC7 corpus

of the MUC7, Fox, and Illinois concepts that are in the vocabulary of An. The main reason to do so is that the utility and the difficulty of recognizing some types against others varies. Note that evaluation values of unsupported concepts are represented by an empty bar in the charts and the symbol "-" in the table. For each corpus, we first mapped the response of individual annotators into the ones with the types in the corpus and then compared the matching. For the Reuters dataset we give a selection of concepts in the following manner: **(i)** filter down to concepts with instances occurring in all of the testing folders and with at least 10 occurrences; **(ii)** organize the concepts by their F-score for An, restricting to a subset including roughly half of the concepts from the top quartile, and half from the lowest quartile **(iii)** within the subsets above, select subsets so as to avoid having more than two subclasses of any given concept. We thus selected concepts

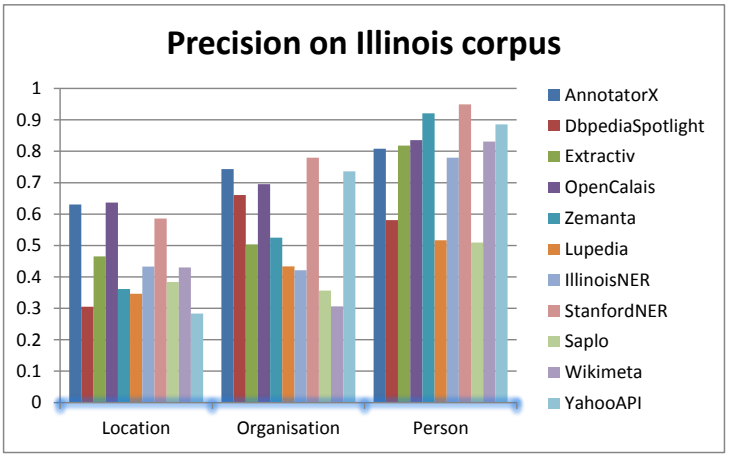


Figure 2.7: Precisions of individual annotators on Illinois corpus

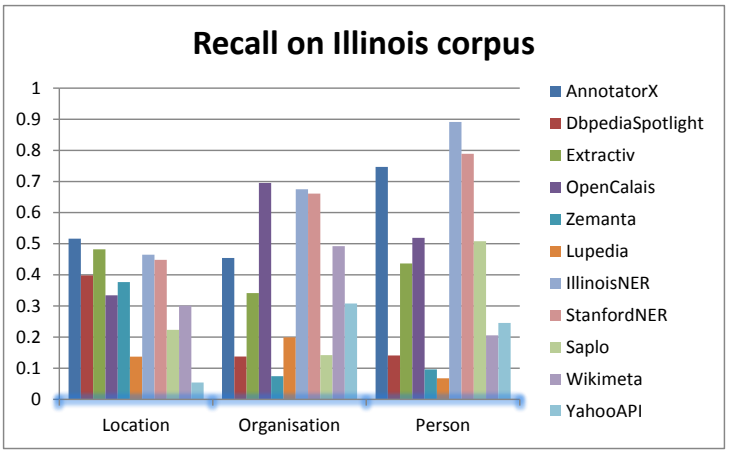


Figure 2.8: Recalls of individual annotators on Illinois corpus

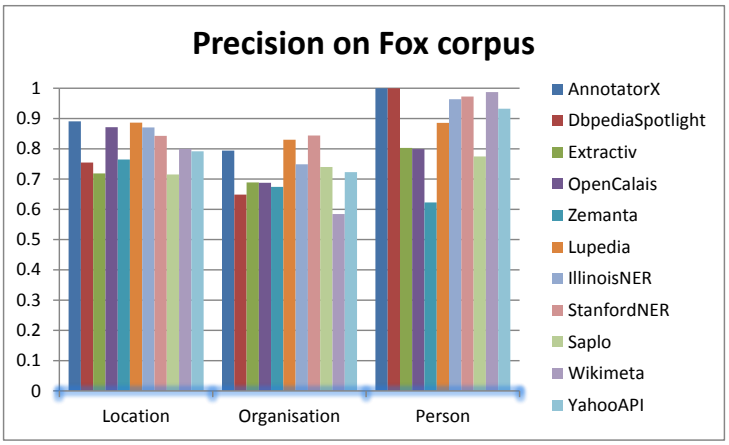


Figure 2.9: Precisions of individual annotators on Fox corpus

with sufficient support, diversity in performance, and diversity within the ontology.

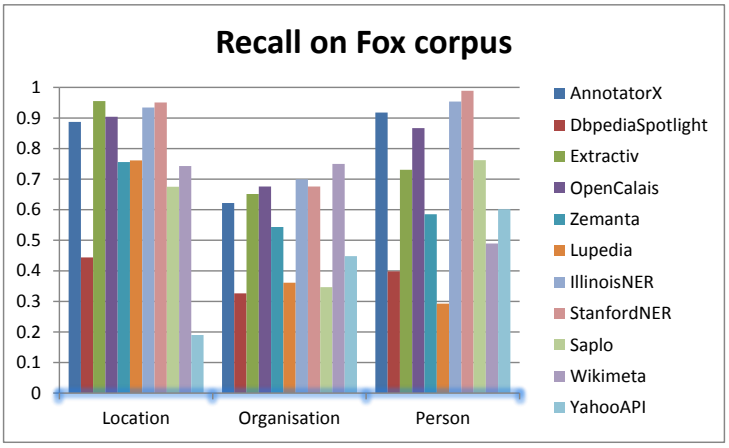


Figure 2.10: Recalls of individual annotators on Fox corpus

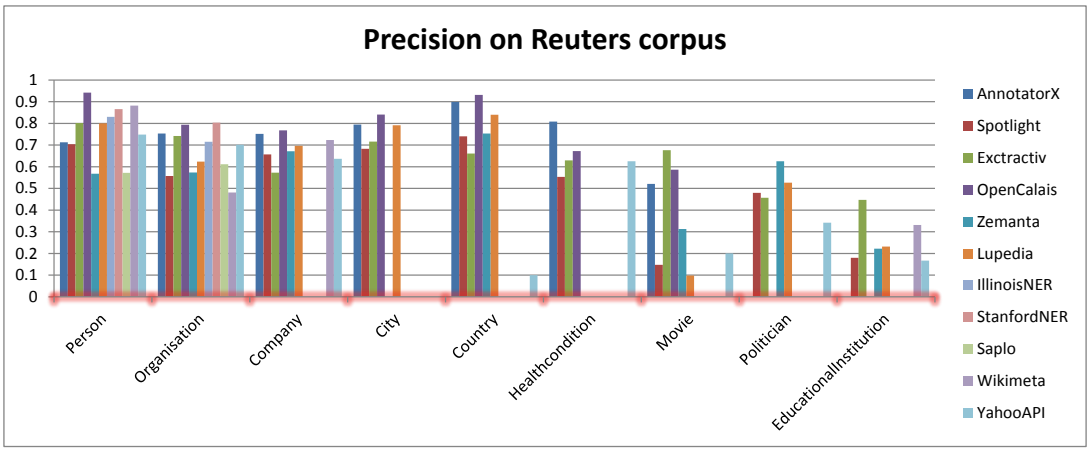


Figure 2.11: Precisions of individual annotators on Reuters corpus

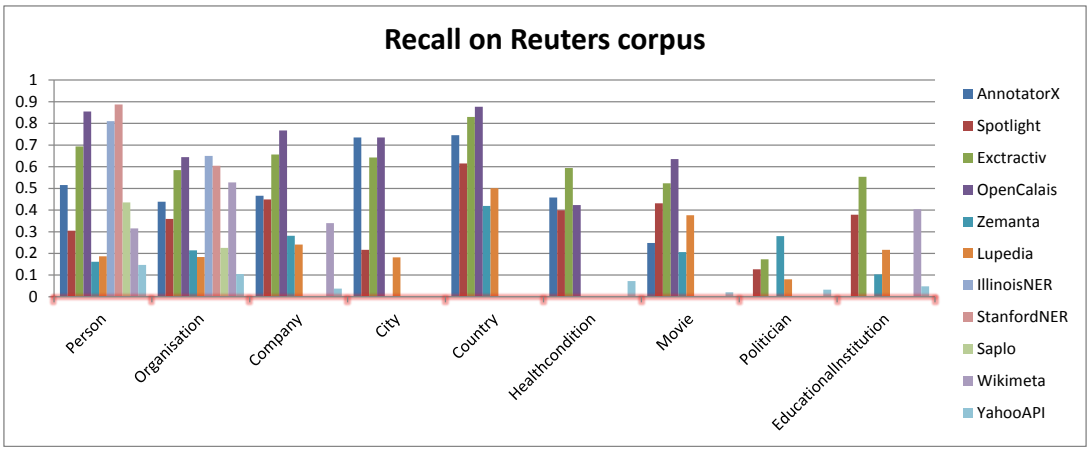


Figure 2.12: Recalls of individual annotators on Reuters corpus

Table 2.3 summarises important observations from the benchmarking results. We see

Table 2.3: Observations from the benchmarking results

Observations	Examples
Accuracy on one concept among annotators varies *greatly*.	E.g., Person (YahooQl: 24% vs. Alchemy: 89.6%)
Performance differences within an annotator are quite *dramatic*.	E.g., Extractiv: Country (73.5%) vs. Politician (18.8%)
No single annotator has a clear advantage.	E.g., OpenCalais has a higher performance on concepts but only with limited coverage of the overall vocabulary
All annotators contribute to recall.	E.g., Spotlight: Planet and Extractiv: RelativeDate

from the results that each annotator varies greatly in performance, ranging from below 10% in F-score to above 90%. If we focus on concepts that occur in the list for multiple annotators, we also see tremendous variation in performance, with no single annotator having a clear advantage. Overall, software-based systems achieve competitive results in recognizing entities of top-level concepts. By contrast, the web API Saplo yields inferior performance on those concepts. We observe that OpenCalais has a higher performance on most of the concepts that are within its vocabulary, but it has a deficit on a few shared concepts, and on many important classes, such as Date, it can identify only small subclasses. Alchemy and Extractiv overall have satisfactory performance on both general and specialised concepts. YahooQl in most cases obtains the worst scores w.r.t. F-score (e.g. $<$20%) The two DBpedia-based annotators Spotlight and Lupedia obtain good precision which are in most cases above 70%. However, they show poor recall on all the testing corpora especially in those general concepts. One possible reason for this is that they rely on the DBpedia dataset which may only be limited to well-known entities such as celebrities.

The performance differences within an annotator are quite dramatic, and surprisingly many of the concepts which exhibit the worst performance are not obviously the most semantically complex. For example, Movie would seem like something straightforward to handle by referencing an online database such as IMDB, but the recall numbers for Alchemy, Spotlight, and Zemanta were 20%, 28%, and 17% respectively. Also, the performance of annotators on very high-level concepts, such as those in MUC7, is generally higher than the performance on more specific concepts. However the example of Zemanta shows that this is not universally true.

Another interesting observation is that all annotators contribute to recall, i.e., each annotator contributes some annotations that are not produced by any of the others.

Table 2.4: F1 performance of individual annotators (%).

		Alchemy	Spotlight	Extractiv	OpenCalais	Zemanta	Lupedia	IllinoisNer	StanfordNer	Saplo	Wikimeta	YahooQL
MUC7	Date	.5	-	**71.3**	**.6**	-	-	-	-	-	66.7	-
	Organisation	63	49.2	79.6	78.5	40.1	24.8	81.4	**90**	37.8	53.4	**22.3**
	Person	65.4	33.8	78	92.4	**13.5**	15.4	88.6	**95.8**	51.1	37.3	23.8
	Location	78.3	57.8	79	81	28.7	61.8	84.4	**91.5**	53.6	56.9	**14.1**
	Money	-	-	**80.5**	**92.7**	-	-	-	-	-	-	-
	Percent	-	-	**94.2**	-	-	-	-	-	-	-	-
	Time	-	-	**20.9**	-	-	-	-	-	-	-	-
Illinois	Person	77.6	22.7	56.9	64	17.4	**11.2**	80.6	**85.4**	49.5	31.1	36.7
	Organisation	56.4	22.8	40.7	69.5	**13**	25	50	**70**	17.6	36.6	42.2
	Location	**56.8**	34.6	47.4	43.8	36.9	18.2	43.5	47.8	.25.1	32.1	**6.7**
Fox	Person	95.7	57	76.5	83.2	60.3	**42.6**	95.7	**98.1**	74.3	63.1	71.5
	Organisation	69.7	**43.4**	67	68.2	60.2	49.4	71	**71.4**	45.1	64.4	52.4
	Location	89.1	55.9	82	88.7	76	81.4	**89.8**	88.6	67.1	76.3	**27.7**
Reuters	Person	**89.6**	42.5	74.1	89.6	24.9	30.2	81.9	87.6	49.1	46.4	**24.5**
	Organisation	55.3	43.5	65.2	**71**	30.9	28.2	68.1	68.9	32.7	50.2	**18.1**
	Company	57.2	53	60.9	**76.6**	39.2	35.6	-	-	-	45.9	**6.9**
	City	76	32.6	67.3	**77.9**	-	**28.9**	-	-	-	-	-
	Country	81.3	66.9	73.5	**90.3**	53.2	62.3	-	-	-	-	**0.1**
	HealthCondition	55.4	44.3	**59.1**	47.2	-	-	-	-	-	-	**12.6**
	Movie	30.1	19.4	**55**	21.9	13.9	-	-	-	-	-	**3.7**
	EducationalInstitution	-	22	**48.7**	-	14.1	22.1	-	-	-	34.1	**7.1**
	Politician	-	18.8	23.2	-	**37.7**	13.6	-	-	-	-	**5.9**

2.4 Conflicts of individual annotators

Given that individual annotators can vary radically in accuracy and that we have to consider all of them, a question is to what extent the semantics of the annotations can be useful to determine errors in the annotations. We take one coarse measurement of this, based on the amount of *annotator conflicts*.

We measure two kinds of annotation conflicts: a *basic conflict* occurs when one annotator annotates the span with a concept C, and another annotator which has (a superclass of) C in its vocabulary fails to annotate the same span with it. Clearly, this represents a situation where the annotators have "different opinions". For annotators with low recall, a basic conflict may be a weak indicator of a problematic annotation. Thus we also track *strong conflicts*, denoting situations when two annotators annotate the same span with disjoint concepts C and C'. Table 2.5 shows the number of basic and strong conflicts in each datasets – both the number of annotated spans in which a given type of conflict (basic and strong) occurs, and the total number of conflicts of each type. An annotated span is a span which contains at least one annotation.

Table 2.5: Conflicts statistics of individual annotators

Test Corpus	Annot. Span	Basic Confl.	Basic Span	Strong Confl.	Strong Span
MUC7	30,689	36,756	26,174	3,501	2,483
Reuters	29,162	21,639	15,654	2,937	1,981
Fox	798	943	605	185	68
Illinois	1,493	1,486	1,195	179	83

Although one might expect that conflicts are rare, the results show that they are extremely frequent. A major source of conflicts is that annotators have very limited capability of using context to distinguish ambiguity in meaning. *Bloomberg* sometimes refers to a company, and sometimes to a person; *Jaguar* can refer to an animal or to a car; *Chelsea* can refer to a place, a person, or a sports team; *Notting Hill* can refer to a place or a movie. Different annotators tend to favor particular solutions to these ambiguities. Note that the number of conflicts is restricted by the limited overlap in the vocabularies and limited recall of the annotators. For example, it is very rare for three annotators to be mutually strongly conflicting on the same span, since it is unlikely that all three will simultaneously annotate the span.

We also track conflicts in spans, e.g. annotators agreeing on a concept, but disagreeing on the span. Table 2.6 reports, for each corpus, on the number and type of span conflicts, in particular:(i) the total number of annotations, (ii) the number of annotations with same span, (iii) the number of annotations having one span strictly contained within the other, (iv) the number of annotations having overlapping spans, and (v) the total number of annotations having conflicting spans (i.e., either contained or overlapping). The statistics of span conflicts shows that besides semantic labels, the boundaries of entities also suffer from a high degree of inconsistency among individual annotators.

Table 2.6: Span conflicts.

Test Corpus	Tot. Annot.	Same Span	Containment	Overlap	Conflicts
MUC7	96,111	205,289	43,262	958	44,220
Reuters	87,666	198,745	37,743	898	38,641
Fox	2,730	8,107	2,012	38	2,050
Illinois	5,430	7,029	4,719	263	4,982

2.5 Summary

Named entity annotators automatically create rich semantic metadata for the content of unstructured documents, by which the meaning of documents can be exploited for a wide

range of useful purposes. This chapter investigated a collection of prevalent annotators including free applications and web service APIs. We then conducted an integration at the level of NE types. A translation was described that maps annotator vocabularies into a common ontology. With this in hand, we were able to gain new insight into the behaviour of distinct annotators, such as their accuracy and the amount of conflicts.

Chapter 3

Reconciling Semantic Annotations

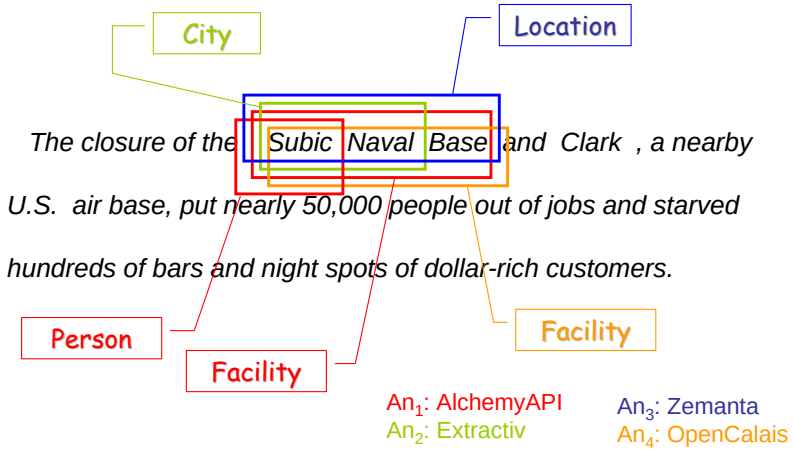
As mentioned in Chapter 2, the ecosystem of semantic annotators is now becoming increasingly diverse. Even though the vocabularies are different among these annotators, there are many semantic interconnections among these annotators. Moreover, the state-of-the-art annotators are inherently imprecise, resulting in conflicting information of the extraction results. The diversity in the capabilities and the vocabularies of the annotators give evidence for the necessity of *semantic annotation integration*, which aims at resolving conflicts as well as detecting and reacting to patterns of reliability and unreliability among annotators to provide trustworthy reconciled annotations. This chapter overviews the approaches to performing *ontology-aware aggregation*, introducing supervised machine learning-based aggregation methods. We consider several possibilities including outputting the results with probabilities and simply the most likely answer. We further experimentally compare these approaches with respect to ontology-unaware supervised approaches, and to individual annotators.

3.1 Motivation

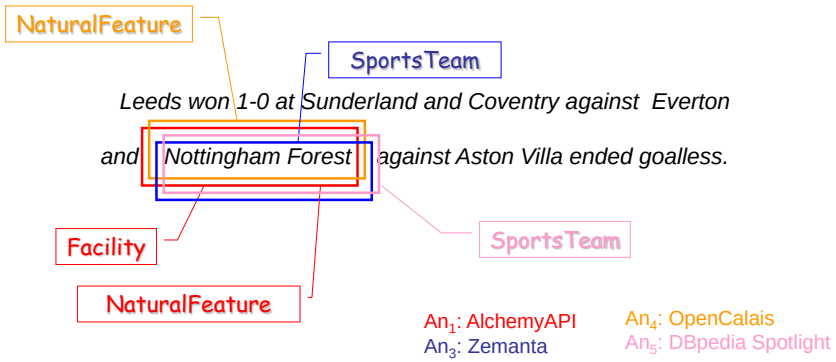
As shown in Chapter 2, each annotator supports a distinct vocabulary of semantically-related entity types, varying from a few high-level standard classes to a taxonomy of general and specialized entity classes. In addition, these most widely-used annotators suffer from serious accuracy deficiencies. By making use of a merged ontology, the vocabularies are found to mutually overlap with each other in different degrees. The statistical results in Section 2.4 reveal the fact that the semantic annotators often provide conflicting opinions in both aspects of entity types and entity spans.

Consider the example displayed in Figure 3.1(a), which shows a fragment of text from the Reuters corpus, mentioning a US naval base in the Philippines and classified by several Web-based annotators. The results include some high-level concepts (a Location, a Person), as well as some more specific ones. Notice that the annotation of a

city is only returned by the Web-based annotator EXTRACTIV: indeed, several of the annotators simply do not have this concept in their vocabularies. Also note that annotators give differing classifications of the same snippet – one annotator determining that "Subic Naval Base" is a Location and others determining that it is a Facility. Three of them (An_1:Facility, An_4:Facility, and An_3:Location) share the same span. Two others (An_1:Person and An_2:City) have different (but overlapping) spans. A second example is shown in Figure 3.1(b). Here we see several flavors of differences in annotator opinions. "Nottingham Forest" is labeled as a NaturalFeature by OPENCALAIS and ALCHEMY. On the other hand, the annotations Facility and SportsTeam are clearly both incompatible with the prior classifications, as well as with each other.



(a) Conflicting and Re-enforcing Annotations



(b) Entity Annotations

Figure 3.1: Examples of Entity Annotations

Both the overlap and the diversity of annotators motivate the need for semantic annotation integration: middleware that produces a unified annotation with improved quality on top of diverse semantic annotators. In this study, it is referred to as ROSEANN

(Reconciling Opinions of Semantic Annotators). On the one hand, with more annotators, we can achieve a better coverage of the semantics by taking the advantage of the abundance of useful vocabularies. For example, recent work on web-scale wrapper induction (see, e.g., [20, 79]) uses semantic annotations to lower the need for human supervision. On the other hand, in the presence of inconsistency, we can improve the correctness of integrated annotations by leveraging ensemble wisdoms.

Although empirical results about the performance of individual annotators were obtained in Section 2.3, they are *not* meant to indicate an intrinsic pattern of which one is correct on which concepts. Results can vary as different datasets are used; furthermore, these annotators are frequently modified, with the modifications impacting both their vocabularies and their scope. Thus the goal of this work will not be to tune an aggregation algorithm to a particular dataset (e.g., by hard-coded rules capturing which annotator is seen to be reliable), but to come up with a general methodology for aggregating annotator opinions, not relying on hand-tuned rules or weights.

Our approach to integration will be influenced by several distinctive aspects of semantic annotation. First, semantic annotations do not form a flat space of tags for document snippets, but rather associate snippets with a meaning, with the space of meanings being highly structured – two annotations with different concepts may thus still be compatible (e.g. if the concepts are in a subclass relation). As shown in Section 2.1, in many cases, Web-based annotators provide mappings of instances and concepts to a publicly-available formal ontology – for example SPOTLIGHT links to the well-known DBpedia ontology. Even in cases where such mappings are not provided, ontological rules can often be conservatively inferred (e.g. a Facility is disjoint from an Attorney), and then utilized as a precise means of capturing compatibility or incompatibility of the judgements given by different annotators. We will investigate integration approaches that are *ontology-aware* where the ontologies help us to understand the semantic relationships that exist between annotations. Another key factor is that many annotators are black-boxes, making an approach based on hand-crafted "integration rules" problematic: a particular Web-based annotator may exhibit a certain pattern of errors at one point, and later change as the underlying rule engine is adapted. We will thus look for aggregation methods that are as *domain-independent* as possible, and do not rely on hand-rolled heuristics, focusing completely on the case of "black-box annotator aggregation".

3.2 Aggregation algorithms

As the first step towards the goal, Section 2.2 constructed a merged ontology which implements the integration at the level of vocabularies. In this section, we then consider the

reconciliation strategies on top of the global vocabulary. The technique used for aggregating annotations depends heavily on what information is available about the annotators behaviour. Two approaches are described for an ontology-aware integrator: the first, appropriate for an off-line scenario in which training data is available, makes use of ideas from supervised learning by adapting a Maximal Entropy Markov model to the setting of semantic annotation; the second, more appropriate for on-line use in the absence of training data, relies on ideas from judgement aggregation, which deals with aggregation of the opinions of experts while respecting logical constraints.

3.2.1 Preliminaries and assumptions

We deal with a scenario in which we have the following given notations:

- $<$*Document*$>$. Let $\mathcal{T}$ be a domain of tokens. A raw document D is represented as a sequence of (linearly ordered) tokens $D = \langle t_1, \cdots, t_n \rangle$, where $t_i \in \mathcal{T}$.

- $<$*Annotator*$>$. Let $\mathcal{AN}$ be a finite set of annotators $\mathcal{AN} = \{\text{AN}_1, \cdots, \text{AN}_m\}$, which perform named entity recognition.

- $<$*Ontology*$>$. Let Ω denote the merged ontology in Section 2.2, where axioms of subsumption, equivalence and disjointness are considered. Let $\mathcal{L}_\Omega$ be the set of concept classes in Ω.

- $<$*Annotation*$>$. An annotator produces an annotation $\underline{\mathsf{a}}$ in D by marking an *interval* (or *span*) $[\alpha, \beta)$ of tokens with a label l, where $l \in \mathcal{L}_\Omega$. The reason for using a token interval to represent a snippet rather than others such as a set of tokens is that semantic annotations are often unique to the position that a snippet locating in. Two identical token sets can be overlaid by different annotations by individual annotators because they have different context. We denote the interval covered by $\underline{\mathsf{a}}$ as $\mathsf{intv}(\underline{\mathsf{a}})$ and the label of $\underline{\mathsf{a}}$ as $\mathsf{type}(\underline{\mathsf{a}})$, thus we have $\mathsf{intv}(\underline{\mathsf{a}}) = [\alpha, \beta)$ and $\mathsf{type}(\underline{\mathsf{a}}) = l$. We also have $\mathsf{owner}(\underline{\mathsf{a}})$ to denote the annotator which produces $\underline{\mathsf{a}}$.

- $<$*Annotated document*$>$ An annotated document $D^+_{\mathcal{AN}} = \langle D, \mathcal{A}^D_{\mathcal{AN}} \rangle$ consists of the raw document D and a set of annotations after running annotators $\mathcal{AN}$ on D, where $\mathcal{A}^D_{\mathcal{AN}} = \{\underline{\mathsf{a}} | \mathsf{intv}(\underline{\mathsf{a}}) \subseteq D \wedge \mathsf{owner}(\underline{\mathsf{a}}) \in \mathcal{AN}\}$.

- $<$*Orthogonal span*$>$. Two spans $\hat{s}_1 = [\alpha_1, \beta_1)$ and $\hat{s}_2 = [\alpha_2, \beta_2)$ are orthogonal (or disjoint) iff $\hat{s}_1 \cap \hat{s}_2 = \varnothing$.

- $<$*Overlapping span*$>$. Two spans $\hat{s}_1$ and $\hat{s}_2$ are overlapping iff $\hat{s}_1 \cap \hat{s}_2 \neq \varnothing$.

We give the assumptions we make before we propose the aggregation approaches :

1. For simplicity, we assume that all the base annotators behave independently as "black boxes", although some works in truth finding such as [23] will consider the inter-dependency of opinion providers, e.g., one source could be a copier of another.

2. We assume that all the base annotators give their opinions with full confidence. One reason is that only a few of the annotators are able to return annotations with certain confidence scores (e.g., EXTRACTIV). Moreover, annotators often use different scales which are difficult to normalize.

3. We assume that all the local opinions from individual annotators have already been mapped to the global vocabulary.

4. In terms of whether single or multiple labels are available, we assume:
   (i) Multiplicity of labels in input annotators: we allow spans to overlap, and that multiple labels can be associated with the same span.
   (ii) Multiplicity in output: a span can be assigned to multiple labels in the (true) output.

We finally define the problem of *ontology-aware* reconciliation of semantic annotations as:

Definition 1. *Given an annotated document* $D^{+}_{\mathcal{AN}} = \langle D, \mathcal{A}^{D}_{\mathcal{AN}} \rangle$ *and the global vocabulary* Ω, ***reconciling semantic annotations*** *means generating an annotated document* $D'^{+}_{\mathcal{AN}} = \langle D, \mathcal{A}'^{D}_{\{\overline{\text{AN}}\}} \rangle$ *by using a meta-annotator* $\overline{\text{AN}}$*, such that for any* $\langle \underline{\mathsf{a}}_1, \cdots, \underline{\mathsf{a}}_k \rangle$ *where* $\underline{\mathsf{a}}_1, \cdots, \underline{\mathsf{a}}_k \in \mathcal{A}'^{D}_{\{\overline{\text{AN}}\}}$ *and* $\mathsf{intv}(\underline{\mathsf{a}}_1) = \cdots = \mathsf{intv}(\underline{\mathsf{a}}_k)$*, we have* $\Omega \vDash \mathsf{type}(\underline{\mathsf{a}}_1) \sqcap \cdots \sqcap \mathsf{type}(\underline{\mathsf{a}}_k) \neq \bot$. □

Here the meta-annotator $\overline{\text{AN}}$, or federated-annotator, is an aggregator which encapsulates an ensemble strategy of reconciling diverse opinions in $\mathcal{A}^{D}_{\mathcal{AN}}$. The ensemble strategy needs to consider at least three aspects: (1) detecting a mention to an entity in D, (2) identifying the boundaries that delineate the name of a detected entity and (3) classifying the entity to its type in the domain of Ω and, more importantly, making sure that the resulting annotations for one single span is compatible w.r.t. Ω.

3.2.2 Supervised aggregation algorithms

Aggregation in the presence of supervision. Given the input annotation opinions which are inherently imprecise and uncertain, straightforward unsupervised methodologies such as solutions relying on a "voting" mechanism always fall short in the presence

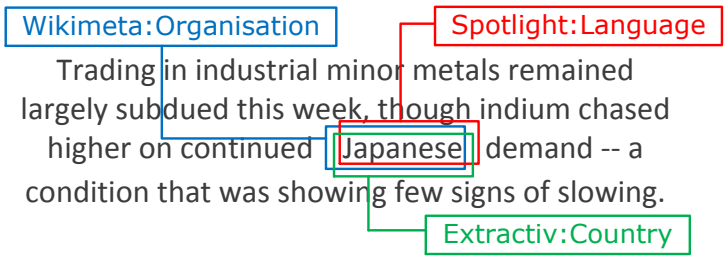


Figure 3.2: An example of imprecise annotation input

of low quality inputs (e.g., see Figure 3.2). Hence, we first consider a *supervised approach* to annotation integration, which leverages the existence of a training set to find patterns that indicate correctness or incorrectness of annotator results. For example, in Figure 3.2, we expect Nationality to be the integrated result for "Japanese", although none of the input annotations is correct. Our model will be general enough that we can detect a wide class of patterns of behaviour within annotators.

Given as input an annotated document which consists of a sequence of tokens, we can translate the integration problem in Definition 1 into the problem of input-output modelling of sequential data, which is a fundamental problem arising in many machine learning applications such as Part-of-Speech tagging [43]. This kind of problem can often be cast as one of estimating a "state" or "label" sequence $S = \langle s_1, s_2, \cdots, s_n \rangle$ for a given observation sequence $O = \langle o_1, o_2, \cdots, o_n \rangle$. Existing popular sequential models include generative models and discriminative models. Generative models, such as HMMs [99], are not necessarily appropriate since they learn a full joint distribution of the input and output sequences $\mathrm{P}(S, O)$ and thus often require more training data in order to achieve good performance [43]. In contrast, discriminative models only model the conditional distribution of label sequence given the observations $\mathrm{P}(S|O)$, so they are considered by us more appropriate for our problem.

Related discriminative models include *Maximal Entropy Markov Model* (MEMM) [60] which learns an observation dependent transition from current label to the next label and Conditional Random Field (CRF) [51] which normalizes the probability of a label sequence globally rather than locally as MEMM does. We adopt MEMM which has gained wide popularity for a variety of individual extraction tasks in recent years. On the one hand, Maximal Entropy learners are popular for base annotators, and thus are a natural choice for aggregation. On the other hand, our aggregation decisions generally rely on local properties of an annotated document, but locally have sequential dependence which can be captured by the architecture of MEMM. Furthermore, compared to methods such as CRFs, they are also easier to train. Note that a full weighing of the (quite numerous) alternative supervised methods was not a goal of this study.

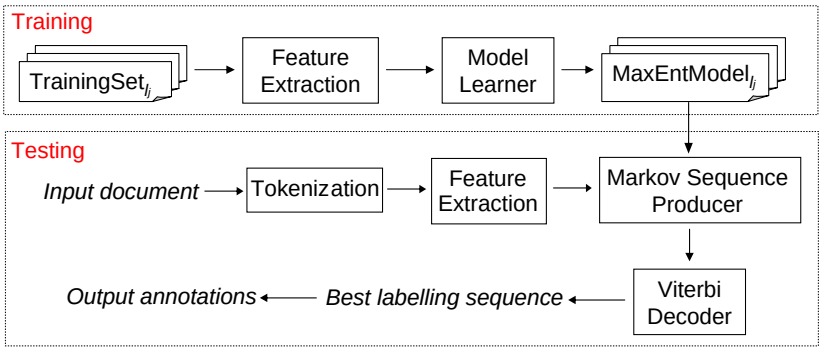


Figure 3.3: MEMM Framework

As in traditional Markov models, they allow for capturing patterns of sequential behaviour in a probabilistic transducer; a MEMM is a tuple $\langle \Sigma, \Delta, Q, q^0, \delta, \omega \rangle$ where Σ and Δ are finite *input alphabet* and *output alphabet* respectively, Q is a finite set of *states*, $q^0 \in Q$ is the *initial state*, $\omega : Q \times \Sigma \times Q \rightarrow \Delta^*$ is an *emit function* and $\delta : Q \times \Sigma \times Q \rightarrow [0, 1]$ is a probabilistic *transition function* satisfying: for $\forall \langle q, o \rangle \in Q \times \Sigma$, it holds $\Sigma_{q' \in Q} \delta_{\rightarrow}(q, o, q') = 1$. The inputs consist of sets of features token by token. As with other maximal entropy models, they allow for feature functions that overlap, without requiring strong independence assumptions. The *transition* and *emit* functions capture the system dynamics when a new input observation item is consumed.

Applied in this setting, the details of the elements of MEMM under the scenario of ROSEANN are explained below:

Input alphabet. Recall that the input is a sequence of observations $O = \langle o_1, o_2, \cdots, o_n \rangle$ where $o_i \in \Sigma$. Adapted to a sequential model, a standard *begin/in/out* (B-I-O) encoding scheme is adopted in our approach. Let $\mathcal{L}'_\Omega = \{B_, I_\} \times \mathcal{L}_\Omega$. $\mathcal{L}'_\Omega$ is thus an assertion that a token is a part of a given concept, along with a B-I decoration for its position within the concept. In other words, $\mathcal{L}'_\Omega$ is the set of all the possible atom labels with B-I decoration. Each input token t is accompanied by its corresponding labelling vector, named as *evidence vector*. An evidence vector, as defined in Definition 2, is composed of m associated label sets, one for each base annotator AN_i. Intuitively, $\vec{Ev}(t)$ encapsulates what all the base annotators "say" about the labelling of t. Note that we do not assume that the individual annotator opinions are consistent (see Example 1).

Based on the above discussion, we let the input alphabet $\Sigma = \mathcal{T} \times \mathcal{E}$ where $\mathcal{T}$ is the domain of tokens and $\mathcal{E}$ is the domain of evidence vectors. In particular, if an annotator AN_i provides an empty set in the evidence ($L_i^t = \emptyset$), we use a special label Other to indicate that token t is out of any of the classes in Ω.

Definition 2. *Let t be a token within a document and $\mathcal{AN} = \{\text{AN}_1, \cdots, \text{AN}_m\}$ is the set of base annotators. We define the* ***evidence*** *w.r.t. token t as $\vec{Ev}(t) = \langle\ (\text{AN}_1 : L_1^t), \cdots, (\text{AN}_m : L_m^t)\rangle$ where $\text{AN}_i \in \mathcal{AN}$ and $L_i^t \subseteq \mathcal{L'}_\Omega$.* □

Example 1. Consider the motivating example in Figure 3.1(a), the evidence of token "Subic" is $\vec{Ev}(\text{"}Subic\text{"}) = \langle(\text{ALCHEMY} : \mathsf{B_Person} \wedge \mathsf{B_Facility}), (\text{EXTRACTIV} : \mathsf{B_City}), (\text{OPENCALAIS} : \mathsf{B_Facility}), (\text{ZEMANTA} : \mathsf{B_Location})\rangle$. And for token "Base", we have $\vec{Ev}(\text{"}Base\text{"}) = \langle(\text{ALCHEMY} : \mathsf{I_Facility}), (\text{EXTRACTIV} : \mathsf{Other}), (\text{OPENCALAIS} : \mathsf{I_Facility}), (\text{ZEMANTA} : \mathsf{I_Location})\rangle$. □

States. For state space Q, we also adopt the B-I-O encoding scheme and let $Q(\Omega) = \{B_, I_\} \times \mathsf{Con}(\mathsf{Classes}(\Omega))$, where $\mathsf{Con}(\mathsf{Classes}(\Omega))$ is the set of collections of concepts that are consistent with ontology Ω. A symbol (B, S) where S is a consistent collection of concepts, indicates a position that is simultaneously the beginning of a snippet for an entity that is simultaneously a member of each element of S, while (I, S) indicates that the position is in the interior of such a snippet. We let Q be the power set of $Q(\Omega)$. We identify the empty subset as Other indicating that a token is out of the classes in Ω. Elements of Q can be mapped to a boolean combination in the obvious way: conjoining all the concepts in the list and dropping the set of decorations.

Conceptually, our set of states satisfy certain sanity conditions – e.g. that the corresponding combinations are consistent in the ontology. We do not have to verify these conditions, since the state set Q in practice only consists of the gold-standard labels that come from the training set, which will always satisfy these. For example, for the token "Subic" within Figure 3.1(a) in the training set, the corresponding gold-standard labels will not turn out to be unreasonable ones such as $\mathsf{B_Time} \wedge \mathsf{B_Person}$. However, as we support valid nesting annotations, we allow a token to be within multiple inconsistent annotations. For example, the token "Oxford" in "Oxford University" in the training set will be labelled with $\mathsf{B_City} \wedge \mathsf{B_University}$.

Initial state. We let $q^0 = \mathsf{Begin}$, which represents the beginning state of transducing an input sequence.

Output alphabet. In our solution, the output alphabet Δ is identical to the state space Q, which means the model emits exactly the state symbol when transiting into a state. Thus for $\forall\langle q, o, q'\rangle \in Q \times \Sigma \times Q$, we have $\omega(q, o, q') = \mathsf{Symbol}(q')$.

Transition functions. As in any Markov model, the key problem is to define the transition probabilities: given two states q, q' along with an input symbol o, define the probability of transiting from q to q' when traversing o. In the Maximal Entropy approach, this conditional probability is set to be the one which maximizes the entropy among those distributions in which the expectation of a distinguished set of feature functions agree

Algorithm 1: The GIS algorithm for learning ME parameters

Input: $\mathcal{T}^l = \{x_1, \ldots, x_{m_l}\}$: all the transition examples from state l in the training set; convergence condition C [1]; "correction" constant ω [2]

Output: λ: the vector of estimated maximum-entropy parameters corresponding to state l

foreach $f_i \in \mathcal{F}^l$ **do**
  calculate the empirical mean $empirical[f_i]$ from the training set, where $empirical[f_i] = \frac{1}{m_l} \sum_{k=1}^{m_l} f_i(l_{x_k}, O_{x_k})$;
Initialize the parameters $\lambda^{(0)}$, e.g. let $\lambda_i^{(0)} = 1$.
let $j = 0$;
while *C is not reached* **do**
  foreach $f_i \in \mathcal{F}^l$ **do**
    Calculate the expected value $expect[f_i]$ using $\lambda^{(j)}$ by letting $expect[f_i] = \frac{1}{m_l} \sum_{k=1}^{m_l} \sum_{l' \in \mathcal{S}} \delta_l^{(j)}(l'|O_{x_k}) f_i(l', O_{x_k})$, where $\delta_l^{(j)}(l'|O_{x_k})$ is calculated according to Formula 3.1 ;
    Update the parameter by letting $\lambda_i^{(j+1)} = \lambda_i^{(j)} + \frac{1}{\omega} \log(\frac{empirical[f_i]}{expect[f_i]})$;
  let $\lambda^{(j+1)} = \{\lambda_1^{(j+1)}, \ldots, \lambda_n^{(j+1)}\}$;
  j++;
return $\lambda^{(j-1)}$;

with their expectation over a sample distribution. Such a conditional probability must have the form (for intuition, $\delta_q(q'|o)$ is used instead of $\delta(q, o, q')$ below):

$$\delta_q(q'|o) = \frac{1}{Z(q,o)} exp(\sum \lambda_i^q f_i(q', o)) \tag{3.1}$$

Above $\mathcal{F} = \{f_1, f_2, \cdots, f_m\}$ is a finite set of feature functions, that take as input the target states along with the input symbol. $\lambda^q = \{\lambda_1, \lambda_2, \cdots, \lambda_m\}$ are the corresponding weight parameters learned in a training phase of the source state q, and $Z(q, o)$ is a normalization factor for the weights, which is calculated using the formula:

$$Z(q,o) = \sum_{q' \in \mathcal{Q}} exp(\sum \lambda_i^q f_i(q', o))$$

The weight parameters are set so as to make the expectation of the feature functions match their empirical expectation. There are standard methods to calculate the weights by training that achieve this. In particular, we can use the *generalized iterative scaling* (*GIS*) [60] presented in Algorithm 1, an algorithm that iteratively updates the parameters based on expectations defined over their current values.

Feature functions. The calculation of transition probabilities is straightforward once the feature functions are defined and the corresponding weights are trained. Note that feature functions play an important role in capturing patterns between annotator

opinions and aggregated outcomes. Our most fundamental features are based on annotator opinions. Due to the inherent problem of *data sparsity*, looking at the co-occurrence of all the events of the evidence vector in a feature function may suffer from a limited supply of training data. To address this, features are constructed based on *decomposed evidence* on a given token t, which focuses on the individual opinions of the annotators AN_i, as $\pi_{\text{AN}_i}[\vec{Ev}(t)]$ and the co-occurrence of pairwise opinions between different annotators AN_i and AN_j, as $\pi_{\text{AN}_i,\text{AN}_j}[\vec{Ev}(t)]$. Here the operator $\pi_A[\centerdot]$ is the projection about the annotation opinions on a set of annotators A. Therefore we are able to capture very fine-grained patterns, allowing these features to be specific to a particular piece of evidence and target state. For example, the feature function in Example 2 corresponds to the event where one annotator annotates a token as the beginning of a *Facility* and the token is actually the beginning of a *Person*. Example 3 is a feature function considers the co-occurrence of pairwise opinions from individual annotators.

Example 2.

$$f_1(l', o) = \begin{cases} 1, & \text{if } l_{\text{OpenCalais}} = B_Facility \text{ and } l' = B_Person \ ; \\ 0, & \text{otherwise.} \end{cases}$$

Example 3.

$$f_2(l', o) = \begin{cases} 1, \text{ if } l_{\text{OpenCalais}} = B_Facility, \, l_{\text{AlchemyAPI}} = B_Person \\ \quad \text{and } l' = B_Person \ ; \\ 0, \text{ otherwise.} \end{cases}$$

Figure 3.3 shows the general workflow of MEMM. With training completes, MEMM processes an input evidence sequence by constructing a *Markov sequence* based on Maximum Entropy model for each transition, and then decoding it using the standard VITERBI algorithm, returning the most likely labeling sequence. We explain the details below.

Training. Separate Maximum Entropy models were trained for each source state $q \in Q$ via the *GIS* with its corresponding training set $\mathcal{T}^q$. $\mathcal{T}^q$ consists of token events, of which the previous token label is q, as $\mathcal{T}^q = \{\langle \pi_{\text{AN}_i}(\vec{Ev}(t_k)), \pi_{\text{AN}_i,\text{AN}_j}(\vec{Ev}(t_k)), label_{t_k} \rangle \, | label_{t_{k-1}} = q \wedge \text{AN}_i, \text{AN}_j \in \mathcal{AN} \wedge \text{AN}_i \neq \text{AN}_j\}$. The model will ensure that an output annotation is always well-formed (e.g. never emit a concept with "I_" without seeing the corresponding opening tag with "B_" of this concept). Note that the Markov model resulting from this training processes the document per-token, not per-span, and makes no assumptions about the way annotator spans intersect. Thus nested or overlapping spans of base annotators do not require any special pre-processing.

Prediction. A trained MEMM transducer consumes the input from left to right and constructs the corresponding *Markov sequence* by applying the transition functions. A

$1-$ *order Markov sequence* is a sequence of random variables over a finite set of states, with the property that future states depend only on the current states [48]. The formal definition in our setting is given below:

Definition 3. *Given an input sequence* $O = \langle o_1, \cdots, o_n \rangle$ *and a* MEMM *transducer* $M = \langle \Sigma, \Delta, Q, q^0, \delta, \omega \rangle$*, the resulting* Markov sequence *of length* n*, denoted* $s^n_{\langle M,O \rangle}$*, operates over the set of states* Q *and comprises an initial state distribution* $\tau_{0\rightarrow} : Q \rightarrow [0,1]$ *and a transition function* $\tau_{i\rightarrow} : Q \times Q \rightarrow [0,1]$ *for all* $1 \leq i < n$ *where* $\tau_{0\rightarrow}(q) = \delta(\mathsf{Begin}, o_1, q)$*,* $q \in Q$ *and* $\tau_{i\rightarrow}(q, q') = \delta(q, o_{i+1}, q')$*,* $q, q' \in Q$*.* □

Example 4. Figure 3.4 shows an example of the resulting Markov sequence that is constructed by MEMM for the input test snippet "*The closure of the Subic Naval Base.*" assuming that $\mathsf{Classes}(\Omega) = \{Person, City, Facility\}$. Note that the snippet has already been annotated by base annotators as shown in Figure 3.1(a). S_i represents the random variable at time i where $1 \leqslant i \leqslant 8$. The state at time i is represented by a circle with its state name, where C_B and C_I represent B_C and I_C of concept C for brevity, and O denotes Other. The functions $\tau_{0\rightarrow}$ and $\tau_{i\rightarrow}$ are represented by directed edges that are associated with probabilities. For example, $\tau_{0\rightarrow}(O) = 0.97$ is indicated by the edge emanating from Begin on the left, and $\tau_{4\rightarrow}(O, Fac_B) = 0.6$ is indicated by the edge from O to Fac_B between variables S_4 and S_5. Note that transition edges with a zero probability are excluded from the figure. □

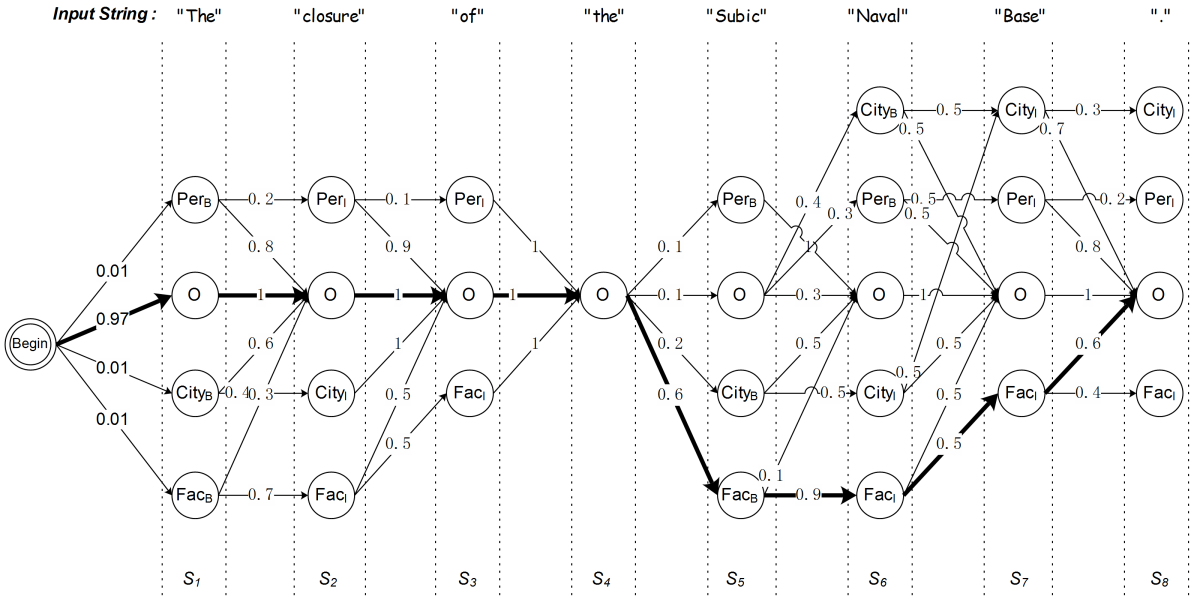


Figure 3.4: Example of Markov sequence

A *Markov sequence* $s^n_{\langle M,O \rangle}$ encodes the probability distribution of the output label sequence over n-long sequence of nodes, where the probability $\mathrm{P}(s|O)$, for an output

state sequence $s = s_1, \cdots, s_n$ ($s_i \in \Delta_M$ and $\Delta_M = \Sigma_M$), is given by Equation 3.2. Each random string generated by the Markov sequence corresponds to a unique directed path in $s^n_{\langle M,O\rangle}$.

$$p(s|O) = \tau_{0\rightarrow}(s_1) \times \prod_{i=1}^{n-1} \tau_{i\rightarrow}(s_i, s_{i+1}) \tag{3.2}$$

Example 5. By applying Equation 3.2, the probability of the output labelling sequence $s = \langle Other, Other, Other, Other, City_B, Other, Other, Other\rangle$, denoted by $P(s|O)$, is calculated by multiplying the transition probabilities along the unique path of s in Figure 3.4, which is $0.97 \times 1 \times 1 \times 1 \times 0.2 \times 0.5 \times 1 \times 1 = 0.097$. □

In order to identify the most likely path of states given a Markov sequence, we make use of the VITERBI algorithm which is an efficient dynamic programming solution of such classic problem. The VITERBI algorithm uses dynamic programming to generate a table with *forward probabilities* $Pr_t(q)$ which in our setting is defined as the probability of being in state q at time t given the observation sequence up to time t. The recursive VITERBI step thus is shown in the following equation.

$$Pr_{t+1}(q) = \sum_{q' \in Q} Pr_t(q') \times \tau_{\rightarrow}(q', q) \tag{3.3}$$

Once the most likely state sequence is generated by MEMM, we translate the B-I decorations into the boundary information of entities and construct the corresponding annotations which are the final output of MEMM. For example, given the best labelling sequence in Figure 3.4 which is marked in bold, we output the resulting annotation set $\mathcal{A}^D_{\mathcal{AN}} = \{\underline{\mathsf{a}}_1 = (\text{“}SubicNavalBase'' := \mathsf{Facility})\}$.

Implementation. The implementation utilizes the library OpenNLP-MAXENT-3.0.0 [67] as the core solution for the Maximum Entropy framework. This is a mature JAVA package for training and running maximum entropy models, where the parameter value of the iteration number is set to be 100 in our implementation. The toolkit OpenNLP Tools [68] is used to deal with common NLP tasks such as tokenization and sentence segmentation.

The complexity of using VITERBI to predict the most probable state sequence is $O(n \centerdot |Q|)$ where n is the length of input sequence and $|Q|$ is the size of states. As the performance of predicting for a new input sequence is highly sensitive to the length of the sequence and the state space of MEMM, several optimizations are applied w.r.t. the above two factors.

Using finer-grained input unit. In tradition, natural language *sentences* or *clauses* are often used as the processing units in NLP-based IE tasks. However, in our case, the

assumption of sentence-based input is less justified. On the one hand, we assume all the linguistic-based features, such as grammatical structure and Part-of-Speech information, are already taken into account by various base annotators while our aggregation algorithms only need to focus on reconciling their output opinions, regardless of whether the input is a complete sentence or not. On the other hand, more and more data to be processed is not necessarily grammatically correct. For example, snippets in web pages are often not full sentences which require a more flexible granularity as the processing units. In addition, even in a grammatically correct document, there exist a large number of sentences or fragments which merely contain plain text without any hint of occurrences of entities (e.g, all the individual annotators say Other), thus in this case it would be less necessary to pass them through the aggregators. This motivates us to only perform aggregation on "meaningful" units of token sequence for the efficiency of MEMM other than an entire sentence or a document.

Based on the above discussion, we thus are only concerned with those spans which are overlaid with "non-Other" annotations, which means the occurrence of a candidate entity to be reconciled is detected when some annotators say "something" over a span. In more detail, we define the sequence unit to be processed by MEMM as a *target connected component* (TCC for short):

Definition 4. *Given an annotated document* $\langle D, \mathcal{A}^D_{\mathcal{AN}} \rangle$*, a target connected component* Λ *consists of a collection of annotations* $\Lambda \subseteq \mathcal{A}^D_{\mathcal{AN}}$*, where for* $\forall \underline{\mathsf{a}} \in \Lambda$ *it holds: 1)* $\exists \underline{\mathsf{a}}' \in \Lambda$ *such that* $\mathsf{intv}(\underline{\mathsf{a}})$ *and* $\mathsf{intv}(\underline{\mathsf{a}}')$ *are overlapping spans and 2)* $\forall \underline{\mathsf{a}}'' \in \mathcal{A}^D_{\mathcal{AN}} - \Lambda$ *we have* $\mathsf{intv}(\underline{\mathsf{a}})$ *and* $\mathsf{intv}(\underline{\mathsf{a}}'')$ *are orthogonal spans.* □

Example 6. Considering again our running example in Figure 3.1(a), within the annotated snippet, there is only one TCC which contains all the five annotations represented by rectangle highlighters. □

Adapting to our sequential model described before, each TCC can further be represented by a token-based observation sequence $TCC = \langle (t_i, \vec{Ev}(t_i)), \cdots, (t_{i+k}, \vec{Ev}(t_{i+k})) \rangle$ where i and k are the index of the beginning token and the length in tokens of TCC respectively, thus t_{i+k} represents the ending token of TCC. Recall that the evidence vector $\vec{Ev}(t)$ is defined in Definition 2. A tokenized TCC sequence is then used as an input unit for MEMM, while for non-TCC token sequences in the document we simply output Other for each token.

Pruning outgoing transitions Some post-processing is performed on the Markov sequence, removing transitions that are (heuristically) unlikely to be relevant. The strategies include:(i) if there is an outgoing transition from a state having probability over a given threshold (currently set to 0.6), we will remove all other outgoing transitions from

the representation, and (ii) if there is an outgoing transition with probability below a given threshold (currently set to 0.001) then we will remove this outgoing transition from the representation.

Besides the above optimization, we gain additional speed-up by: (i) using a hash map to index only the transitions with nonzero probability, taking advantage of the sparseness of the transition matrix; (ii) Running in parallel over each TCC after further exploiting the fact that the proposed Markov model does not consider interactions across TCCs. In our test corpus of Reuters from which more than one thousand states are considered, the time of processing decreases from 20-30 seconds to 0.1-0.2 seconds per document on average by applying the optimization above.

3.2.3 Unsupervised aggregation algorithms

In the absence of dependable training data, RoseAnn provides a fully unsupervised alternative to MEMM based on the notion of *weighted repair*. This notion is a weighted extension of the approach adopted for consistent query answering over inconsistent knowledge bases [75], where the weighting represents, roughly speaking, the amount of support or opposition that is accorded to a given repair action.

Consider the example Figure 3.1(a) where a span $\hat{s}$ is tagged by several annotators. Entity types can be identified with atomic propositions, and the ontology relationships can be considered as propositional constraints – e.g. if C and D are disjoint entity types, our logical theory includes the constraint $C \rightarrow \neg D$. Thus we can translate the ontology Ω to a propositional theory T_Ω. We say that an annotator *supports* an entity type C if it tags the span $\hat{s}$ with (a subclass of) C. Dually, we say that an annotator *opposes* when it tags $\hat{s}$ with a class disjoint from C or when an annotator fails to tag $\hat{s}$ with (a superclass of) C that is in its vocabulary (opposition via omission). We associate to each identified type C an integer value $\mathsf{AtomicScore}(C)$, representing the degree of support for or opposition to C by annotators. The general form of our scoring function is:

$$\begin{aligned}\mathsf{AtomicScore}(C) = \Sigma_{\mathsf{A}\in\mathsf{Anns}}\\ \Sigma_{D\sqsubseteq C\in\Omega}\mathsf{SupportWeight}_{\mathsf{A},D}\cdot\mathsf{Support}(\mathsf{A},D)\\ -\Sigma_{D\sqcap C=\bot\in\Omega}\mathsf{SupportWeight}_{\mathsf{A},D}\cdot\mathsf{Support}(\mathsf{A},D)\\ -\Sigma_{C\sqsubseteq D\in\Omega}\mathsf{OmitWeight}_{\mathsf{A},D}\cdot\mathsf{Omit}(\mathsf{A},D)\end{aligned}$$

Above, Anns denotes the set of annotations, $D \sqsubseteq C \in \Omega$ indicates that from the rules of ontology Ω one can prove D is a subclass of C, and $D \sqcap C = \bot \in \Omega$ indicates that Ω implies disjointness of D and C. $\mathsf{Support}(\mathsf{A}, D)$ is 1 if annotator A tags the span s with D, and is 0 otherwise. $\mathsf{Omit}(\mathsf{A}, D)$ is 1 iff A has D in its vocabulary, but failed to

tag span s with D. $\mathsf{SupportWeight}_{\mathsf{A},D}$ and $\mathsf{OmitWeight}_{\mathsf{A},D}$ are non-negative $[0,1]$-valued weights that indicate how much weight the tagging of A with D or the omission of D by A should have.

Given the atomic scores, a boolean combination of entity types that is consistent with the ontology is computed. Our *weighted repair* (WR) algorithm first takes the union of all entity types returned by any annotator, which can be considered as a conjunction of entity types σ_{Init}. A repair operation Op is either a deletion of an entity type occurring as a conjunct within σ_{Init} or an insertion of a entity type that is absent from σ_{Init}. The application of Op to σ_{Init} produces a new formula. For a deletion of class C, it is formed by removing every conjunct corresponding to a subclass of C while adding the negation of C, while for an insertion it is formed by conjoining with a proposition corresponding to C. A set of repairs is *internally-consistent* if no two operations conflict, e.g., we do not delete a class C and also insert a subclass of C. For an internally-consistent set of repairs $S = \{\mathsf{Op}_1 \ldots \mathsf{Op}_n\}$, the application on σ_{Init}, denoted $S(\sigma_{\mathsf{Init}})$ is defined as the result of applying the Op_i in any order. A repair set is *non-redundant* if we do not delete or insert two entity types in a subclass relation. A *solution* is an internally-consistent, non-redundant repair set S such that $S(\sigma_{\mathsf{Init}})$ is consistent with T_Ω.

Our goal is to find a solution with maximal *aggregate score* among all solutions, where the aggregate score of a solution S is:

$$\Sigma_{\mathsf{Ins}(C)\in S}\mathsf{AtomicScore}(C) - \Sigma_{\mathsf{Del}(C)\in S}\mathsf{AtomicScore}(C)$$

That is, an operation that deletes an entity type C incurs the penalty $\mathsf{AtomicScore}(C)$, while an insertion of an entity type C incurs the negative of $\mathsf{AtomicScore}(C)$ as a penalty.

Since multiple repairs can achieve the maximal score, we impose ranking criteria: (i) Given two solutions with the same score and different numbers of repairs, we prefer the smaller one. (ii) Given solutions $S_1 = S' \cup \{\mathsf{Ins}(C_1)\}$ and $S_2 = S' \cup \{\mathsf{Ins}(C_2)\}$, with C_2 a subclass of C_1, we prefer S_2, i.e., the one that inserts more specific classes.

ROSEANN computes the optimal solution by reducing the above optimization problem to integer linear programming (ILP). With reference to the example of Figure 3.1(a), WR returns as output a solution with a single Language annotation since it is logically consistent and also with less opposition from the other annotators. On the other hand, MEMM returns a solution with an annotation of type Nationality. MEMM learns from the training set that there is a correlation between the annotations Country and Language provided by EXTRACTIV and SPOTLIGHT, and the entity type Nationality.

3.3 Evaluation

Trivially, it is clear that the usage of aggregation increases the coverage of entities. What distinguishes the aggregators would be their accuracy. We thus focus on evaluating their accuracy in this study. Averages of the precision, recall, and F-score are given, measured in the ontology-aware way given in Section 2.3.2. The averages are performed in two standard ways: a *macro-average* that averages first over each concept, and then over concepts, and a *micro-average* that averages over each annotation (thus giving more weight to the concepts that are more highly represented in the test set)(see Table 3.1 for detail definitions).

Table 3.1: Macro and Micro F-score definitions

	MICRO-AVERAGING	MACRO-AVERAGING
PRECISION	$\frac{\sum_{i=1}^{\|C\|} TP_i}{\sum_{i=1}^{\|C\|}(TP_i+FP_i)}$	$\frac{1}{C}\sum_{i=1}^{\|C\|}\frac{TP_i}{TP_i+FP_i}$
RECALL	$\frac{\sum_{i=1}^{\|C\|} TP_i}{\sum_{i=1}^{\|C\|}(TP_i+FN_i)}$	$\frac{1}{C}\sum_{i=1}^{\|C\|}\frac{TP_i}{TP_i+FN_i}$

We carried out an extensive experimental evaluation of our aggregation methods. A first set of experiments has been devoted to demonstrate and quantify the benefit of aggregation over individual annotators. A second set of experiments compares the various aggregation techniques, including those provided by the state-of-the-art competitors such as FOX [33] and NERD [74].

3.3.1 Individual vs. aggregated

We evaluated our aggregation methods using the same datasets and experimental setting which have been described in Section 2.3.1. We compared the performance of both WR and MEMM against individual annotators and against a naïve Baseline aggregation that simply collects all annotations from individual annotators and, after mapping the returned concepts to the global ontology, returns the union of all annotations regardless of consistency. Table 3.2 to Table 3.12 report the 10-fold cross-validation results of each aggregation approach against each of the individual annotators, where highlighted values represent the best performance. Since the focus is on accuracy (and not, e.g., vocabulary coverage), the comparison for each annotator is made *only* w.r.t. the concepts that are in the annotator's vocabulary – thus a different set of key concepts for each annotator.

Our first observation is that both of our aggregation methods consistently outperform individual annotators in F-score except for OPENCALAIS and ALCHEMY. OPENCALAIS and ALCHEMY have a slightly higher F-score (i.e., 3% better on average) than any

of the aggregators on the Illinois corpus and Reuters corpus respectively. However, it is worth noting that their vocabularies represent only 18% of all concepts in the gold standard. On average, the aggregation via WR results in a 15% increase in performance, while MEMM produces an average of 22% increase with a peak of 66% improvement over ZEMANTA. As expected, the Baseline aggregation delivers worse overall performance than our aggregation methods, with a relatively high recall but a drastically lower precision, worse than any of the individual annotators.

Supervised vs. unsupervised. MEMM delivers the best performance among the aggregators. We see from the figures that for MUC7, with rich training examples over a small number of target concepts, MEMM delivers high *micro-average* and *macro-average* F-scores (mostly above 90%). There is a degradation in *macro-average* scores in the case of concepts recognised by OPENCALAIS and EXTRACTIV. The decrease is caused by some key concepts which have limited training data support in our corpus. For example, PhoneNumber has very few instances in the news-oriented Reuters corpus (a total of 36 occurrences). Since the examples are extremely sparse in a training set, MEMM has only 24% in F-score over PhoneNumber, while the other annotators are able to achieve around 80%. It is unsurprising that individual annotators can do well on phone numbers, given that they can be mostly recognised via regular expressions.

In general, MEMM learns which annotator to trust at a concept-level through global training. Consider, e.g., the key concept Person: the performance of individual annotators varies within a wide range in terms of F-score – that is from 14% to 93% on MUC7 (see Table 2.4); 26% for ZEMANTA, 43% for SPOTLIGHT, 63% for ALCHEMY, 75% for EXTRACTIV and 91% for OPENCALAIS on Reuters. MEMM learns to trust the one with more reliable behaviour over Person – namely, OPENCALAIS– and thus achieves a comparatively high accuracy (94% and 89% on two dataset respectively). There are extreme cases where MEMM does better than all annotators put together: for example, MEMM, on CommercialOrg improves the accuracy by more than 20% against individual annotators and more than 10% against the other two aggregators that are aware of the concept. This is possible because the annotators often correctly annotate Organisations, and the probability of an Organization being a CommercialOrg is high; by detecting this pattern, MEMM can notice many CommercialOrg instances. Obviously, this kind of pattern is highly data-dependent, and in scaling the dataset out this particular inference may not longer hold (and hence would not be applied by MEMM given sufficient training). But the result does show that when patterns within the data or the annotators do exist, they can be detected by our aggregator.

WR has precision competitive with MEMM. This is because it believes in a concept C only if the majority of the "judges" who know about C labelled the span with C or one of

its subclasses – indeed, such a signal turns out to be strongly correlated with correctness. When all the annotators are competent over a certain concept, the recall is also comparable with individual annotators, and in some cases is superior. For example, the majority of the annotators are competent on concept Country, with F-score above 75%, with only ZEMANTA being significantly lower. For this concept WR achieves a precision of 90% while still having recall comparable with the best annotator, OPENCALAIS. However a more common situation is when the recall of individual annotator is really low. In this scenario WR has much lower recall, since most of the instances are considered to lack support. For the concept Person discussed above, only OPENCALAIS and EXTRACTIV have good recall, which lowers the recall, and hence the F-score, of WR.

The key advantage of WR is resilience to sparsity in the dataset. We see from the example of PhoneNumber (see the discussion of MEMM above) that MEMM performs quite poorly when training data is sparse, while WR has score comparable with individual annotators.

Table 3.2: ALCHEMY vs. Aggregators

		MicroPrec.	MicroRec.	MicroF1	MacroPrec.	MacroRec.	MacroF1
MUC7	MEMM	95.4	87.9	**91.5**	95.1	86.9	**90.8**
	WR	72.5	84.1	77.9	77.3	82.9	80.0
	ALCHEMY	85.9	45.4	59.4	76.4	43.8	55.2
	Baseline	44.1	84.5	57.9	54.7	83.3	66.0
Illinois	MEMM	76.9	54.4	63.0	71.3	50.1	58.4
	WR	50.6	78.1	61.2	52.8	71.8	60.4
	ALCHEMY	77.6	60.1	**67.3**	72.8	57.2	**63.7**
	Baseline	40.3	79.0	52.8	41.2	75.0	52.8
Fox	MEMM	91.9	88.5	**90.0**	91.3	89.2	**90.1**
	WR	71.4	87.5	78.6	71.9	88.7	79.3
	ALCHEMY	90.2	80.7	85.0	89.6	80.9	84.9
	Baseline	34.8	90.3	50.1	36.1	90.7	51.3
Reuters	MEMM	85.0	78.7	**81.7**	71.1	59.4	**64.5**
	WR	68.4	82.4	74.8	61.0	64.3	62.6
	ALCHEMY	79.5	58.9	67.7	61.7	43.9	51.2
	Baseline	42.8	89.4	57.9	42.7	75.6	54.5

Table 3.3: SPOTLIGHT vs. Aggregators

		MicroPrec.	MicroRec.	MicroF1	MacroPrec.	MacroRec.	MacroF1
MUC7	MEMM	96.2	93.9	**95.0**	96.4	94.3	**95.3**
	WR	70.2	91.0	79.2	72.6	91.3	80.8
	SPOTLIGHT	69.4	37.4	48.5	70.2	36.2	47.7
	Baseline	40.7	91.3	56.3	43.3	91.5	58.8
Illinois	MEMM	76.9	54.4	**63.0**	71.3	50.1	58.4
	WR	50.6	78.1	61.2	52.8	71.8	**60.4**
	SPOTLIGHT	30.3	14.5	19.0	29.8	16.6	19.8
	Baseline	40.2	79.0	52.8	41.2	75.0	52.8
Fox	MEMM	91.9	88.5	**90.0**	91.3	89.2	**90.1**
	WR	71.4	87.5	78.6	71.9	88.7	79.3
	SPOTLIGHT	77.3	37.6	50.2	76.8	37.6	50.2
	Baseline	34.8	90.3	50.1	36.1	90.7	51.3
Reuters	MEMM	84.9	77.5	**81.1**	63.6	51.4	**56.7**
	WR	63.8	82.8	72.0	54.2	55.6	54.8
	SPOTLIGHT	66.0	44.6	53.1	45.0	32.0	37.2
	Baseline	36.6	88.1	51.6	38.2	65.7	48.2

Table 3.4: EXTRACTIV vs. Aggregators

		MicroPrec.	MicroRec.	MicroF1	MacroPrec.	MacroRec.	MacroF1
MUC7	MEMM	95.4	85.2	**90.0**	94.3	77.6	**85.1**
	WR	73.3	81.5	77.2	85.9	74.6	79.8
	EXTRACTIV	86.6	68.3	76.4	91.6	64.0	75.3
	Baseline	45.0	82.0	58.1	72.6	75.3	73.9
Illinois	MEMM	76.9	54.4	**63.0**	71.3	50.1	58.4
	WR	50.6	78.1	61.2	52.8	71.8	**60.4**
	EXTRACTIV	55.7	34.9	40.4	50.2	35.5	40.9
	Baseline	40.2	79.0	52.8	41.2	75.0	52.8
Fox	MEMM	91.9	88.5	**90.0**	91.3	89.2	**90.1**
	WR	71.4	87.5	78.6	71.9	88.7	79.3
	EXTRACTIV	73.5	79.2	76.1	73.7	77.9	75.6
	Baseline	34.8	90.3	50.1	36.1	90.7	51.3
Reuters	MEMM	84.4	76.9	**80.5**	69.4	59.7	64.2
	WR	71.8	76.0	73.8	68.9	65.4	**67.0**
	EXTRACTIV	71.5	69.4	70.4	64.8	61.6	63.2
	Baseline	52.1	81.6	63.5	62.0	71.9	66.6

Table 3.5: IllinoisNer vs. Aggregators

		MicroPrec.	MicroRec.	MicroF1	MacroPrec.	MacroRec.	MacroF1
MUC7	MEMM	96.2	93.9	**95.0**	96.4	94.3	**95.3**
	WR	70.2	91.0	79.2	72.6	91.3	80.8
	IllinoisNer	85.8	83.0	84.3	85.9	84.1	84.9
	Baseline	40.7	91.3	56.3	43.3	91.5	58.8
Illinois	MEMM	76.9	54.4	63.0	71.3	50.1	58.4
	WR	50.6	78.1	61.2	52.8	71.8	**60.4**
	IllinoisNer	57.1	73.0	**63.7**	54.4	67.7	60.1
	Baseline	40.2	79.0	52.8	41.2	75.0	52.8
Fox	MEMM	91.9	88.5	**90.0**	91.3	89.2	**90.1**
	WR	71.4	87.5	78.6	71.9	88.7	79.3
	IllinoisNer	87.0	85.9	86.4	86.1	86.2	86.1
	Baseline	34.8	90.3	50.1	36.1	90.7	51.3
Reuters	MEMM	87.9	79.9	**83.7**	87.7	80.4	**83.9**
	WR	60.8	88.5	72.0	61.4	89.0	72.7
	IllinoisNer	81.6	67.4	73.8	81.3	69.3	74.8
	Baseline	31.6	94.8	47.4	32.7	94.9	48.7

Table 3.6: LUPEDIA vs. Aggregators

		MicroPrec.	MicroRec.	MicroF1	MacroPrec.	MacroRec.	MacroF1
MUC7	MEMM	96.2	93.9	**95.0**	96.4	94.3	**95.3**
	WR	70.2	91.0	79.2	72.6	91.3	80.8
	LUPEDIA	81.4	23.5	36.4	75.3	23.7	36.0
	Baseline	40.7	91.3	56.3	43.3	91.5	58.8
Illinois	MEMM	76.9	54.4	**63.0**	71.3	50.1	58.4
	WR	50.6	78.1	61.2	52.8	71.8	**60.4**
	LUPEDIA	42.1	13.3	19.9	43.2	13.5	19.7
	Baseline	40.2	79.0	52.8	41.2	75.0	52.8
Fox	MEMM	91.9	88.5	**90.0**	91.3	89.2	**90.1**
	WR	71.4	87.5	78.6	71.9	88.7	79.3
	LUPEDIA	90.4	51.8	65.6	86.7	47.2	60.9
	Baseline	34.8	90.3	50.1	36.1	90.7	51.3
Reuters	MEMM	84.9	77.5	**81.1**	63.6	51.4	**56.7**
	WR	63.8	82.8	72.0	54.2	55.6	54.8
	LUPEDIA	74.5	34.3	46.9	43.2	21.5	23.5
	Baseline	37.4	89.9	52.8	37.9	67.9	48.6

Table 3.7: OPENCALAIS vs. Aggregators

		MicroPrec.	MicroRec.	MicroF1	MacroPrec.	MacroRec.	MacroF1
MUC7	MEMM	95.4	88.1	**91.6**	95.4	88.3	**91.7**
	WR	73.0	84.3	78.2	81.1	84.7	82.9
	OPENCALAIS	90.6	61.2	73.0	85.1	65.0	73.5
	Baseline	44.6	84.8	58.5	62.5	85.2	72.1
Illinois	MEMM	76.9	54.4	**63.0**	71.3	50.1	58.4
	WR	50.6	78.1	61.2	52.8	71.8	**60.4**
	OPENCALAIS	72.6	51.5	59.6	60.3	44.8	51.0
	Baseline	40.3	79.0	52.8	41.2	75.0	52.8
Fox	MEMM	91.9	88.5	**90.0**	91.3	89.2	**90.1**
	WR	71.4	87.5	78.6	71.9	88.7	79.3
	OPENCALAIS	80.1	81.9	80.9	78.6	81.6	79.8
	Baseline	34.8	90.3	50.1	36.1	90.7	51.3
Reuters	MEMM	81.2	73.0	**76.9**	69.8	55.4	61.6
	WR	66.4	78.8	72.0	60.1	67.1	63.4
	OPENCALAIS	80.2	71.6	75.6	69.1	58.9	**63.5**
	Baseline	49.3	85.7	62.6	48.5	77.2	59.5

Table 3.8: StanfordNer vs. Aggregators

		MicroPrec.	MicroRec.	MicroF1	MacroPrec.	MacroRec.	MacroF1
MUC7	MEMM	96.2	93.9	**95.0**	96.4	94.3	**95.3**
	WR	70.2	91.0	79.2	72.6	91.3	80.8
	StanfordNer	95.7	88.5	92.0	95.6	89.5	92.5
	Baseline	40.7	91.3	56.3	43.3	91.5	58.8
Illinois	MEMM	76.9	54.4	63.0	71.3	50.1	58.4
	WR	50.6	78.1	61.2	52.8	71.8	60.4
	StanfordNer	82.3	67.9	**74.0**	77.2	63.3	**68.9**
	Baseline	40.2	79.0	52.8	41.2	75.0	52.8
Fox	MEMM	91.9	88.5	**90.0**	91.3	89.2	**90.1**
	WR	71.4	87.5	78.6	71.9	88.7	79.3
	StanfordNer	87.5	85.7	86.4	88.7	87.2	87.7
	Baseline	34.8	90.3	50.1	36.1	90.7	51.3
Reuters	MEMM	87.9	79.9	**83.7**	87.7	80.4	**83.9**
	WR	60.8	88.5	72.0	61.4	89.0	72.7
	StanfordNer	86.1	73.8	79.4	85.4	74.7	79.7
	Baseline	31.6	94.8	47.4	32.7	94.9	48.7

Table 3.9: Saplo vs. Aggregators

		MicroPrec.	MicroRec.	MicroF1	MacroPrec.	MacroRec.	MacroF1
MUC7	MEMM	96.2	93.9	**95.0**	96.4	94.3	**95.3**
	WR	70.2	91.0	79.2	72.6	91.3	80.8
	Saplo	67.9	35.8	46.8	68.3	38.0	48.8
	Baseline	40.7	91.3	56.3	43.3	91.5	58.8
Illinois	MEMM	76.9	54.4	**63.0**	71.3	50.1	58.4
	WR	50.6	78.1	61.2	52.8	71.8	**60.4**
	Saplo	45.6	33.0	37.7	41.6	29.1	33.1
	Baseline	40.2	79.0	52.8	41.2	75.0	52.8
Fox	MEMM	91.9	88.5	**90.0**	91.3	89.2	**90.1**
	WR	71.4	87.5	78.6	71.9	88.7	79.3
	Saplo	73.1	59.7	65.4	74.3	59.4	65.6
	Baseline	34.8	90.3	50.1	36.1	90.7	51.3
Reuters	MEMM	87.9	79.9	**83.7**	87.7	80.4	**83.9**
	WR	60.8	88.5	72.0	61.4	89.0	72.7
	Saplo	68.4	38.2	49.0	65.7	37.5	47.7
	Baseline	31.6	94.8	47.4	32.7	94.9	48.7

Table 3.10: YahooQL vs. Aggregators

		MicroPrec.	MicroRec.	MicroF1	MacroPrec.	MacroRec.	MacroF1
MUC7	MEMM	96.2	93.9	**95.0**	96.4	94.3	**95.3**
	WR	70.2	91.0	79.2	72.6	91.3	80.8
	YahooQL	87.1	11.4	20.1	86.8	11.4	20.2
	Baseline	40.7	91.3	56.3	43.3	91.5	58.8
Illinois	MEMM	76.9	54.4	**63.0**	71.3	50.1	58.4
	WR	50.6	78.1	61.2	52.8	71.8	**60.4**
	YahooQL	78.9	20.9	32.5	63.5	20.3	29.8
	Baseline	40.2	79.0	52.8	41.2	75.0	52.8
Fox	MEMM	91.9	88.5	**90.0**	91.3	89.2	**90.1**
	WR	71.4	87.5	78.6	71.9	88.7	79.3
	YahooQL	84.1	35.8	49.8	81.6	41.3	54.1
	Baseline	34.8	90.3	50.1	36.1	90.7	51.3
Reuters	MEMM	83.9	75.9	**79.7**	73.9	60.2	**66.2**
	WR	63.6	81.3	71.4	64.4	63.6	63.9
	YahooQL	76.3	5.9	11.0	28.6	2.8	5.1
	Baseline	39.1	90.2	54.6	46.5	77.7	58.1

Table 3.11: Wikimeta vs. Aggregators

		MicroPrec.	MicroRec.	MicroF1	MacroPrec.	MacroRec.	MacroF1
MUC7	MEMM	96.1	90.1	**93.0**	93.8	78.4	**85.4**
	WR	70.8	87.2	78.2	81.9	75.4	78.5
	Wikimeta	62.9	40.6	49.3	42.4	25.0	31.4
	Baseline	40.7	91.3	56.3	43.3	91.5	58.8
Illinois	MEMM	76.9	54.4	**63.0**	71.3	50.1	58.4
	WR	50.6	78.1	61.2	52.8	71.8	**60.4**
	Wikimeta	42.3	33.8	36.8	52.2	33.3	39.9
	Baseline	40.2	79.0	52.8	41.2	75.0	52.8
Fox	MEMM	91.9	88.5	**90.0**	91.3	89.2	**90.1**
	WR	71.4	87.5	78.6	71.9	88.7	79.3
	Wikimeta	74.1	67.2	70.2	79.1	66.1	71.6
	Baseline	34.8	90.3	50.1	36.1	90.7	51.3
Reuters	MEMM	85.3	75.0	**79.8**	72.7	59.1	**65.1**
	WR	60.8	83.0	70.1	54.5	68.5	60.7
	Wikimeta	61.1	38.2	47.0	39.8	24.6	30.4
	Baseline	37.3	89.2	52.6	40.9	77.9	53.6

Table 3.12: ZEMANTA vs. Aggregators

		MicroPrec.	MicroRec.	MicroF1	MacroPrec.	MacroRec.	MacroF1
MUC7	MEMM	96.2	93.9	**95.0**	96.4	94.3	**95.3**
	WR	70.2	91.0	79.2	72.6	91.3	80.8
	ZEMANTA	71.6	19.7	30.8	76.2	17.8	28.8
	Baseline	40.7	91.3	56.3	43.3	91.5	58.8
Illinois	MEMM	76.9	54.4	**63.0**	71.3	50.1	58.4
	WR	50.6	78.1	61.2	52.8	71.8	**60.4**
	ZEMANTA	47.4	13.9	20.1	37.5	15.5	19.1
	Baseline	40.2	79.0	52.8	41.2	75.0	52.8
Fox	MEMM	91.9	88.5	**90.0**	91.3	89.2	**90.1**
	WR	71.4	87.5	78.6	71.9	88.7	79.3
	ZEMANTA	71.1	64.5	67.3	68.7	62.8	65.2
	Baseline	34.8	90.3	50.1	36.1	90.7	51.3
Reuters	MEMM	84.7	76.0	**80.1**	62.0	49.8	**55.0**
	WR	67.1	80.2	73.1	53.1	48.9	50.8
	ZEMANTA	69.0	29.4	41.1	51.1	20.5	29.1
	Baseline	41.7	87.0	56.4	40.0	60.0	47.9

3.3.2 Comparative evaluation

We compared the performance of our aggregation methods against the FOX [33] and NERD [74] NER aggregators (discussed in Section 3.7) on all datasets.

Although our focus is on Web-based annotators, the comparative evaluation requires the use of the same annotators used by FOX, namely the STANFORDNER and ILLINOIS-NER (both software-based), as well as the web-based annotators ALCHEMY. Figure 3.5 to Figure 3.12 summarize the results, which clearly show that our aggregation techniques are at worst comparable with FOX and NERD and in average outperform the competitor aggregators of more than 15% in F-score on the MUC7, Illinois, and Reuters datasets. An interesting case is the Illinois corpus that suffers from high data sparsity. In fact, not only does NERD show a better behaviour, but also MEMM is slightly outperformed by WR on this dataset.

3.3.3 Performance and scalability

The last set of experiments studies the amount of resources required to compute a solution for the integration. Figure 3.13 and Figure 3.14 plot the (per-document) average computation time for both WR and MEMM w.r.t. an increasing number of annotators (2 to 11) and for different corpora. For WR, an optimal solution for a single document can be produced on the order of 300 milliseconds in the worst-case. In addition, WR's performance is directly correlated with the number of annotated spans to be processed,

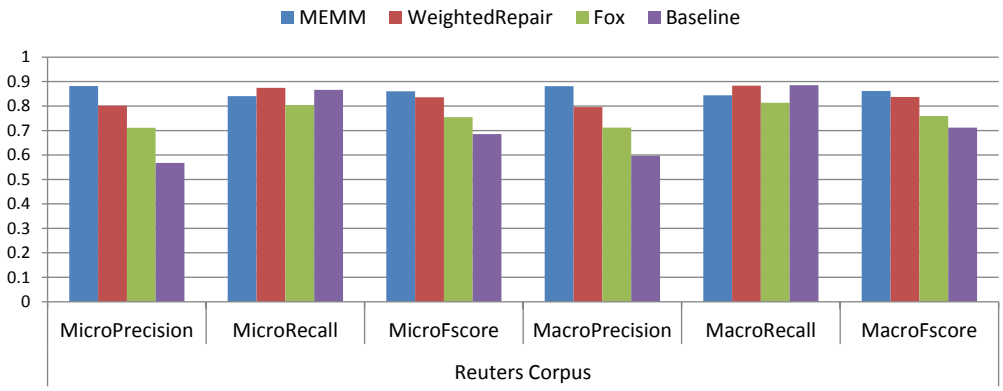


Figure 3.5: Aggregators vs. Fox on Reuters

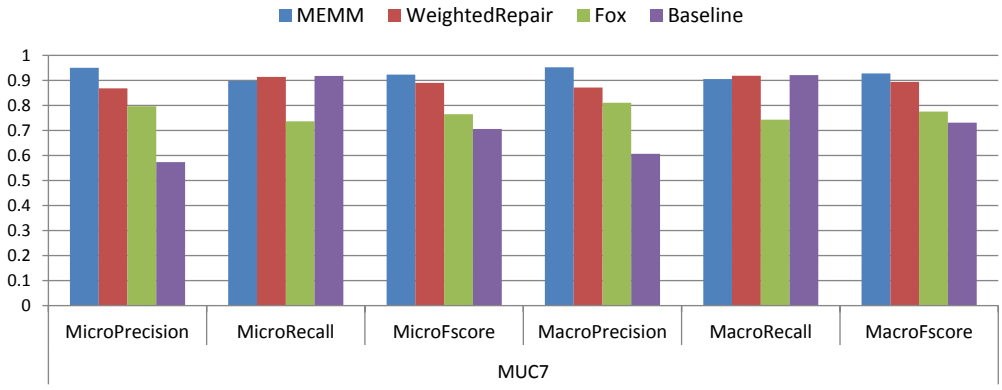


Figure 3.6: Aggregators vs. Fox on MUC7

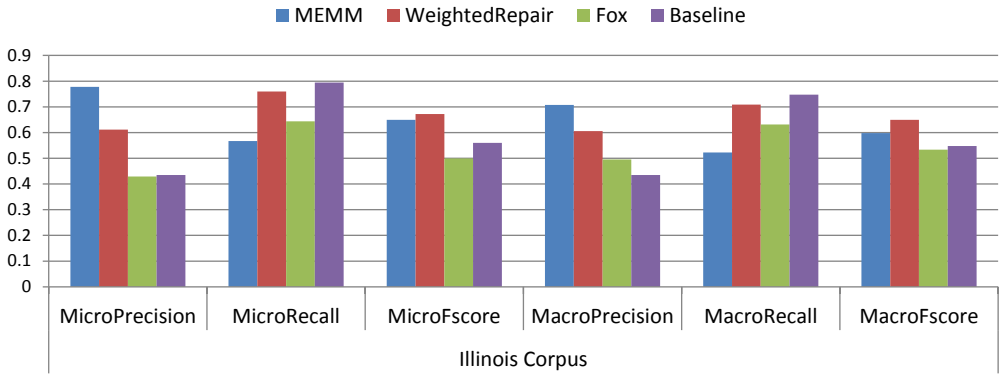


Figure 3.7: Aggregators vs. Fox on Illinois

and that is considerably higher in the MUC7, Illinois corpus and Reuters corpora than in the Fox corpus. This is different from MEMM, where the time to compute an optimal solution is affected by the number of concepts to be taken into account and on which MEMM has been trained. On the Reuters corpus, solutions can be computed on the order of hundreds of milliseconds while on the other corpora, where the number of concepts

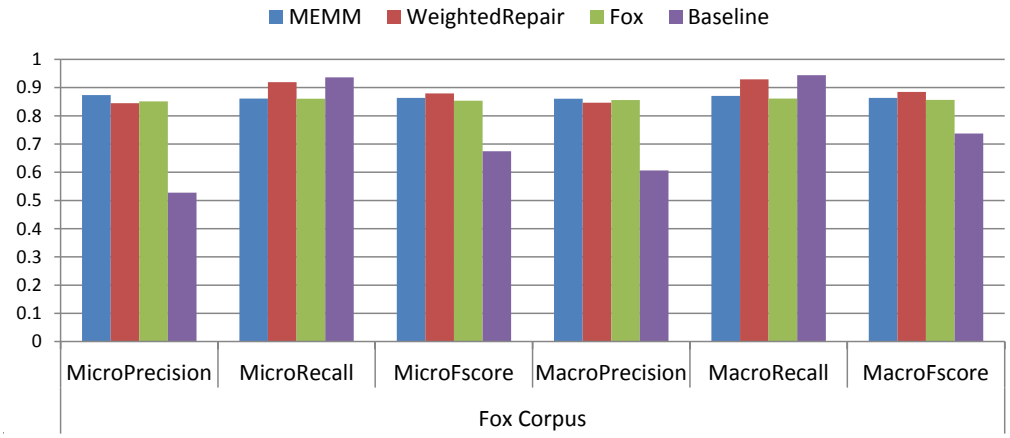


Figure 3.8: Aggregators vs. FOX on Fox

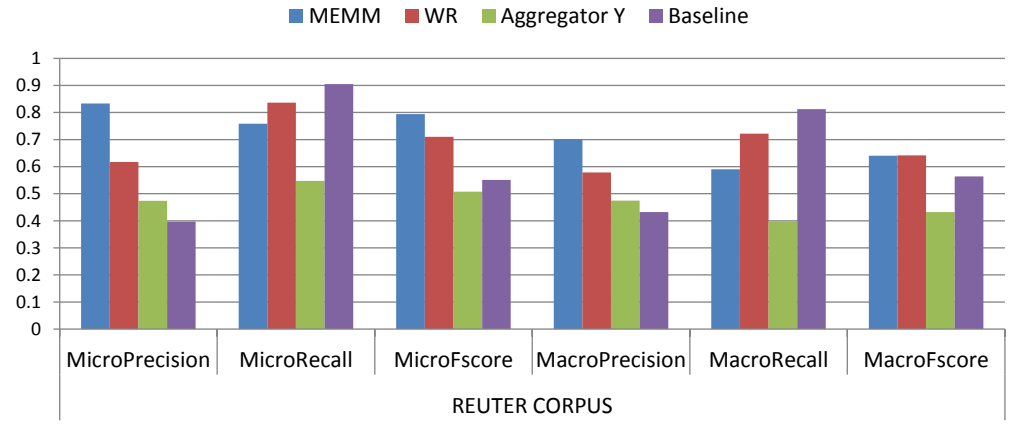


Figure 3.9: Aggregators vs. NERD on Reuters

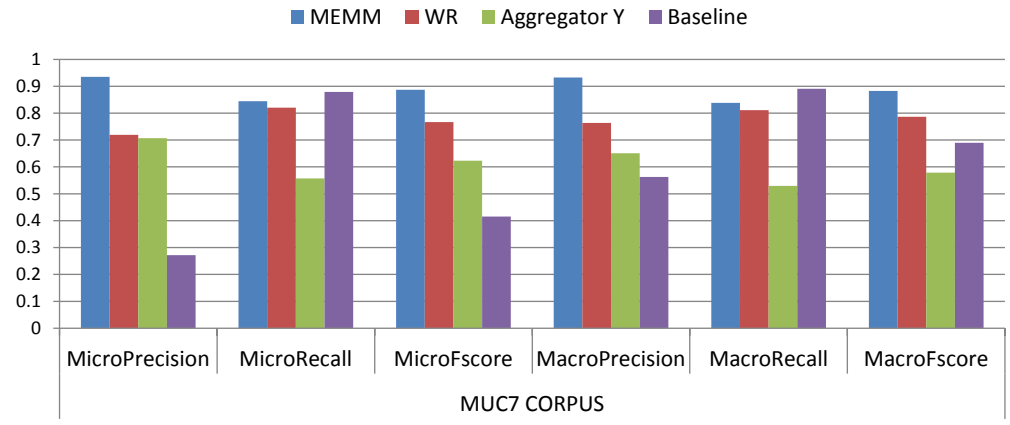


Figure 3.10: Aggregators vs. NERD on MUC7

involved is lower, MEMM's performance is comparable with WR.

Figure 3.15 summarizes the scalability on the training time of MEMM. It is worth

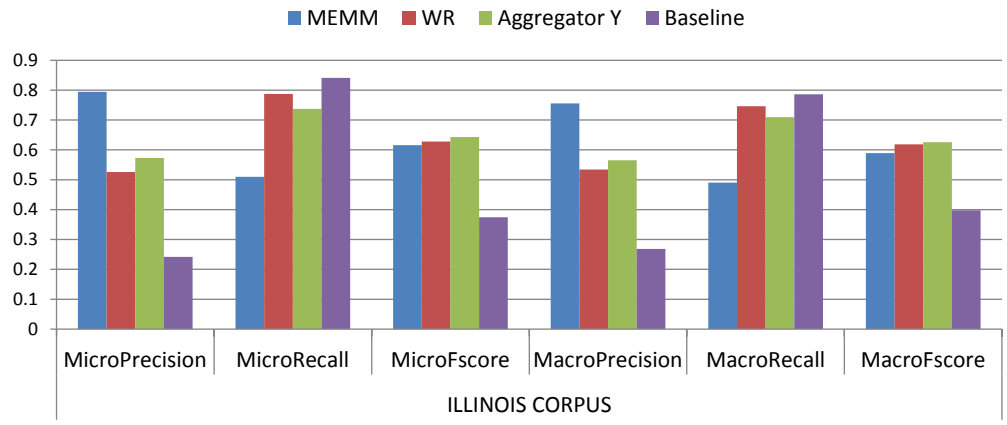


Figure 3.11: Aggregators vs. NERD on Illinois

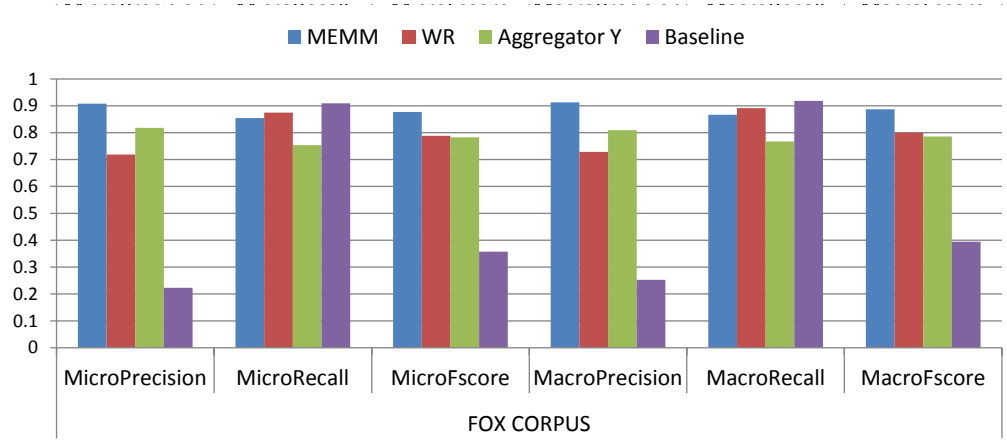


Figure 3.12: Aggregators vs. NERD on Fox

mentioning that training MEMM takes between 68 msec (2 annotators and 3 concepts on the Fox corpus) and 2 mins (11 annotators and 215 concepts on the Reuters corpus). Overall, the time required to produce an optimal solution with MEMM (resp. WR) is dominated by at least one (resp. two) orders of magnitude by the time required to send the various requests to the online annotation services and collect the results. As an example, EXTRACTIV takes more than 60 seconds to annotate a medium-sized (i.e., 3-4KB) document. For both MEMM and WR memory consumption is negligible if compared with the size of the collected annotations.

Overall, MEMM shows its superiority both against individual annotators and the existing competitors. This is particularly true when there are sufficient representative training data available. One limitation that MEMM may suffer from is the sparsity issue for which extra strategies are required. We will further investigate this in Section 3.4.

Table 3.13: Summary of corpus size per document.

Test Corpus	Min. Length	Max. Length	Avg. Length	Average Size (KB)
MUC7	1,414	8,108	4,498	4.1
Reuters	943	8,876	3,035	3.2
Fox	387	11,713	2,365	0.2
Illinois	83	477	4,719	2.3

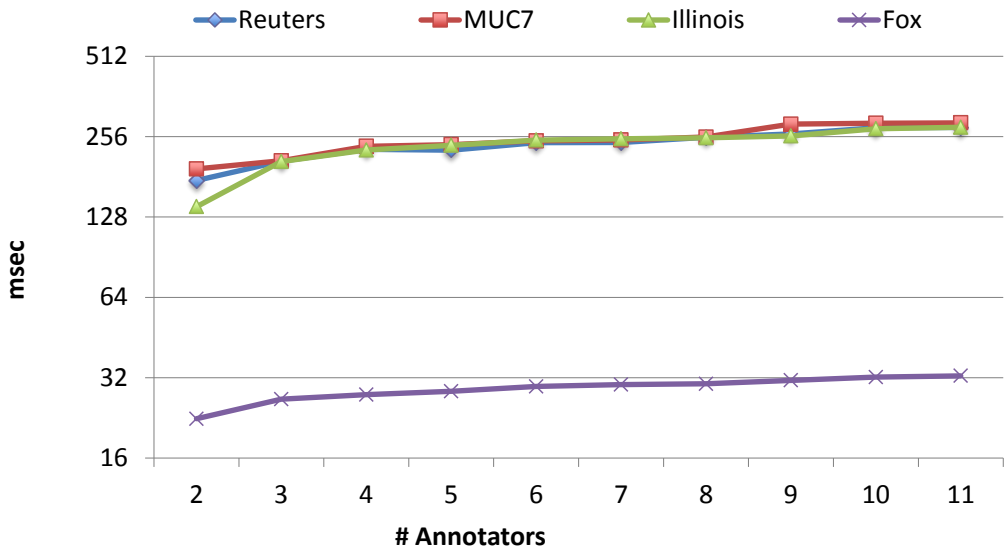


Figure 3.13: Scalability of WR

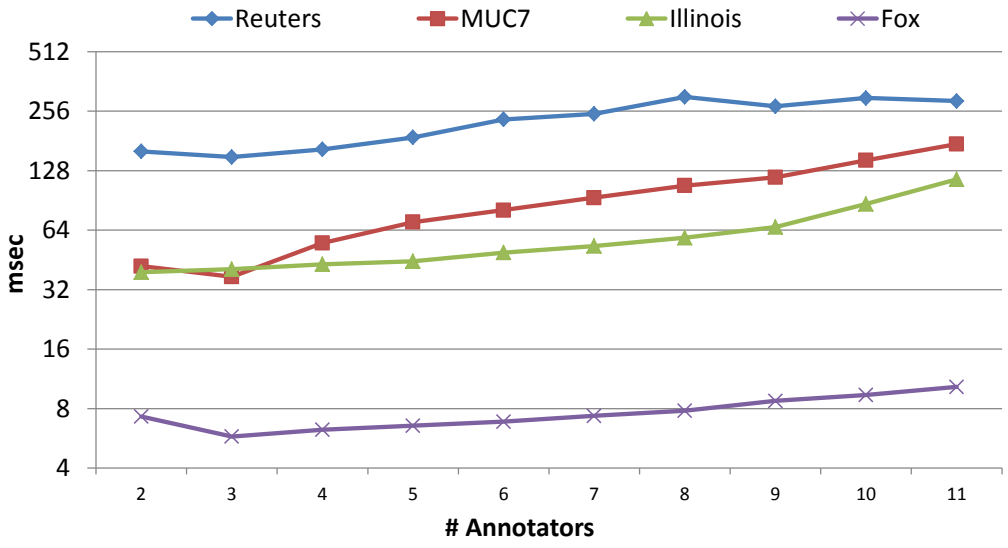


Figure 3.14: Scalability of MEMM Testing

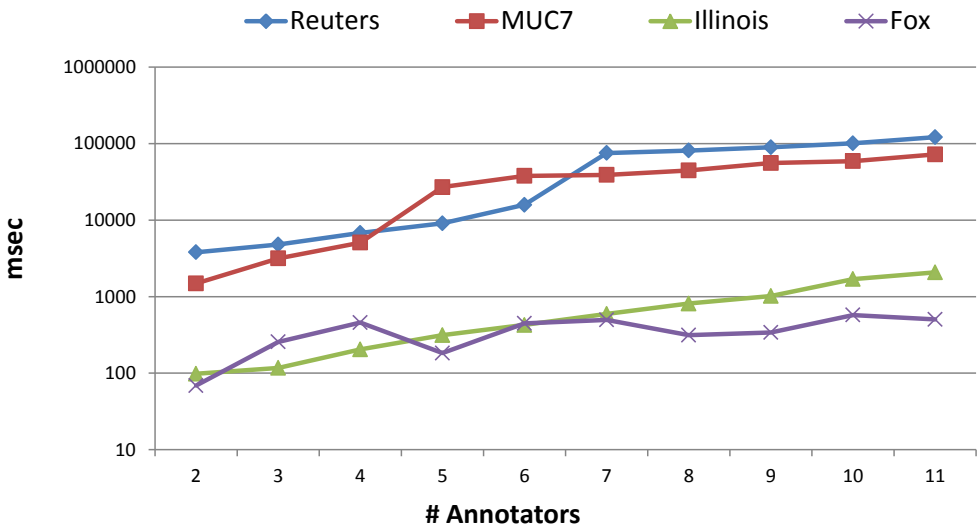


Figure 3.15: Scalability of MEMM Training

3.4 Alternatives to MEMM

Section 3.2.2 describes a supervised aggregation approach, which uses a Maximum Entropy Markov model to capture the probability distribution of the reconciled labelling sequence and utilizes the VITERBI algorithm to provide the most likely output. However, using VITERBI on top of a probabilistic annotator may intuitively seem like a very crude approach. We are discarding all the quantitative information from the probabilistic annotator, and we are ignoring all the answers other than the best one. In practice the top-1 answer may not always be correct due to the issues such as being weakly trained by the limited support of training data which makes the model unable to capture the original probability distribution.

Consider the example shown in Figure 3.16. Given the pruned Markov sequence of the TCC "this week", the VITERBI-based MEMM fails to output the correct labelling sequence $\langle\ B_RelativeDate,\ I_RelativeDate \wedge B_Calendar\rangle$. Instead, it returns the path in bold as the most likely answer. We can observe from the markov sequence that the transition from the initial state Begin is quite uncertain given the input evidence of the token "this" (the maximum outgoing probability is only 0.01) because of the rare input evidence. During the VITERBI decoding, the model returns the path with the highest likelihood 0.009×1.0: $\langle\ B_Event \wedge B_MedicalFacility,\ B_RelativeDate\rangle$ which is obviously wrong.

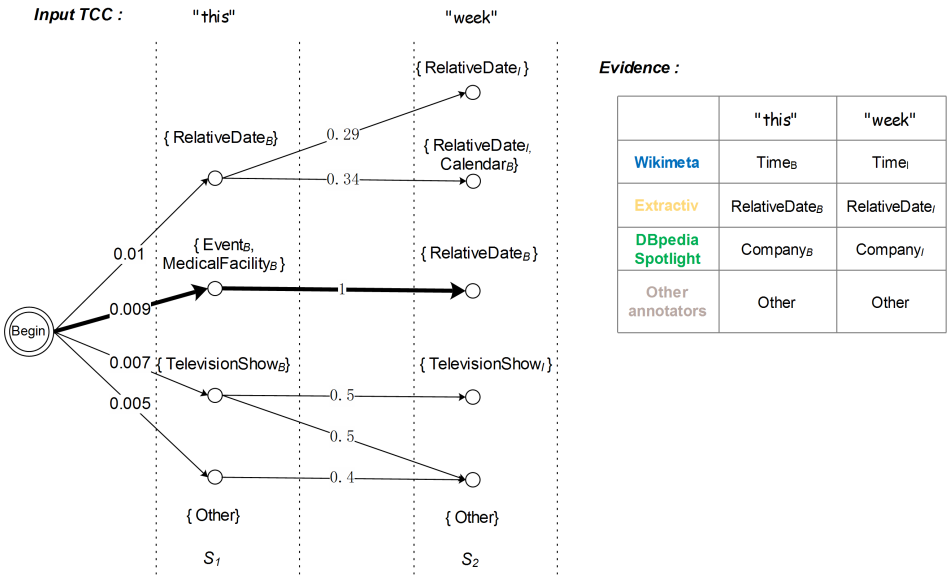


Figure 3.16: An example of incorrect top 1 answer

Based on the above discussion, we want to further investigate whether we can do better by either keeping more answers or retaining the quantitative information about

the uncertainty. In this section, we consider several possibilities as alternatives to the version implemented in Section 3.2.2.

3.4.1 MEMM with multiple answers

Instead of returning a single deterministic answer as VITERBI does, we first consider returning multiple answers, given a Markov sequence and a specific position within it. Let $\mathrm{T}^{(m)}$ be the matrix of which each entry is the m-step transition probability ($m \geq 1$), such that the entry in the ith row and jth columns is $\mathrm{T}^{m}_{i,j}$. Let $\mathrm{T}_{\tau_{m\rightarrow}}$ be the one-step transition matrix at position m which can be obtained directly from the transition function $\tau_{m\rightarrow}$ (see Definition 3):

$$\mathrm{T}_{\tau_{m\rightarrow}} = \begin{pmatrix} \tau_{m\rightarrow}(q_1, q_1) & \tau_{m\rightarrow}(q_1, q_2) & \cdots & \tau_{m\rightarrow}(q_1, q_n) \\ \tau_{m\rightarrow}(q_2, q_1) & \tau_{m\rightarrow}(q_2, q_2) & \cdots & \tau_{m\rightarrow}(q_2, q_n) \\ \vdots & \vdots & \ddots & \vdots \\ \tau_{m\rightarrow}(q_n, q_1) & \tau_{m\rightarrow}(q_n, q_2) & \cdots & \tau_{m\rightarrow}(q_n, q_n) \end{pmatrix}$$

Then we calculate $\mathrm{T}^{(m)}$ as follows by induction on m :

$$\mathrm{T}^{(m)} = \mathrm{T}_{\tau_{m\rightarrow}} \cdot \mathrm{T}^{(m-1)}$$

We can further compute the probability distribution of states after m steps by applying the following formula:

$$\mathrm{Pr}^{m} = \mathrm{T}_{\tau_{0\rightarrow}} \cdot \mathrm{T}^{(m)}$$

Here $\mathrm{T}_{\tau_{0\rightarrow}} = \langle \tau_{0\rightarrow}(q_1), \cdots, \tau_{0\rightarrow}(q_n) \rangle$ is a vector of size n representing the initial distribution of the states at time 0. If we want to know the probability of ending in state q after m steps, we just get the corresponding entry $\mathrm{Pr}^{m}(q)$ from the resulting vector Pr^{m} as the required answer.

Threshold-based decoding (MEMM-threshold). Given a l-long Markov sequence and a threshold σ as inputs, the decoder returns annotations whose concepts have a probability over σ. Specifically, it scans over the input Markov sequence: for each position m ($1 \leqslant m \leqslant l$) which is the beginning of a TCC, it calculates the m-step probability distribution of states Pr^{m} ; then it aggregates the probability SCORE for each atomic concept C ; finally, it returns the concepts whose score lies above the given threshold. Such annotations will be treated as the annotations overlaying the span of the TCC. Here the aggregated probability is computed as below:

$$\textsc{Score}(C) = \sum_{q \in Q \wedge \pi_C(q) = C} \mathrm{Pr}^{m}(q) \tag{3.4}$$

, where Q is the state space and $\pi_C(q)$ means projecting the state q to the concept C (recall that each state is a boolean combination of atomic concepts with B-I decoration). Example 7 illustrates how to calculate the aggregated score for an atomic concept using the probability distribution of states in a fixed position.

Example 7. Take the Markov sequence in Figure 3.16 as an example, in position 2 (corresponding to the token "week"), $\textsc{Score}(\mathsf{RelativeDate}) = \Pr^2(\mathsf{I_RelativeDate}) + \Pr^2(\mathsf{I_RelativeDate} \wedge \mathsf{B_Calendar}) + \Pr^2(\mathsf{B_RelativeDate}) = 0.01 \times 0.29 + 0.01 \times 0.34 + 0.009 \times 1 = 0.015$. □

Top-*k* concepts decoding (MEMM-topk-concepts). Instead of truncating concepts based on an absolute numeric threshold, a MEMM-topk-concepts decoder returns concepts whose aggregated probability Score is the top- k highest. Here Score for each atom concept is calculated in the same way as before. For example, by applying the top-*1* decoder on the Markov sequence in Figure 3.16, RelativeDate will be returned as the output annotation of "this week", since $\textsc{Score}(\mathsf{RelativeDate}) = 0.01$ which is the highest among all the states of S_1.

Experiments. In order to validate the usefulness of these alternatives compared with the VITERBI version, we use the Reuters corpus described in Section 2.3.1 whose vocabulary forms a hierarchical taxonomy. All the reconciliation algorithms in the rest of this section use the 11 individual annotators mentioned in Section 2.3.3 and the experiments are conducted by means of 10-fold cross validation. We compare these methods against the VITERBI-based MEMM (MEMM-Viterbi) using the same measurement (e.g., precision and recall) as we did before. For MEMM-topk-concepts and MEMM-threshold, due to their inherent preference to general concepts, we compute the F-score using 14 top-level concepts which are direct descendants of the root.

Figure 3.17 and Figure 3.18 show the comparison results of the two alternatives. Each approach is evaluated with varying parameters ($\sigma = 0.01, 0.2$ and $k = 1, 3, 10$). Overall, there is always a tradeoff between the performance and the number of answers, which is controlled by the input parameters such as the score thresholds σ and k. With a larger k (e.g., $k = 10$) or smaller σ (e.g., $\sigma = 0.01$), we are able to boost the recall (e.g., obtaining a minimal improvement of 10%). However, the precision drops significantly, where the lose is around 30% for MEMM-threshold and 50% for MEMM-topk-concepts considering both Micro and Macro scores. When some "medium" values are chosen, e.g., $\sigma = 0.2$ and $k = 3$, MEMM-threshold and MEMM-topk-concepts perform similarly to MEMM-Viterbi in terms of Micro and Macro F-scores.

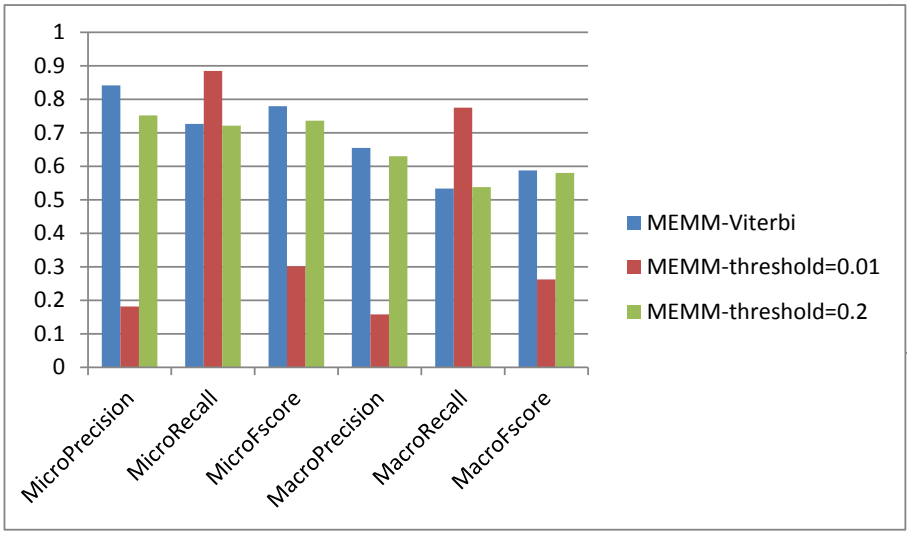


Figure 3.17: MEMM-Viterbi vs. MEMM-threshold

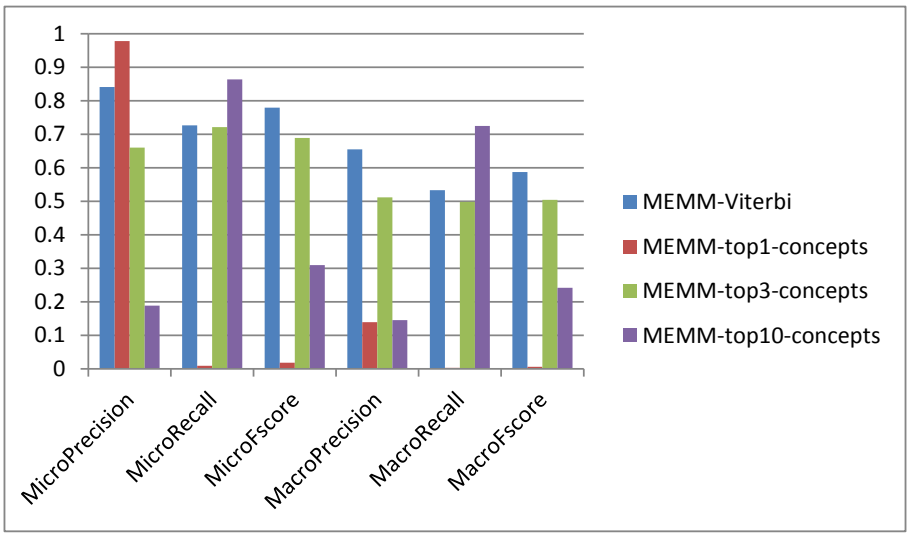


Figure 3.18: MEMM-Viterbi vs. MEMM-topk-concepts

3.4.2 MEMM with quantitative information

In addition to the above approaches, we also consider retaining the quantitative information about uncertainty which could be generated by the Markov sequence. We first consider the method to keep all the probabilities of the labelling sequences, then discuss the ones that keep only the top k highest probabilities of the answers.

MEMM yielding probabilities (MEMM-prob). For each position m in the input Markov sequence, if the position is the beginning of a TCC, we then calculate SCORE for all the atomic concepts by applying Equation 3.4 and keep the list of pairs $\langle C, \text{SCORE}(C)\rangle$ for each atomic concept C as the output of the TCC.

Top-k boolean combination decoding (MEMM-topk-bool). While intuitively MEMM-threshold and MEMM-topk-concepts will prefer high-level concepts in the taxonomy as they can always draw the scores from their subclasses, the MEMM-topk-bool decoder returns the boolean combinations whose probability is the top-k highest, where each boolean combination is associated with a renormalized probabilities over the k-highest probabilities. In order to make annotations well-formed, we still work at the level of TCCs instead of individual tokens. For example, MEMM-top1-bool outputs the result that consists of the best annotations at the beginning token of a TCC, while MEMM-Viterbi takes the best path of the entire TCC.

Example 8. Consider again the example TCC in Figure 3.16. When $k = 2$, concepts $\{\langle \mathsf{RelativeDate}, 0.53\rangle, \langle \mathsf{Event}, 0.47\rangle, \langle \mathsf{MedicalFacility}, 0.47\rangle\}$ will be returned, where $\textsc{Score}(C) = \frac{0.01}{0.01+0.009} \approx 0.53$. □

Experiments. Two datasets described in Section 2.3.1 are selected for MEMM-prob: the Reuters corpus which has comparatively richer concepts and the Illinois corpus on which MEMM-Viterbi performs poorly as reported in the previous section. For MEMM-topk-bool, only the Reuters dataset is tested since the method is motivated by the scenario of complex vocabularies. As these alternatives output probabilistic answers, we adopt the *square error* [37] as a comparison indicator, which is defined as $\varepsilon = (1-p)^2$ where p is the probability of the correct answer. Specifically, if the answer from VITERBI is correct, the error is 0, otherwise it is 1. We calculate the square error for all the TCCs. See below for an example of how to compute the square error.

Example 9. In Figure 3.16, RelativeDate is one correct annotation of the TCC. MEMM-prob returns the annotation RelativeDate with the probability 0.01, so we have $(1-0.01)^2$ for the square error. MEMM-Viterbi misses the correct answer, thus it gets an error of 1. □

Table 3.14: MEMM-prob vs. MEMM-Viterbi on overall square error

	Reuters Corpus	Illinois Corpus
MEMM-Viterbi	0.2269	0.34
MEMM-prob	**0.187**	**0.30**
MEMM-top1-bool	0.2274	-
MEMM-top3-bool	**0.186**	-
MEMM-top10-bool	**0.184**	-

Table 3.14 summarizes the overall square error of the testing datasets, where from MEMM-Viterbi to MEMM-prob the square error drops around 4% on both datasets.

Table 3.15: Square error on the concepts of the Illinois corpus

Concepts	Cardinality	$\varepsilon - Viterbi$	$\varepsilon - prob$
Person	180	0.328	**0.272**
Location	65	0.492	**0.482**
Organisation	162	0.630	**0.562**
Other	905	0.190	**0.179**
Misc	141	0.972	**0.825**

In terms of MEMM-topk-bool, MEMM-top1-bool performs similarly to MEMM-Viterbi, while MEMM-top3-bool and MEMM-top10-bool achieve roughly the same improvement as MEMM-prob does. Table 3.15 shows the square error w.r.t. the concepts in Illinois corpus, where for each concept we calculate error by only considering all the real occurrences of an instance belonging to this concept. The average error reduction from MEMM-Viterbi to MEMM-prob is about 6% over these concepts. The improvement indicates that keeping the quantitative information may lead to more accurate results, especially when models have to be trained using limited training data (e.g., the Illinois corpus).

3.4.3 MEMM using additional features

MEMM using co-occurrence features (MEMM-co). While ordinary MEMM only extracts features from local evidence of a TCC, the idea of MEMM-co is to use as features the co-occurrence of another entity in the context of the current token. The motivation is that, intuitively, a SportEvent is more likely to "meet" an Athlete rather than a Musician. Due to the Markov property, we thus look for the co-occurrence prior to the given token which means using the annotation results of previous TCCs as features in the prediction of the current TCC. Note that during training, we use the gold-standard annotations which are prior to the current token. We also implement a version named MEMM-co-perfect where in the prediction phase, the prior entities come from the gold-standard rather than the prediction results used by MEMM-co.

Experiments. We compare MEMM-co with MEMM-Viterbi on Reuters corpus again. As shown in Figure 3.19, there is no significant improvement from the perspective of the whole set of concepts, even for the "ideal" case –MEMM-co-perfect. One possible reason is that not all the prior entities are discriminative in our testing scenario which focuses on a generic newswire domain. For example, in the Reuters corpus, Country and City occur in the context of every docs, as do most of the high-level entities. However, we can see that if we zoom in to a subset which includes only concepts that are "tightly-semantically-related" (e.g., many disjointness and containment constraints connecting

them), we can see a benefit from using co-occurrence features. For example, Figure 3.20 shows the accuracy improvement on the concepts related to sports by applying the entity co-occurrence features.

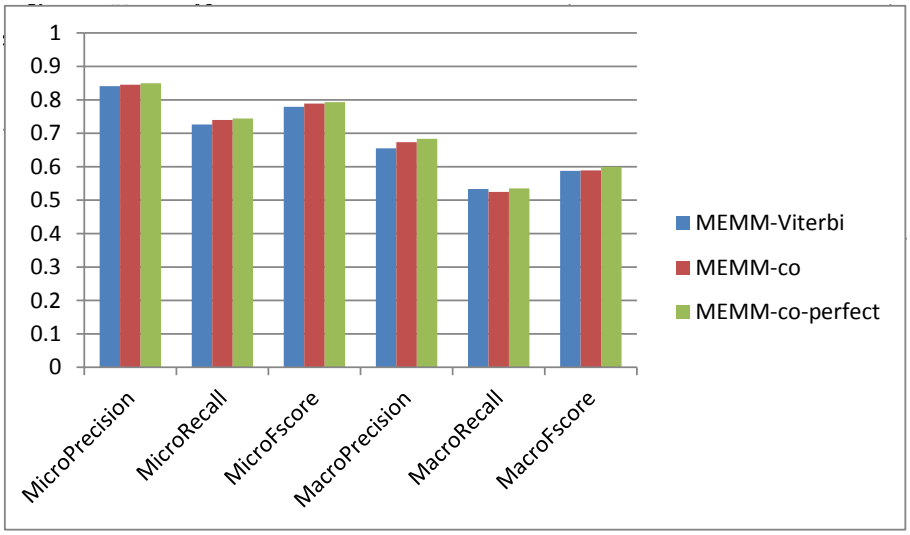


Figure 3.19: MEMM-Viterbi vs. MEMM-co

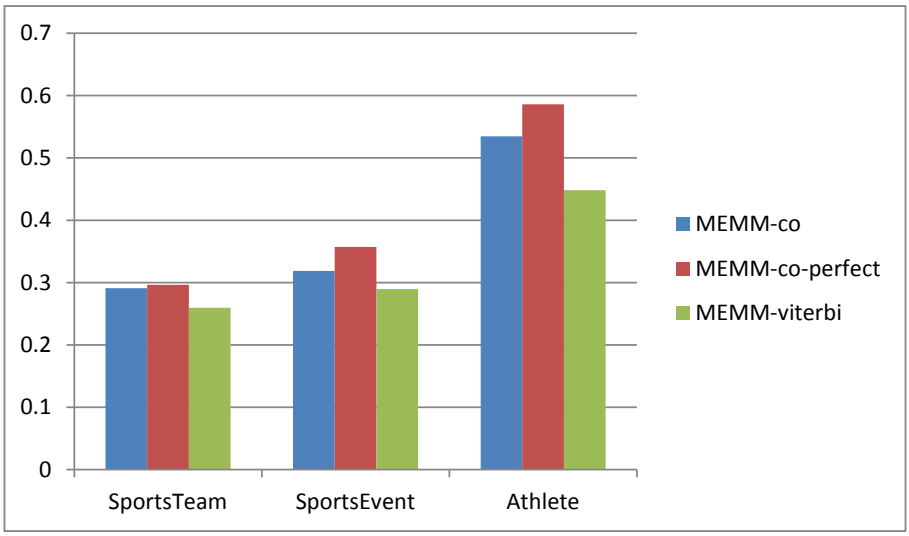


Figure 3.20: MEMM-Viterbi vs. MEMM-co on F-score of sports-related concepts

3.4.4 MEMM using hybrid models

MEMM using backup models (MEMM-backup). As discussed before, the input evidence can be quite sparse. Figure 3.21 shows the amount of support on the evidence features in the Reuters training set, where the features are the individual opinions of

the base annotators, and the co-occurrence of pairwise opinions between different annotators. We can observe from Figure 3.21 that most of the features suffer from limited support (e.g., $\approx 94\%$ features have support below 100). We thus consider switching off VITERBI and instead turning on certain "backup" models in the scenarios where some of the evidence is sparse. The condition of measuring the sparsity of evidence is based on empirical statistics along with a given threshold. The "backup" models could either be probabilistic models (e.g., MEMM-prob) or other reconciliation algorithms (e.g., WR).

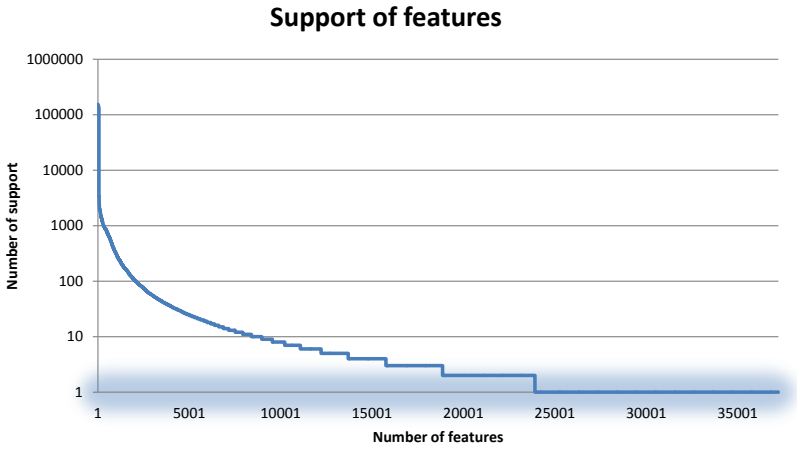


Figure 3.21: Statistics about evidence support

Experiments. We implement a version of MEMM-backup which will turn on the WR answers if encountering *sparse evidence* during the procedure of transducing a Markov sequence, denoting this as MEMM-backup-wr. More specifically, the sparse evidence is simply defined to be the input evidence vector with *at least* one opinion of individual annotators whose support is below a given threshold θ. For the evaluation, we select 23 sparse concepts from the global vocabulary whose entity cardinality is small in our training set (e.g. ≤ 50) and show how the approach works compared with the original MEMM-Viterbi by varying the threshold θ, e.g., let $\theta = 10, 50$.

The preliminary results are compared in Figure 3.22, from which the performance of detecting these sparse concepts shows a significant gain, e.g., a maximum 16% increase on Micro F-score and 9% on Macro F-score. Table 3.16 also lists a selected subset of results based on concepts. The results show that applying MEMM-backup-wr can make use of the key advantage of WR, which is resilience to sparsity in the dataset. Considering the example of PhoneNumber, the backup model achieves 0.79 in F-score which is even better than WR itself does.

In summary, as shown from our empirical results: (i) keeping more answers does not outperform MEMM-Viterbi using the conventional evaluation matrix; (ii) keeping

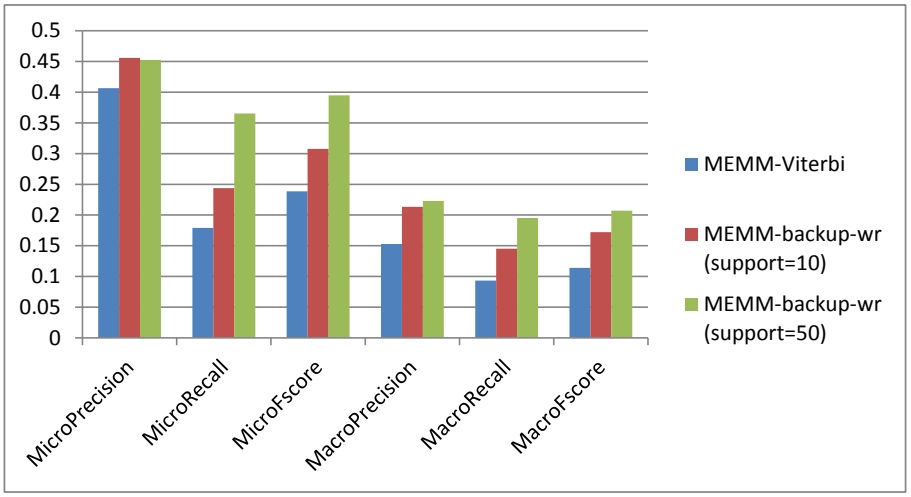


Figure 3.22: MEMM-backup-wr vs. MEMM-Viterbi on sparse concepts in Reuters corpus

Table 3.16: MEMM-backup-wr vs. MEMM-Viterbi on F-score of selected sparse concepts

Concepts	Cardinality	MEMM-Viterbi	MEMM-backup-wr ($\theta = 10$)	MEMM-backup-wr ($\theta = 50$)	WR
StockExchange	16	0	0.187	0.253	0.347
PhoneNumber	36	0.242	0.327	0.79	0.782
Mayor	7	0	0.1	0.1	0.173
Island	49	0.13	0.263	0.217	0.286
MilitaryOrg	16	0.1	0.15	0.16	0.177
RadioStation	11	0	0.167	0.167	0.143
ScreenActor	39	0.129	0.169	0.176	0.178

quantitative information about the uncertainty is able to provide more accurate answers than MEMM-Viterbi does; (iii) using entity context information can help the recognition of domain-specific entities; (iv) and applying hybrid models such as combining MEMM-Viterbi and WR will boost the performance in aggregating opinions of sparse evidence. Note that we are not aiming for a "perfect" alternative to MEMM-Viterbi fitting all circumstances. The investigation in this section is for providing a broader view of applying MEMM-based techniques to the problem of reconciliation. Naturally, the usage of these different variations can depend on the concrete application scenarios.

3.5 The ROSeAnn system

In this section, we overview the ROSeAnn system for the management of semantic annotations, based on the algorithmic ideas in the previous sections. ROSeAnn provides users with a unified view over the opinion of multiple independent online and standalone

annotators, linking them to an integrated ontology of their vocabularies. It supports document formats such as plain text documents and live web pages. It allows users to understand and reconcile conflicts between annotations via *ontology-aware aggregation*. ROSEANN incorporates both the supervised aggregator MEMM proposed in Section 3.2.2, and the unsupervised aggregator WR in Section 3.2.3.

3.5.1 System architecture

The ROSEANN architecture is shown in Figure 3.23. Before a ROSEANN instance is launched, there are several items which need to be configured: (i) ROSEANN parameters, which specifies all the parameters required such as the pool of base annotators to be aggregated, the parameters of their API calls and the background ontology which defines the constraint axioms of the vocabularies. We list these parameters in a configuration file in XML format; (ii) reasoning services (e.g., via an OWLIM SPARQL endpoint). Users interact with the graphical interfaces of ROSEANN, from which raw documents including plain text and HTML documents are submitted and the corresponding annotated document is displayed.

The ROSEANN core workflow processes documents through the following pipeline: (i) strip the text content from the original documents; (ii) submit the text to individual annotators using the parallel annotating module. Both the Web service-based and standalone annotators are invoked in parallel through the *annotator APIs*; (iii) if an aggregation is requested, then pass the annotated document with individual opinions to *aggregator APIs* where each returns a set of logically-consistent merged annotations; (iv) render and visualize the resulting annotated texts or web documents using the *visualization module*. The *visualization module* relies on JAVA SWING for documents of plain text and PDF. For web documents, *visualization module* takes the mapping from the offset in stripped text to the corresponding pointer of DOM nodes in the HTML document, and defines the CSS properties for a browser to render the page.

ROSEANN currently federates 11 annotators, namely: OPENCALAIS, EXTRACTIV, SPOTLIGHT, ALCHEMY, ZEMANTA, LUPEDIA, WIKIMETA, SAPLO, YAHOOQL, STANFORDNER, and ILLINOISNER. ROSEANN also supports two aggregation methods: MEMM and WR proposed in Section 3.2. Additional base annotators and reconciliation algorithms can be seamlessly added by means of the ROSEANN APIs.

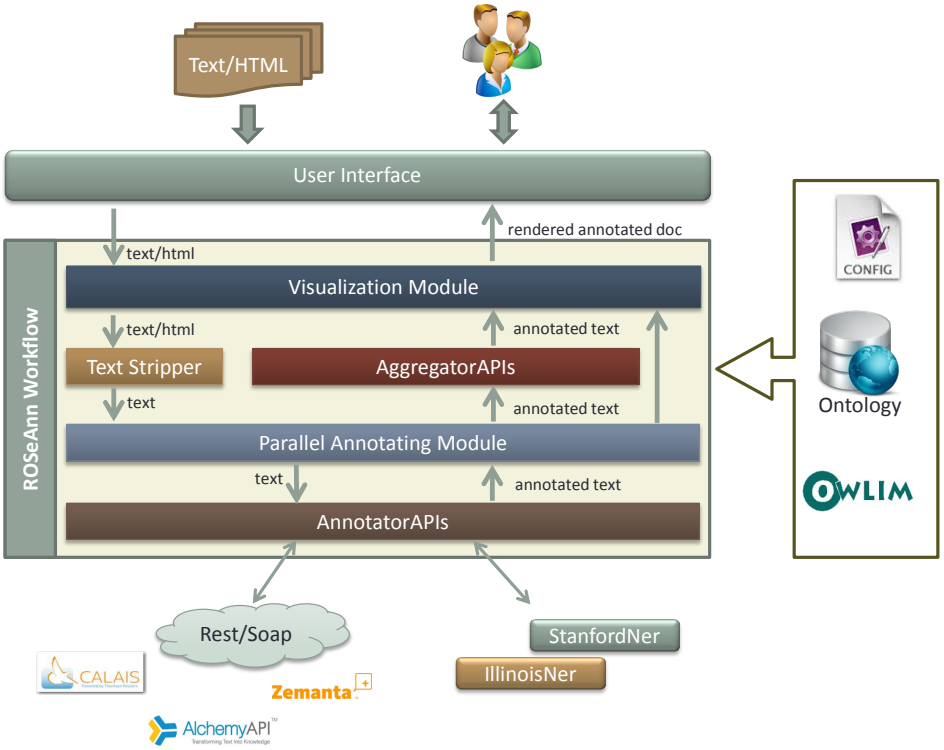


Figure 3.23: ROSeAnn system architechure

3.5.2 The ROSeAnn APIs

3.5.2.1 The ROSeAnn JAVA APIs

ROSeAnn comes with a JAVA API allowing developers to utilize in further applications, where the main abstractions underlying the ROSeAnn API are:

Annotators. This is the basic abstraction in ROSeAnn and it represents a standalone or an online annotator together with its configuration parameters (see Figure 3.24 as the class design). Application developers can instantiate and configure an annotator using parameters such as the type of concepts to return and a timeout, as well as specific parameters of the annotator when known. For those annotators requiring a user key, ROSeAnn allows application developers to add their private keys.

Document models. An abstraction for representing diverse source documents such as plain text documents, PDFs, and HTML documents in a uniform fashion.

Annotated document model. An annotated document model is composed of the source text to be annotated, the set of annotations that return by annotators/aggregators and the set of conflicts. The API supports retrieval of annotations by entity types (e.g., Person), annotators (e.g., OpenCalais) and by document spans, e.g., a [start, end]

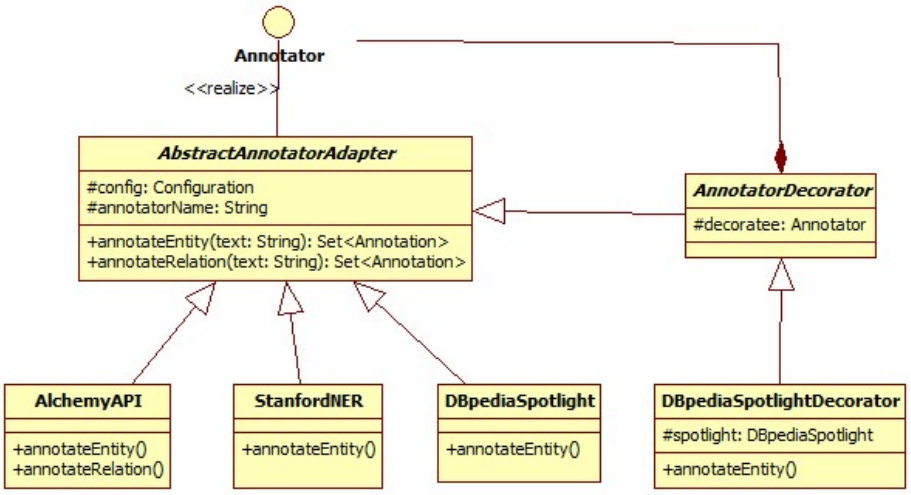


Figure 3.24: ROSeAnn annotators API

character range within the document. Also, application developers can retrieve conflicts by type (e.g., logical and omission), and by document span.

Parallel annotator. Individual annotators can be combined into an annotation pool, an abstraction used to invoke multiple annotators in parallel on a given set of resources (i.e., documents). The advantage of annotation pools is the possibility to share configuration parameters that are common to multiple annotations, thus avoiding the burden of configuring each annotator in the same way. When dealing with HTML documents, ROSeAnn supports the annotation of different sections of the DOM with different annotation pools, e.g., DOM elements, attributes, and scripts. This is particularly useful in certain applications where it is important to distinguish annotations on visible parts of the DOM (e.g., for information extraction), from those on invisible content such as attributes and scripts (e.g., for program analysis). Each annotation-pool is configured with different annotators that are going to be invoked for reconciliation.

Aggregators. The reconciliation of sets of annotations coming from different annotators is carried out via an aggregator. This abstraction represents a reconciliation algorithm, e.g., MEMM. Application developers can extend ROSeAnn with their own aggregation algorithms and then use the ROSeAnn API and GUI to manipulate and display the results. A particular class of aggregators is the trainable aggregator, representing aggregators algorithms requiring training. ROSeAnn provides abstractions to carry out training of aggregators programmatically.

3.5.2.2 The ROSeAnn web APIs

In addition to the JAVA API, we also allow users to make use of ROSeAnn through a web service endpoint. The web service API provides REST endpoints to compute semantic annotations in plain text and web documents. Example API calls are given as below.

```
curl -i -X GET
   http://163.1.88.61:9091/roseann/text?text=Obama+Barack
```

The above invokes ROSeAnn on text annotation with default parameters. As a consequence, only annotators which do not require an API key will be invoked without reconciliation. The default timeout is set to 60 seconds.

Here is an example of a GET request which invokes the web page annotation service of ROSeAnn:

```
curl -i -X GET
   http://163.1.88.61:9091/roseann/web?url=http://www.google.com
```

The response is a JSON object containing, e.g., the status code (successful or not), the annotated text (for DOMs is only the visible text on the web page), the annotators which were able to complete, the reconciliation algorithms which were able to provide a reconciled annotation, the annotations, their conflicts, and those annotations of a type which is unknown to ROSeAnn. Details, such as API documentation, are given in [76].

3.5.3 User interface

ROSeAnn provides a graphical interface (Figure 3.25) enabling users to load both static text and live web documents into the tool. Web navigation is provided by driving a Firefox web browser via Selenium WebDriver[3].

The main menu bar on the top of the GUI (**A**) supports document loading, annotation, a ROSeAnn configuration, as well as the possibility to save the annotated documents and to browse the entity types of the mapping ontology via the ROSeAnn SPARQL Endpoint[4]. Annotated documents are accessible from the left-hand side of the GUI (**B**), together with documents coming from the Reuters and MUC7 corpora, which are used as pre-loaded benchmarks for our evaluation.

Textual documents are visualized in the main text area of the tool (**C**), while web documents are visualised in an independent browser window (**D**). In both cases, the user can interact with the document and the web browser before starting the annotation process.

[3] `http://docs.seleniumhq.org/`.

[4] `http://163.1.88.38:8081/openrdf-workbench/`.

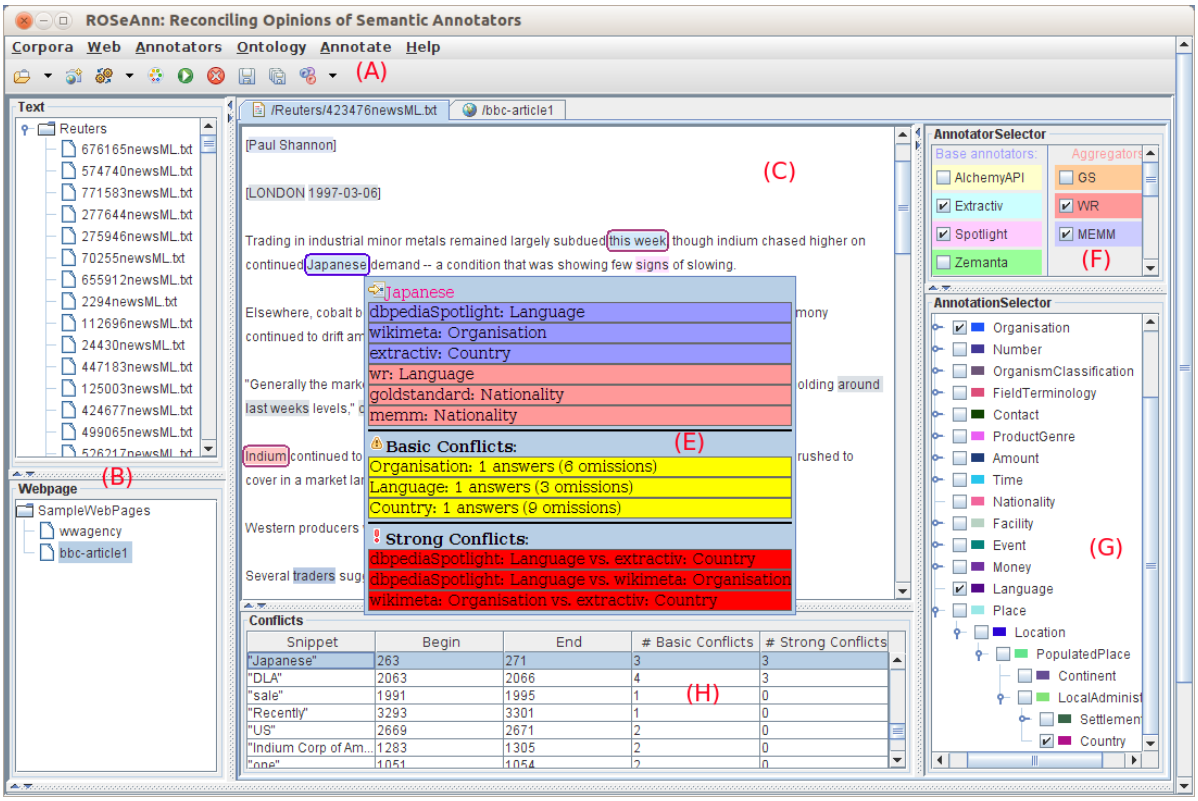


Figure 3.25: ROSeAnn text annotation

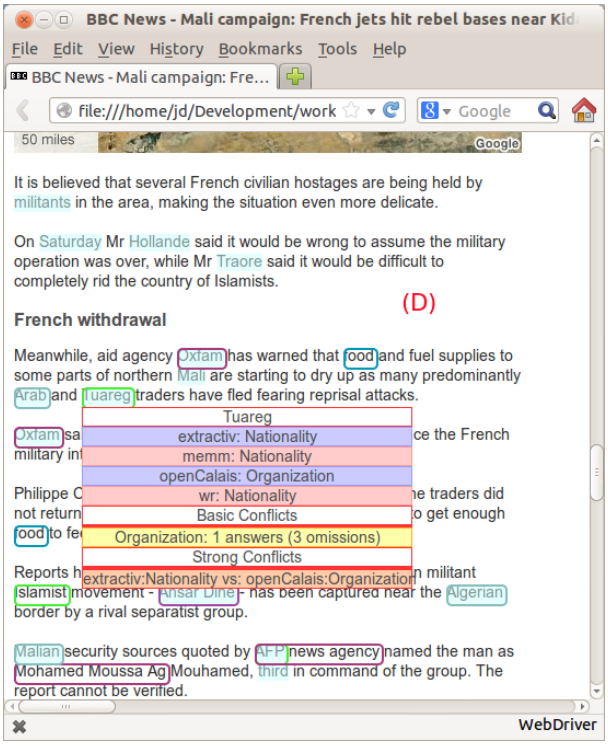


Figure 3.26: ROSeAnn web annotation

After a document has been annotated, ROSeAnn highlights the recognised entities in the main text area or in the web browser. The highlighting consists of a colored border around the identified entities representing an entity type in the ontology, and a colored background representing the annotator or aggregator recognising that particular entity. For documents coming from the Reuters and MUC7 corpora, we also provide the gold-standard annotations. By hovering over the highlighted entities, ROSeAnn provides the list of opinions for all annotators and aggregators specifying which annotator contributed a given entity type, together with the basic and strong conflicts (**E**). A different background on the tooltip is used to distinguish the opinions of annotators from those of the aggregators. When an annotator provides also links to LOD, e.g., to DBpedia, ROSeAnn makes available those anchors to the user in the tooltip.

On the right-hand-side of the GUI we list all annotators (**F**) that identified at least one entity in the current document and the identified entity types organised into a hierarchy (**G**) that corresponds to the structure of our mapping ontology. The user can decide which annotators, aggregators and entity types to visualise in the main text-area or in the browser.

At the bottom of the GUI we provide a table listing all conflicts generated by the annotators in the given document (**H**). In particular, we report (from the left to the right in the table) the text snippet involved in the conflict, the start and end offset of the text span in the document, and the number of basic and strong conflicts occurring on that span. After selecting a row in the conflict table, ROSeAnn blinks the span involved in the conflicts in the text area or in the browser.

3.6 Summary

A MEMM-based approach has been presented for knowledge-aware integration of entity extractors, which considers each entity extractor as a black box and does not assume any prior knowledge about their performance or competence w.r.t. a certain set of entities. As a part of our joint work, an unsupervised aggregation approach WR was also briefly introduced which makes use of local judgement by logical constraints. These voting-based approaches rely heavily on the competence of base annotators and are limited to making judgements locally, while MEMM is able to learn a wide range of patterns between annotators' opinions and outcomes. These approaches have been experimentally compared both with respect to each other and to individual annotators. The results show clear benefits for aggregation of annotation, while also giving new insight into the behaviour of Web-based annotators. We also investigated alternatives to MEMM-Viterbi, such as approaches based on querying the probabilistic annotation generated by the

Markov sequence. In the end of this chapter, the ROSEANN system and its accompanied APIs were introduced.

3.7 Related work

The work in this chapter is rooted in information extraction, but uses techniques from data integration, inconsistency-tolerant reasoning, and truth-finding: we review a few of the most-related works below.

Many previous approaches addressing the integration of multiple NER systems have been based on variations of voting mechanisms [50]. As also noticed in [82], the reliability of voting-based aggregation is strictly connected to the type and the number of the annotators. In particular, annotators have often a very limited coverage of concepts that are claimed to be recognised and this affects the performance of voting mechanisms that do not assume prior-knowledge about single annotators. An obvious way of overcoming this problem is *biasing* the voting mechanism by assigning different weights to the annotators that somehow reflect the confidence in its annotations. In [82] the weights are determined using an exponential model borrowed from meta information retrieval systems [3]. A major problem with this model is that weights are determined on a per-annotator basis without considering that an annotator might have very different performance on different concepts, as also proven by our experimental evaluation. Our unsupervised approach makes use of the ontology in voting, thus distinguishing the annotator-concept pairs. Biasing is also often complemented with thresholding [44], that considers only annotations with a minimum of support. As shown by our experimental evaluation, there are entities that can be correctly typed by a single annotator and thresholding can lead to a noticeable drop in recall. An example of this is the concept CommercialOrg that can only be recognised by EXTRACTIV. Our repair-based approach, in contrast, does not need to consider any arbitrary threshold.

A different way of combining annotators is to use machine learning techniques to determine the optimal combination of annotators. The most popular techniques are neural networks [33], support vector machines [24], classifier *stacking* [92, 95, 32], and conditional random fields (CRFs) [82]. The underlying idea of all these approaches is to combine classifiers in a meaningful way to obtain a composite model. Being generic, these techniques often fail to take into account specific information about the semantics of the text. None of the approaches above uses ontological knowledge to determine logical conflicts caused by the aggregation. A recent approach, part of the NERD project, proved that background knowledge [74] – that already proved beneficial for generating the annotations – can also help in the integration of multiple extractors by locating conflicting

and logically incoherent annotations. However, the NERD ontology does not contain disjointness constraints and therefore these conflicts cannot be automatically detected. Moreover, we are aware that NERD uses machine learning techniques for annotation integration as reported in [25], but we do not know which technique is applied when invoking the NERD web service. Our approach explicitly uses ontological knowledge to locate and discard logically inconsistent opinions from the individual annotators. To the best of our knowledge, our approach is also the first one considering MEMM for semantic annotator aggregation. In future work we are considering how other machine learning methods such as CRFs, can be adapted to take into account ontological rules. There is a vast array of variants of both CRFs and MEMMs (e.g., in the objective function used in training [43]), and so the trade-off between them, even in the absence of semantic considerations, is quite complex. In Section 3.3, we compared our approaches experimentally to the main learning-based approach which was freely available, FOX, we could not find an implementation of [32], which won a CoNLL competition, but we note that it performed experiments on a Reuters corpus similar to the one we tested on, reporting the F-score that are very similar to the ones for both of our approaches.

In either the unsupervised or supervised setting, it is possible to adopt more sophisticated techniques when deep knowledge about the annotators is available. On the learning side, Michelakis et al. [62] look at the combination of the annotation rules adopted by each annotator. As in MEMM, it makes use of a maximal entropy classifier, but with "evidence vectors" being the output of rule-based annotators where the the rule sets are assumed to be available (as is the case in software-based annotators such as GATE). Unlike either of our approaches, semantic relationships within the vocabulary of annotations are not considered. Differently from Si et al. [82], we use only contextual information within a given span – this is much more restricted, but also much less costly. Our unsupervised method is close in spirit to classic judgement aggregation techniques [38], where the final solution captures judgements that are at a minimum aggregate-distance from those of individual experts. However, the traditional judgement aggregation setting assumes a "white-box" model where the methods used by individual experts are transparent to the aggregator.

In a different context – that of webpages asserting facts about entities – work has been done on trying to judge the trust and dependence of sources [23, 35]. For example, iterative algorithms within the PageRank/Hits family are applied in Galland et al. [35] to determine trustworthiness. Here trust/reliability/dependence is naturally attributed globally to a websource, since the assertions being judged are not assumed to belong to a hierarchy of types. We see all these techniques as complementary to our aggregation methods since they exploit redundancy across different web sources to determine true

assertions. It is possible, in principle, to apply these techniques on top of our aggregation algorithm to further validate the annotations across documents deemed to refer to the same facts, e.g., news from different websites describing the same event.

Another interesting setting is the one of SOFIE [87] where common-knowledge rules are applied on top of automatically-generated ontological assertions to determine, discard or repair logically inconsistent facts in the ontology. Again, these techniques are complementary to aggregation when additional knowledge is available from the annotation sources. In particular, common-knowledge rules can be used to improve the recognition of nested annotations, like those in Figure 2.1, by restricting the valid combinations of concepts that can appear in a containment relationship, e.g., the name of a university usually contains also the name of the city in which it is located, while illness names can embed the name of the scientist who discovered them.

Chapter 4

Managing and Querying Semantic Annotations

As mentioned in the introduction, semantic annotations provide rich and useful metedata that enhances the text. In addition, many of these annotations are linked to standard ontologies (e.g. DBpedia and YAGO) which serve as background knowledge repositories. Both the annotations and the external knowledge allow users to explore broader and deeper information beyond the original documents. Thus there is an increasing need for a uniform way to access and manipulate documents overlaid with semantically-enriched metedata. In this chapter, we describe QUASAR, a system for the traversal between unstructured text and the available semantic knowledge. It allows structured queries that meld multiple facets of annotated documents: annotations and their structure, their relationship with document structure, the semantic facts they represent and their inferences within a given ontology. This chapter starts with introducing the data model and the QUASAR language, following by an explanation of the implementation architecture on top of a relational DBMS and an ontology reasoner, and finally presents the user interfaces of the system.

4.1 Motivation

We begin with illustrating the goal in detail through a motivating example. Figure 4.1 shows an article about explorers with diverse semantic annotations, including entity annotations (e.g., "Sabrosa" is a City and "Strait of Magellan" is a Place), sentiment annotations (e.g., snippets in positive tones) and entity relations (e.g., "Ferdinand Magellan" was born in "Sabrosa"). In addition to the associated semantic categories, many of the annotations can be linked to standard ontologies showing external background knowledge. For example, City is a subclass of Place and "Magellan" refers to the famous traveller.

Consider the motivating query in the free text of Figure 4.1, which is "What places has Magellan visited?". The query shows how a user may focus on fine-grained information needs. Observe that the desired answers are located within the same paragraphs with the person "Magellan", which include some "explicit" places such as "Strait of Magellan" and "implicit" places such as islands and countries. In order to retrieve the above answers, a system is required to search by taking into consideration the innate document structure, along with the "external knowledge" such as an *ontology* about those semantic annotations that are overlaid (e.g., DBpedia in Chapter 2).

However, existing approaches are incapable of providing satisfactory solutions. On the one hand, general purpose database solutions may have difficulties in uniformly modelling and intuitively querying the above information. For example, we could store the annotated documents and external knowledge in traditional relational databases and apply SQL to query all the information, but in this case we lack the support of making use of implicit knowledge from reasoning. It is possible to make use of a SPARQL query engine with built-in reasoners (e.g., Virtuoso [90]) to answer the query, where all the information is modelled as RDF triples. However, from a user's perspective, it would be difficult to translate their information need to a collection of low level queries. On the other hand, there do exist application-specific systems which perform semantics-based information retrieval, such as [49, 45, 29]. However as mentioned in the introduction, these systems are insufficient to answer the queries, because they merely filter documents by the explicit occurrences of annotations.

The above discussion motivates a system that is able to provide users (or applications) a means to access and explore annotated documents considering:(i) the document structure to navigate the specific information units by proximity, (ii) the explicit annotation structure to specify constraints of the emerged semantics and (iii) the external knowledge to increase the quality of query results. Drawing from the above idea, QUASAR (QUerying Annotation, Structure And Reasoning), is thus proposed as such a system for structured querying of annotated documents dealing with all the above aspects.

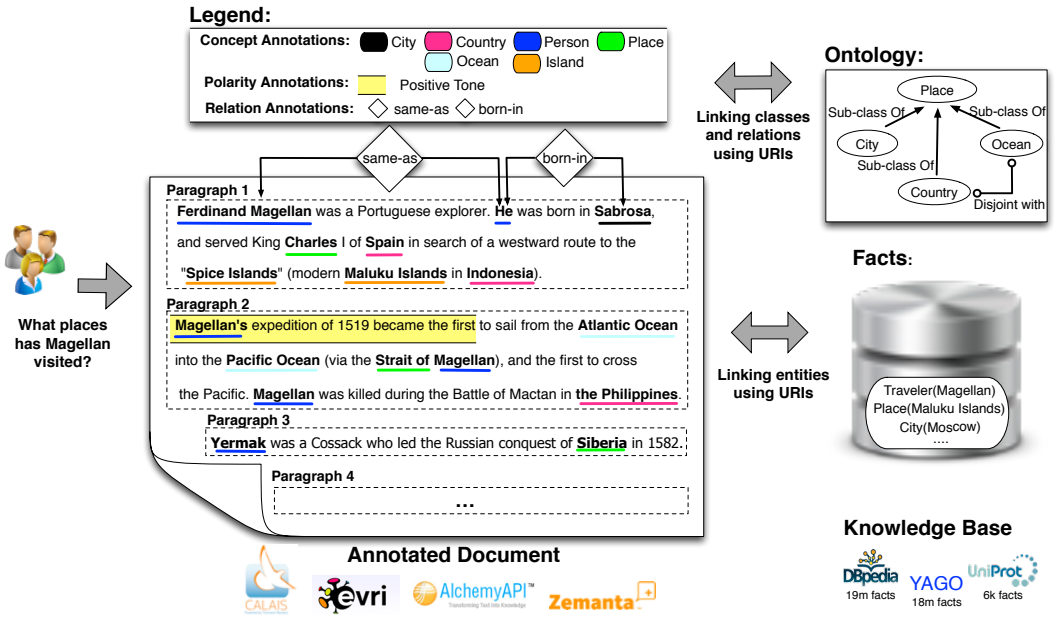


Figure 4.1: Motivating example

4.2 Data model

In this section, we propose a logical data model which represents semantic annotations, unstructured documents and external knowledge bases, as well as the relationships among them. The data model contains three components: the document structure, document annotations and knowledge bases, where each component will be explained through the example in Figure 4.2.

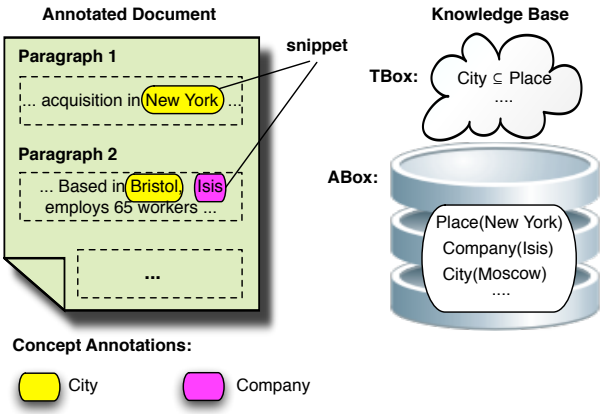


Figure 4.2: Data model example

4.2.1 The QUASAR model

Document Structures. Documents in plain text are assumed to be divided up into a hierarchy of blocks, which can be referred to and navigated within by queries. As depicted in Figure 4.3, in our data model the hierarchy is of depth three: a document is divided into paragraphs, then into a sequence of sentences, and then into characters. In fact, the data model allows extending to other hierarchical structures such as XML.

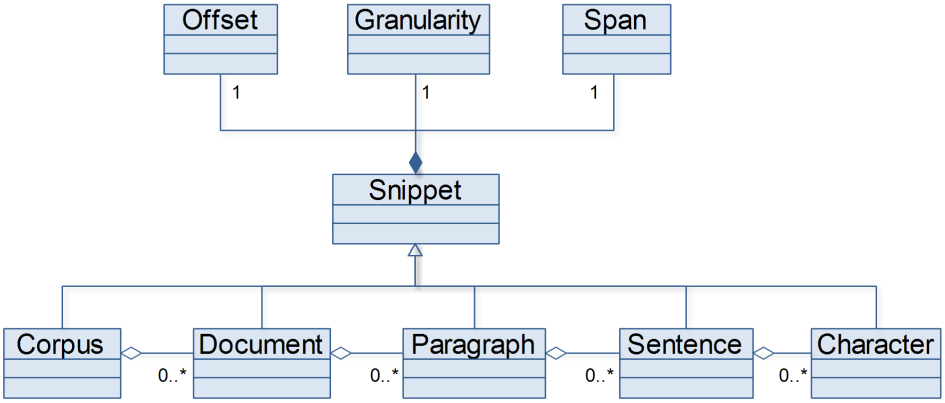


Figure 4.3: Data model of document structure

- *<Snippet>* A *snippet* (or *region*) refers to a contiguous sequence of blocks in the same level of the document structure (e.g., the first three sentences in one paragraph). In Figure 4.2 we have three paragraphs where the first one has a sentence containing a snippet of the char sequence "New York". A snippet is uniquely identified by a triple specifying *offset*, *granularity* and *span*.

- *<Granularity>* The *granularity* determines the structural level (or depth) at which a snippet is (e.g., sentence or char).

- *<Offset>* The *offset* describes the starting point of a snippet within a document.

- *<Span>* The *span* is an integer value specifying the number of blocks a snippet covers at corresponding granularity. For example, if the granularity of a snippet is sentence, then "span = 4" means that the snippet has a span crossing four successive sentences.

We assume that the granularity of each snippet is "pure", which means that a (well-formed) snippet will not span the boundary of a document component having higher-level granularity than that of itself.

Document Annotations. Figure 4.4 describes the annotation structure in the data model, from which we observe that the *annotation* serves as the bridge between the knowledge and the textual content.

- *<Annotation>* An *annotation* associates metadata with a region. As mentioned in the introduction, the attached metadata varies from one annotator to another. These semantic annotators could have diverse annotating purposes, such as entity and relation annotation, sentiment annotation and topic annotation. For example, in Figure 4.2, an entity annotator annotates "New York" with City and "Isis" with Company, along with the URI disambiguating the snippet, e.g. the URI of "Isis" in DBpedia.

- *<Assertion>* We consider the metadata uniformly as assertions. An *assertion* states a fact either from a knowledge base or produced by annotators. It is composed of a *predicate* and participating arguments.

- *<Predicate>* In general, predicates of annotation assertions are labels attached by annotators. These labels are usually from pre-defined vocabularies. Examples include topic categories (e.g., Finance), sentiments (e.g., Positive in Figure 4.1) and named entity classes (e.g., Company in Figure 4.2).

- *<Instance>* An instance is a reference to a particular entity (e.g., "Barack Obama"). An assertion may require one or more arguments. For convenience, we treat every assertion as the one which has at least one argument. Thus a known instance is represented by its URI in the arguments, while an unknown instance is treated as a special kind of argument based on the corresponding snippet annotated. For example, for sentiment analysis, we fit annotations into this model by having sentiment classifications as unary predicates on entities or text regions. Similarly, topic annotations (e.g. as provided by tools like ALCHEMY) are modelled by having unary predicates for topic categories, with the argument being the entity representing the specific text region. For entity/entity-relation annotations, we normally have the entity/relation type as a predicate and the canonical name either being URIs if disambiguated or the mention (or snippet) otherwise.

Knowledge bases (KBs). KBs can be employed as external background repositories. As shown in Figure 4.4, a KB consists of a *TBox* or *ontology* which defines rule assertions, e.g., that a city is a place (in Figure 4.2 we represent this with a subclass relationship $City \sqsubseteq Place$); and an *ABox* which is a collection of fact assertions that instantiate classes and binary relations. In Figure 4.2 we have three ABox assertions, e.g., Company(Isis)

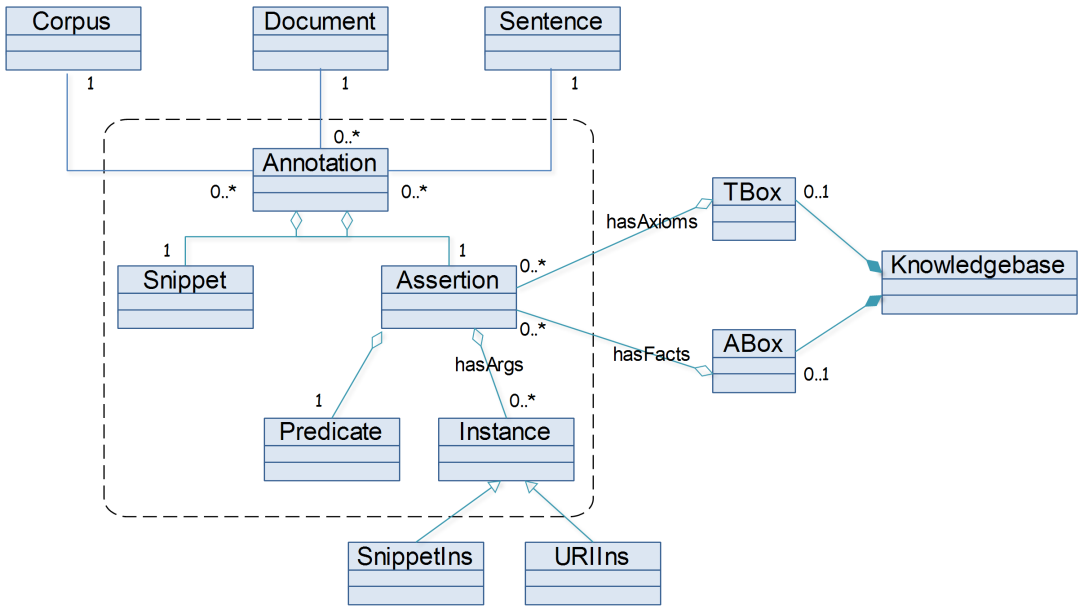


Figure 4.4: Data model of annotation structure and knowledge bases

which says that "Isis" is a member of the class Company. We assume that every assertion comes with the URIs of its components, e.g., Company(Isis) comes with the Wiki pages of Isis and of the term company. Note that the KB allows users to query over "extended" facts – facts that could either already exist in the KB, or be inferred by reasoners. In Figure 4.2 we can infer Place(Moscow).

Using URIs we can link assertions from a KB to annotated snippets of annotated documents, and thus the KB can be employed as an external background resource to enhance the quality of query answering over annotated documents. QUASAR has abstract interfaces corresponding to each component of the data model – e.g. for accessing annotators and the KB, and for loading and accessing annotated documents.

4.2.2 Preliminaries and assumptions

There are several assumptions we make before the design of the data model:

1. We assume that the element in the top level of the document structure is corpora, and each corpus in the corpora contains a collection of free-text documents. Each document has the syntactic structure as shown in Figure 4.5:

2. We assume that the corpora has already been annotated by one or more annotators. We allow arbitrary annotations in our data model.

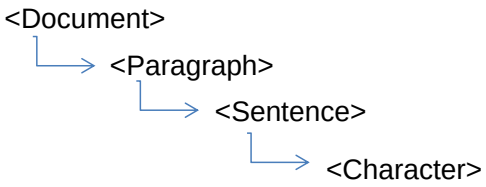


Figure 4.5: Structure of free-text document

3. In this data model, we do not consider any ensemble strategies to aggregate different opinions over annotators as we did in Chapter 3. In addition, we do not model the confidence of these annotators.

4. We assume that any term of an assertion predicate comes from a global vocabulary. For example, Annotator1: Film and Annotator2: Movie are mapped to the single one Film.

4.3 The QUASAR language

We now present the QUASAR language which allows querying over annotations, document structures, and information derived from annotations using reasoning.

4.3.1 Syntax

The general form of a query block contains three subclauses: a *SELECT* subclause, a *FROM* subclause and a *WHERE* subclause. More precisely:

SELECT	[*annotationVars*]	(*required*)
FROM	[*corpora annotationVars*]	(*required*)
WHERE	[*constraints*]	(*optional*)

The detailed grammar is described in Figure 4.6, where the symbols are defined in Table 4.1. We *SELECT* annotations *FROM* a sequence of corpora, associating each annotation with a variable (or alias). In the *WHERE* clause we can impose several kinds of constraints, defined as *atomConstraint* in Figure 4.6. For simplicity, only *conjunctive* constraints are considered in this language. An *atomConstraint* could either be an annotation content constraint which can be explicit or implicit, or an annotation proximity constraint. Each *atomConstraint* is a binary condition that associates two operands by using a relation operator *relOp*, where at least one operand is an annotation attribute (a

Table 4.1: QUASAR syntax symbol summary.

Symbol	Description
::=	The element to the left of the symbol is defined by the constructs on the right.
*	The preceding construct may occur zero or more times.
+	The preceding construct may occur one or more times.
?	The preceding construct may occur no more than once.
[...]	The constructs within the braces are grouped together.
|	An exclusive OR.
BOLDFACE	A keyword; although capitalized in the grammar diagram, keywords are not case-sensitive.
Italic	Non-terminals.
"..."	Terminals.

qualified attribute name) such as the proximity attribute (see *annoProximityAttr*) or the annotation assertion attribute (see *annoAssertionAttr*).

Explicit property constraints *explicitPropertyCnst* include annotation proximity constraints *annoProximityCnst* and annotation content constraints *annoContentCnst*. In proximity constraints, the proximity and the order of annotations are considered. The order is implicitly determined by the location at which annotations occur in the text. Expressions are allowed in *annoProximityCnst* which are either integer constants, attribute name operands or infix expressions with arithmetic operators (see the recursive rules of *annoProximityExp*). The annotation content constraints state *exactly* what predicates or entities or assertions should occur in an annotation (defined as *annoAsstPredicateCnst*, *annoAsstArgCnst* and *annoAssertionCnst* respectively).

An implicit content constraint *implicitPropertyCnst* states that a predicate or entity occurring in an annotation should satisfy certain properties with respect to an ontology. An *implicitPropertyCnst* is denoted as an atomic assertion comparison followed by *OntologyFilter* which introduces a constraint based on facts inferred from the ontology. Within *OntologyFilter*, a conjunctive assertion formula $assertionFormula_{\phi_\wedge}$ is specified of which each conjunct is either an atomic TBox assertion *atomTBoxAssertion* or an atomic ABox assertion *atomABoxAssertion*. The main difference between *atomTBoxAssertion* and *atomABoxAssertion* is that the latter allows the predicate to be a variable while the former does not.

4.3.2 Semantics

The semantics of the QUASAR language is given through a collection of valuation functions. Each of the paragraphs below describes, in an increasing level of detail, the seman-

QuasarQuery ::= *QuasarQuery* "UNION" *QuasarQuery* | *SelectQuery*

SelectQuery ::= "SELECT" [*annoVar*$^+$ | "$*$"]
"FROM" [*CorpusName* ".Annotation" *annoVar*]$^+$
["WHERE" *constraints*] ?

constraints ::= *constraints* "AND" *constraints* | *atomConstraint*

atomConstraint ::= *explicitPropertyCsnt* | *implicitPropertyCsnt*

Explicit property constraints

explicitPropertyCnst ::= *annoContentCnst* | *annoProximityCnst*

annoProximityCnst ::= *annoProximityAttr* *relOp* *annoProximityExp*

annoProximityExp ::= *annoProximityExp* *arithOp* *annoProximityExp*
| *annoProximityAttr* | *integerValue*

annoProximityAttr ::= *annoVar* ".snippet. " [" docNum " | " paraNum " | "sentNum"
| " charNum " | $\cdots$]

arithOp ::= "$+$" | "$-$" | "$\times$" | "$\div$"

relOp ::= *equalOp* | "$\leq$" | "$\geq$" | "$<$" | "$>$" | "$\approx$"

equalOp ::= "$=$"

annoContentCnst ::= *annoAssertionCnst* | *annoAsstPredicateCnst* | *annoAsstArgCnst*

annoAssertionCnst ::= *annoAssertionAttr* *equalOp* [*annoAssertionAttr* | *atomABoxAssertion*]

annoAssertionAttr ::= *annoVar* ".assertion "

annoAsstPredicateCnst ::= *annoAsstPredAttr* *equalOp* [*annoAsstPredAttr* | *stringValue*]

annoAsstPredAttr ::= *annoAssertionAttr* ".predicate"

annoAsstArgCnst ::= *annoAsstArgAttr* *equalOp* [*annoAsstArgAttr*| *stringValue*]

annoAsstArgAttr ::= *annoAssertionAttr* ".arg ["*argIdx*"] "

Implicit property constraints

implicitPropertyCnst ::= *annoAssertionAttr* *equalOp*
atomABoxAssertion " [OntologyFilter: " $assertionFormula_{\phi_\wedge}$ "] "

$assertionFormula_{\phi_\wedge}$ ::= $assertionFormula_{\phi_\wedge}$ "AND" $assertionFormula_{\phi_\wedge}$
| *atomABoxAssertion* | *atomTBoxAssertion*

atomTBoxAssertion ::= *stringValue* " (" [*stringValue* | *ABoxPredVar*] $^+$ ") "

atomABoxAssertion ::= [*stringValue* | *ABoxPredVar*]
" (" [*stringValue* | *ABoxInstanceVar*] $^+$ ") "

Figure 4.6: The QUASAR grammar

tics of the action given in the header. Table 4.2 and Table 4.3 describe the terms and the utility methods respectively, which are commonly used across the evaluation functions. For brevity, we omit type compatibility requirements (e.g., in a comparison action) and scope checking (e.g., the existence of selected corpus) from the valuation functions.

Table 4.2: Definitions of terms.

Terms	**Description**
AnnoStore	An annotation store $\Lambda = \{a_1, \cdots, a_n\}$, where the type of each a_i is *Annotation* (defined in Section 4.2)
Ontology	An ontology Ω
Value	$Value = IntegerValue \cup AtomABoxAssertionValue \cup StringValue$
DataType	$DataType = \{ Integer \cup String \cup AtomABoxAssertion \}$
AnnoAttribute	An annoAttribute is characterized for its type and name. $AnnoAttribute = \{ AnnoProximityAttr \cup AnnoAssertionAttr \cup AnnoAsstPredAttr \cup AnnoAsstArgAttr \}$
AnnotationsTuple	A k-tuple $\langle a_1, \cdots, a_k \rangle$, where $a_i \in \Lambda \wedge a_i$: *Annotation*.
ResultSet	An object returned by a *QuasarQuery*, which is composed of a collection of *AnnotationsTuples* and the corresponding *Metadata* object.
Metadata	The metadata (or schema) of a set of *AnnotationsTuple* records the names of corresponding bound annotation variables.

Table 4.3: Definitions of utility methods.

Methods	**Description**
$mk_X()$	Construct the component object X, which involves in parsing procedures.
$newMeta()$	Create a *Metadata* object according to the input *FromField*.
$createRes()$	Create a *ResultSet* object containing the dataset and metadata.
$valOf()$	Get one of the components in tuple t, of which the corresponding variable name is specified as a parameter.
$dataSet()$	Get the tuple set from a *ResultSet* object.
$metaData()$	Get the metadata from a *ResultSet* object.

eval is the entry function to evaluate a *QuasarQuery*. For a union query, it combines the results of two sub-queries.

```
eval(q: QuasarQuery, Λ: AnnoStore, Ω: Ontology) return R:
   ResultSet := {
2  cases q of:
      mk_Union(q1, ''UNION'', q2) → R=eval(q1, Λ, Ω) ∪ eval(q2, Λ, Ω
         )
```

```
4       mk_Sel(q) → R=evalSel(q, Λ, Ω)
}
```

EvalSel conditionally selects a subset of k-tuple *AnnotationsTuple* and projects them to l-tuple *AnnotationsTuple* over the specific set of annotation variable ($l \leq k$).

```
evalSel(q: SelectQuery, Λ: AnnoStore, Ω: Ontology) return R:
   ResultSet := {
2  cases q of:
      mk_SelAll(sf, fs) → R=valueSel(sf, fs, Λ)
4     mk_SelCond(sf, fs, cond) → R=valueSelCond (sf, fs, cond, Λ, Ω
         )
}
```

ValueSel selects all the projected tuples without any constraints. It simply binds each annotation variables with all the annotations within the corresponding corpus and takes the Cartesian product if there are multiple *From* fields. The function requires that the projected annotation aliases are the subset of the aliases specified in the *FromField*. $\Lambda(fs_i)$ returns the corresponding annotation set from the annotation store which is bound to the corpus name in *FromField* fs_i.

```
valueSel(sf: SelectFields, fs: FromFields, Λ: AnnoStore) return R
   : ResultSet := {
2  rs=Λ(fs_1)×···×Λ(fs_n)
   R=createRes(rs, newMeta(fs))
4  cases sf of:
      mk_AllFields() → R=R
6     mk_AnnoVars(ids) → R=project(R, ids)
}
```

Project creates a new *ResultSet*, where the fields after the projection correspond to the annotation variables in ids. The symbol $ids \lhd md$ means a projection from source schema md to a target schema which is only associated with the set of *AnnoVar* ids.

```
project(R: ResultSet, ids: AnnoVars) return R′: ResultSet := {
2  rs = ∅
   md= metaData(R)
4  for ∀t ∈ dataSet(R) →
      rs = rs ∪ {⟨valOf(ids_1, t, md), ··· , valOf(ids_|ids|, t, md)⟩}
6  R′=createRes(rs, ids ◁ md)
}
```

ValueSelCond performs a selection over conditions $cond$, which belongs to the *WHERE* clause in *SelectQuery*.

```
valueSelCond(sf: SelectFields, fs: FromFields, cond: Constraints, Λ
   : AnnoStore, Ω: Ontology) return R: ResultSet := {
2  rs=Λ(fs_1)×···×Λ(fs_n)
   R′=createRes(rs, newMeta(fs))
```

```
    cases sf of:
       mk_AllFields() → R= valueSelCond_All(R′, cond, Ω)
       mk_Identifiers(ids) → R= project(valueSelCond_All(R′, cond,
          Ω), ids)
}
```

ValueSelCond_All selects tuples without a projection. An *AnnotationTuple* t is added to the result set if it satisfies the conditions *cond*.

```
valueSelCond_All(R′: ResultSet, cond: Constraints, Ω: Ontology)
   return R: ResultSet := {
   rs = ∅
   for ∀t ∈dataSet(R′) →
       if valueConstraints(t, metaData(R′), cond, Ω) → rs = rs ∪ {t}
   R= createRes(rs, metaData(R′))
}
```

ValueConstraints takes an input *AnnotationTuple* t and checks whether it satisfies the constraints *cond*. It evaluates *conjunctive* boolean constraints recursively. For simplicity, *only* the boolean operator *"AND"* is considered.

```
valueConstraints(t: AnnotationsTuple, md: Metadata, cond:
   Constraints, Ω: Ontology) return R: Boolean := {
   cases cond of:
      mk_Constraints(cond1, ''AND'', cond2)→ R= valueConstraints(t,
         md, cond1, Ω) ∧ valueConstraints(t, md, cond2, Ω)
      mk_AtomConstraint(c) → R= valueAtomConstraint(t, md, c, Ω)
}
```

ValueAtomConstraint evaluates an atomic comparison for a given tuple, which can be an *ExplicitPropertyCnst* or an *ImplicitPropertyCnst* requiring an ontology.

```
valueAtomConstraint(t: AnnotationsTuple, md: Metadata, c:
   AtomConstraint, Ω: Ontology) return R: Boolean := {
   cases c of:
      mk_ExplicitPropertyCnst(ec) → R= evalExplicitCnst(t, md, ec)
      mk_ImplicitPropertyCnst(ic) → R= evalImplicitCnst(t, md, ic,
         Ω)
}
```

EvalExplicitCnst evaluates explicit property constraints where each constraint is finally interpreted as an arithmetic comparison, a string comparison or an assertion constant comparison. Note that all the atomic proximity constraints are essentially arithmetic comparisons, while each predicate constraint and argument constraint is treated as a string comparison.

```
evalExplicitCnst(t: AnnotationsTuple, md: Metadata, c:
   ExplicitPropertyCnst) return R: Boolean := {
   cases c of:
```

```
        mk_AnnoProximityCnst(intAttrName, relOp, exprs) →
4               valueL = evalOperand(intAttrName, t, md)
                valueR = evalExprs(exprs, t, md)
6               R= compareInteger(valueL, valueR, relOp)
        mk_AnnoAsstPredicateCnst(strAttrName, relOp, str) →
8               valueL = evalOperand(strAttrName, t, md)
                valueR = evalOpString(str, t, md)
10              R= compareString(valueL, valueR, relOp)
        mk_AnnoAsstArgCnst(argAttrName, relOp, str) →
12              valueL = evalOperand(argAttrName, t, md)
                valueR = evalOpString(str, t, md)
14              R= compareString(valueL, valueR, relOp)
        mk_AnnoAssertionCnst(asstAttrName, relOp, asst) →
16              valueL = evalOperand(asstAttrName, t, md)
                valueR = evalAssertion(asst, t, md)
18              R= compareAssertion(valueL, valueR, relOp)
}
```

CompareInteger evaluates a comparison between two *Integer* objects.

```
compareInteger(valueL: Integer, valueR: Integer, op: RelOp) return R
   : Boolean := {
2    cases op of
        mk_Greater() → R= (valueL > valueR)
4       mk_GE() → R= (valueL ⩾ valueR)
        mk_Less() → R= (valueL < valueR)
6       mk_LE() → R= (valueL ⩽ valueR)
        mk_Equal() → R= (valueL == valueR)
8 }
```

CompareString evaluates a comparison between two *String* objects. For the operator $\approx$, we simply interpret it as that the *edit distance* is below a given threshold σ. The method *editDist*() computes the edit distance between two strings.

```
compareString(valueL: String, valueR: String, op: RelOp) return R:
   Boolean := {
2     cases op of
         mk_EqualOp() → R= (valueL == valueR)
4        mk_Like() → R= (editDist(valueL, valueR) < σ)
}
```

CompareAssertion evaluates a comparison between two *AtomABoxAssertion* objects. It returns *true* iff both assertion constants have identical predicates and arguments. The utility functions *getPredicate*() and *getArgs*() return the values of the predicate and participating arguments respectively in a given assertion. Note that currently only the equality operator is supported although the language could further provide more complicated semantics of assertion comparisons.

```
compareAssertion(valueL: AtomABoxAssertion, valueR:
   AtomABoxAssertion, op: RelOp) return R: Boolean := {
    cases relOp of:
       mk_EqualOp() → R= comparePredicate(getPredicate(valueL),
          getPredicate(valueR), ''='')∧
          comapreArgumentLst(getArgs(valueL), getArgs(valueR),
          ''='')
}
```

ComparePredicate evaluates a comparison between two predicate objects, where a predicate is either a variable or a string constant. In the case that a predicate is a variable, it is treated as a wildcard. The evaluation always returns *true* if at least one operand is a variable.

```
comparePredicate(valueL: Predicate, valueR: Predicate, op: RelOp)
   return R: Boolean := {
   cases valueL of:
      mk_PredVar() → R = true
      mk_String(strL) →
         cases valueR of:
            mk_PredVar() → R = true
            mk_String(strR) → R= compareString(strL, strR, ''='')
}
```

CompareArgumentLst evaluates a comparison between two lists of arguments.

```
compareArgumentLst(argsL: Arguments, argsR: Arguments, op: EqualOp)
   return R: Boolean := {
    if |argsL| ≠ |argsR| → R = true
    for(i=1, ···, i= |argsL|) →
        if ¬ compareArgument(argsL_i, argsR_i, op) → R= false
    R = true
}
```

CompareArgument evaluates a comparison between two arguments, where each argument is either a string value or a variable. The comparison returns *true* if at least one argument is a variable.

```
compareArgument(argsL: Argument, argsR: Argument, op: EqualOp)
   return R: Boolean := {
   cases argsL of:
      mk_ArgVar() → R = true
      mk_String(strL) →
         cases valueR of
            mk_PredVar() → R = true
            mk_String(strR) → R= compareString(strL, strR, ''='')
}
```

EvalExprs gives the evaluation of integer arithmetic expressions recursively. If an atomic term in the expression is an operand (qualified annotation attribute name), then evaluate

the value of this attribute. If an atomic term is an *IntegerValue* then return the value itself.

```
evalExprs(exprs: AnnoProximityExp, t: AnnotationsTuple, md:
    Metadata) return R: Integer := {
2   cases exprs of:
      mk_AnnoProximityExp(expr1, arithOp, expr2) →
4           valueL = evalExprs(expr1, t, md)
            valueR = evalExprs(expr2, t, md)
6           cases arithOp of:
              mk_Plus() → R= (valueL + valueR)
8             mk_Minus() → R= (valueL − valueR)
              mk_Times() → R= (valueL ∗ valueR)
10            mk_Divide() → R= (valueL/valueR)
      mk_AnnoProximityAttr(op) → R= evalOperand(op, t, md)
12    mk_IntegerValue(i) → R= i
}
```

EvalOperand returns the corresponding value of the specified attribute of an annotation object. For example, let a be an annotation with the assertion Company(Isis). Given opr = "a.assertion.predicate", this function returns "Company". The utility function *getAttrValue*() returns the value of the required annotation attribute from a given tuple.

```
evalOperand(opr: AnnoAttribute, t: AnnotationsTuple, md: Metadata)
    return R: Value := {
2   cases opr of:
      mk_AnnoAssertionAttr(annoVar, annoAssertionAttr) → R=
        getAttrValue(t, md, annoVar, ``assertion'')
4     mk_ProximityAttr(annoVar, annoProximityAttr) →
          cases annoProximityAttr of:
6           mk_docNum()  → R= getAttrValue(t, md, annoVar,
              ``docNum'')
            mk_paraNum()  → R= getAttrValue(t, md, annoVar,
              ``paraNum'')
8           mk_sentNum()  → R= getAttrValue(t, md, annoVar,
              ``sentNum'')
            mk_charNum()  → R= getAttrValue(t, md, annoVar,
              ``charNum'')
10    mk_AnnoAsstPredicateAttr(annoVar, annoAsstPredAttr) → R=
        getAttrValue(t, md, annoVar, ``predicate'')
      mk_AnnoAsstArgAttr(argIdx, annoVar) → R= getAttrValue(t, md,
        annoVar, argIdx)
12  }
```

EvalAssertion returns the value of an assertion by evaluating an assertion constant directly or an attribute operand which is a *AtomABoxAssertion* object.

```
evalAssertion(asst: OpAssertion, t: AnnotationsTuple, md:
    Metadata) return R: AtomABoxAssertion := {
```

```
    cases str of:
        mk_AtomABoxAssertion(s)→ R = s
        mk_Operand(op) → R= evalOperand(op, t, md)
}
```

EvalOpString returns the string value of an operand, where the operand must be of type *String*.

```
evalOpString(str: OpString, t: AnnotationsTuple, md: Metadata)
    return R: String := {
    cases str of:
        mk_String(s)→ R = s
        mk_Operand(op) → R= evalOperand(op, t, md)
}
```

EvalImplicitCnst evaluates an atomic constraint which requires checking inferred facts w.r.t. an ontology. $Filter$ is a conjunctive query over an ontology, where $x_1, \cdots, x_m$ are the variables in assertion operand $asstR$, while $filter'$ is a boolean query after binding $x_1, \cdots, x_m$ with corresponding values of the predicate and arguments in assertion constant $valueL$ (represented by $valueL \mapsto x_i$). The method *evalOntology*() then determines whether the TBox and ABox in Ω logically imply the query $filter'$.

```
evalImplicitCnst(t: AnnotationsTuple, md: Metadata, c:
    ImplicitPropertyCnst, Ω: Ontology) return R: Boolean := {
    cases c of:
        mk_ImplicitPropertyCnst(asstL, ''='', asstR, filter) →
            valueL = EvalOperand(asstL, t, md)
            if ¬ CompareAssertion(valueL, asstR, ''='') → R= false
            Given filter = ∃y_1, ··· , y_n  Φ_∧(x_1, ··· , x_m)
            filter' = ∃y_1, ··· , y_n  Φ_∧(valueL ↦ x_1, ··· , valueL ↦ x_m)
            R = evalOntology(filter', Ω)
}
```

In practice, *evalOntology*() can be implemented by translating $filter'$ into a corresponding SPARQL or SQL and returns *true* if the query result is not empty. We will see more details later in Section 4.4.2.

4.3.3 Example queries

We now illustrate the QUASAR query language on representative examples below. The following query Q^1_{place} asks for annotated snippets in the corpus "Corp1" that are annotated as a place:

```
SELECT * FROM Corp1.Annotation ?a
WHERE ?a.assertion.predicate = ''Place''
```

Here ?a is an annotation variable. Such a query would be appropriate in a situation where the user does not have access to background information. Note that implicit property constraints can be applied to filter annotations according to "external knowledge". A variant of the query above, Q^2_{place} allows a user to ask for an annotation in Corp1 that is annotated as a subtype of Place:

```
SELECT * FROM Corp1.Annotation ?a
WHERE ?a.assertion = ?Z(?x) [OntologyFilter:SubType(?Z,''Place'')]
```

Here ?Z and ?x are predicate and argument variables, implicitly existentially quantified; OntologyFilter introduces a constraint based on the concept inference from the ontology. The particular ontology being used does not need to be explicitly referenced in the query above, but is in a separate configuration file. Since determining subtype relationships may require reasoning (they may not be explicit in the ontology), our system has a full reasoner embedded in it.

As mentioned before, annotation proximity constraints give restrictions on the position of snippets within the document structure. The following simple query Q_{begin} fetches all the annotated snippets in the body of the document occurring in the first two paragraphs:

```
SELECT * FROM Corp1.Annotation ?a
WHERE ?a.snippet.paraNum ≤ 2
```

The QUASAR system allows to combine annotations from different annotators and vocabularies. Consider the query fetching all snippets labelled with Place and mentioned in a negative tone in the document. The relation operator ≈ means an overlap between the spans of two snippets.

```
SELECT ?b FROM Corp1.Annotation ?a, Corp1.Annotation~?b
WHERE ?a.assertion.predicate = ''Negative''
      AND ?b.assertion.predicate = ''Place''
      AND ?a.snippet ≈ ?b.snippet
```

The flexibility of the language also allows the user to narrow down the answers to those that satisfy very specific criteria, thus addressing the problem of "too many answers" that occurs often in keyword querying. As an example, consider a user who is looking for cities which are the birthplaces of some politicians. The information need can be expressed very specifically with the following query Q_{city}:

```
SELECT * FROM Corp1.Annotation ?a
WHERE ?a.assertion = City(?x) [OntologyFilter: Birthplace(?y, ?x)
      AND Politician(?y)]
```

The language also gives the user the ability to control the extent to which implicit information is utilized in a very fine-grained manner. Consider the situation in which a user wants to find annotations mentioning a place. We have seen one embodiment of this as the query Q^1_{place}. It requires the annotation Place to be explicit in the annotation structure but does not require the entity to be recognized as a specific place known to the KB. We have also seen the alternative formulation Q^2_{place}, in which the annotation can be a subtype of Place, but again not requiring the entity to be recognized as a specific place. In contrast, the user could issue the query Q^3_{place}, asking for annotations that recognize an entity that the ontology knows to be a place:

```
SELECT * FROM Corp1.Annotation ?a
WHERE ?a.assertion = ?Z(?x) [OntologyFilter: Place(?x)]
```

Finally, the user who wants the broadest semantics possible could ask the query Q^4_{place}, which finds anything that can be inferred to be a place:

Q^2_{place} UNION Q^3_{place}.

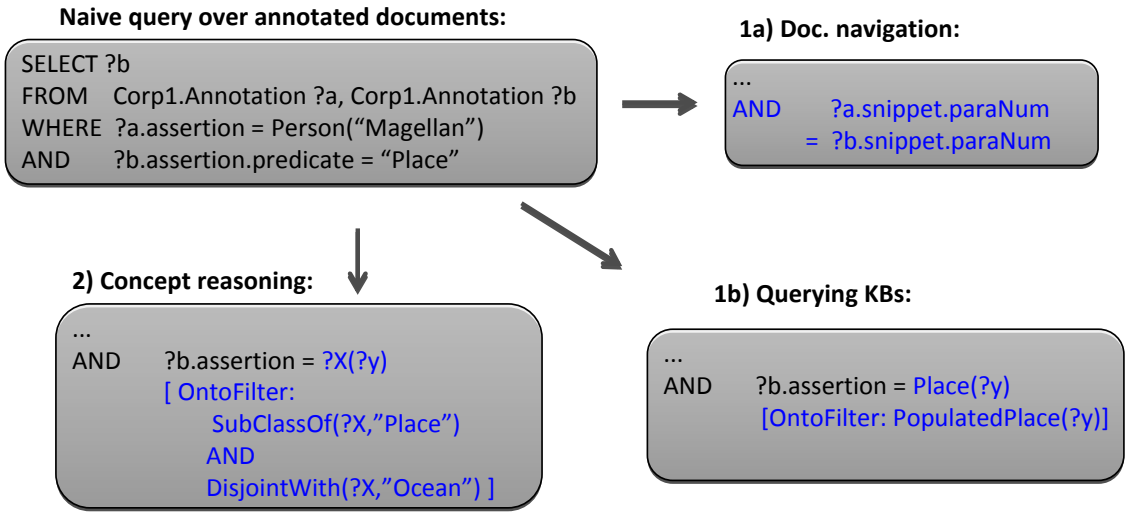


Figure 4.7: Motivating query in QUASAR language

Let us return to the motivating scenario and review the example query in Figure 4.1. For the information need to be focused on, Figure 4.7 and Figure 4.8 show how we benefit from the properties of QUASAR to improve the query results of the motivating query. If we just focus on the explicit syntax of annotations, we could only construct the following query:

```
SELECT ?b FROM Corp1.Annotation ?a, Corp1.Annotation ?b
WHERE ?a.assertion = Person("Magellan")
AND ?b.assertion.predicate = "Place"
```

The above query asks for places which are explicitly stated by annotators and co-occur with the person "Magellan". Obviously, the quality of query results obtained in Figure 4.8 is unsatisfactory, although one of the correct answers is covered. The naive answer set contains not only unwanted answers such as "Siberia" visited by "Yermak" but also some wrong answers resulting from the mistakes of annotators (e.g., "Charles"). In addition, it misses many expected answers such as "Indonesia".

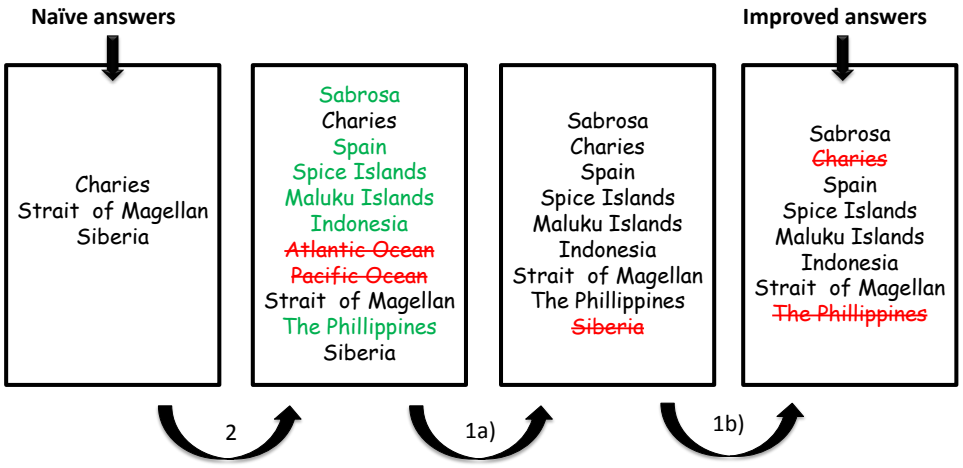


Figure 4.8: Improving answers of the motivating query

However, by applying the extra constraints which take advantage of the features of QUASAR, we are clearly able to tackle the above issues raised by the naive query with blind access of annotated documents. On the one hand, we can increase right answers which are missed initially as relevant types of places (e.g., islands and countries) by specifying the predicate to be subclasses of Place (see 2)). With the "relaxed" condition, we retrieve many "implicit" results such as "Maluku Islands". On the other hand, we can reduce wrong answers by further introducing the constraints by navigating document and querying implicit properties of annotations. For example, we first restrict both annotations ?a and ?b to co-occur near enough with each other, e.g., in the same paragraph (see 1a)). This allows us to eliminate the places which are irrelevant due to wrong parts of documents being queried (e.g., "Siberia"). We can then remove irrelevant places which are less interesting such as Ocean by imposing the constraint that all the returning types are disjoint with Ocean (see 2)). Finally we eliminate places which are not "true" geographic locations such as "Charles" by validating the annotation assertion in the KB, for example, querying the KB whether the annotation instance is a populated place (see 1b)).

In summary, the QUASAR language provides the user the ability to combine document structure, explicit annotation structure, and implicit knowledge, while dealing with multiple vocabularies and annotators. It allows users to pull in implicit results, but allows fine-grained control over whether and how implicit information is used.

4.4 The QUASAR system

The QUASAR system is a prototype query engine, which allows users to compose and execute QUASAR queries over annotated corpora. The system also provides users with visualization and navigation of the querying results. This section first presents the system architecture. Then it introduces the implementation in detail. Finally it describes the graphical user interfaces, zooming into the functionality of each component in the system.

4.4.1 System architecture

Figure 4.9 depicts the architecture of the QUASAR system. The framework consists of three major modules: the annotation harvesting module, the annotation processing module and the query processing module, where the first two are offline while the third one is online.

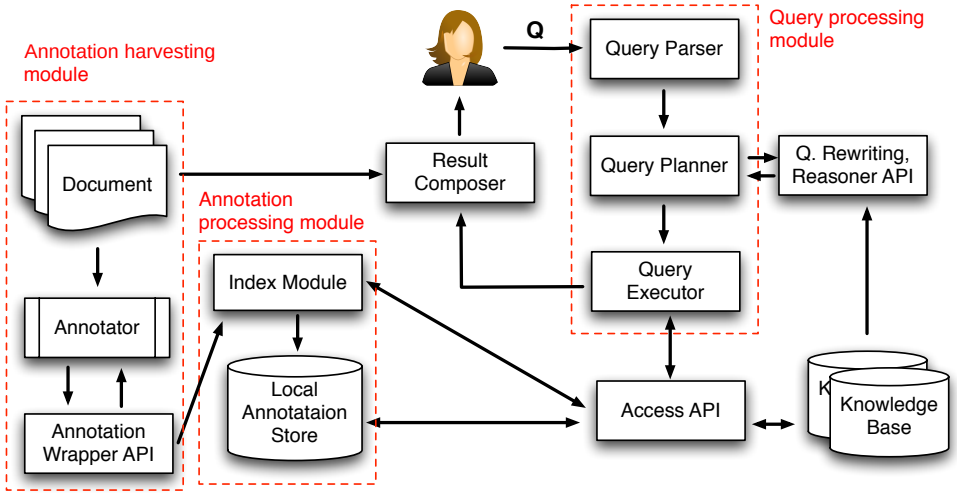


Figure 4.9: System architecture of QUASAR

Offline Modules. The *Annotation Harvesting Module* mainly contains two kinds of components: Annotators and AnnotatorWrappers. In order to integrate annotators with distinct APIs, metadata and response formats, an uniform wrapper interface is defined for the interaction between annotators and the system (see Figure 4.10). Based on the interface AnnotatorWrapper, concrete wrappers must be implemented for each annotator employed (e.g., AlchemyWrapper and OpenCalaisWrapper). Documents are annotated by different annotators, each of which could either be a web service or a centralized software component. Each AnnotatorWrapper submits documents to the corresponding annotator and then harvests the structural information about annotations from the response into the global AnnotationStore. Note that the documents themselves remain external to the

store; the store uses only into those documents for efficiency reasons. As mentioned later in Section 4.4.2, the pointer of a document uses its offset w.r.t. the document granularity.

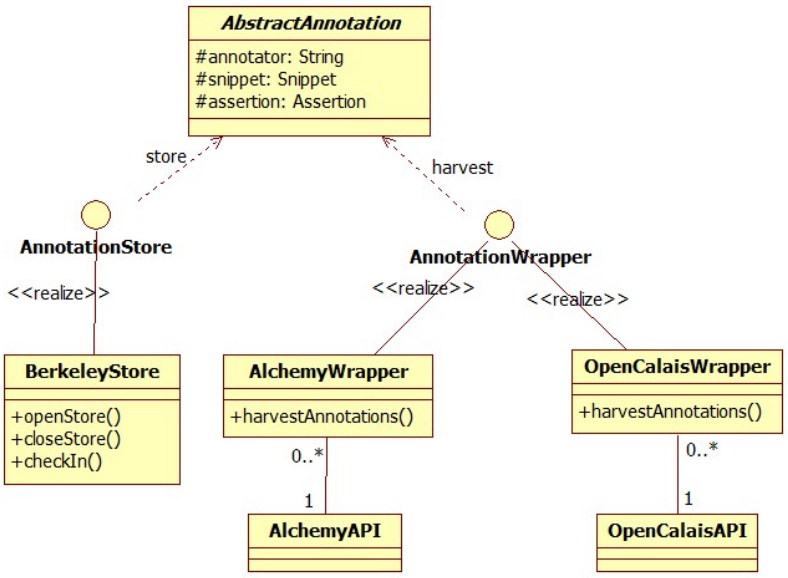


Figure 4.10: Class diagram fragment of offline modules

The *Annotation Processing Module* is responsible for maintaining the indices on the explicit information of annotations. The indexing tasks are performed internally by the AnnotationStore for efficiently accessing all the annotation objects.

Background knowledge is maintained with indexing in the Knowledge base, which can be either external or stored locally in the knowledge repository of the system.

Online Modules. The *Query Processing Module* covers the top-level components for parsing, planning and executing the QUASAR queries, as well as reasoners for inferring implicit information of annotations. Figure 4.11 shows the class diagram about the *Query Processing Module*. The entry point for processing a QUASAR query is the QueryParser. The parsed query statement is then sent to the QueryPlanner. The QueryPlanner decides on a strategy for execution, including: which Reasoner and Knowledge base to use (if multiple reasoners/KBs are available), which indices to use for accessing the annotation store. Note that the QueryPlanner is designed as an interface, of which the method *makePlan*() needs to be implemented by a concrete planner. The *makePlan*() returns a QueryPlan object which represents a physical plan. If a Rewritten-based Reasoner is used, the QueryPlanner interacts with the reasoner to unfold the ontology-related constraints in favor of unions of conjunctive queries (CQs) over the KB. A Rewritten-based Reasoner interface defines some basic functions such as *bindTBox*() to bind the corresponding ontology and *rewrite*() to rewrite a CQ clause. A KB may be accessed remotely (e.g. the

DBpedia public SPARQL endpoint), the join between structural constraints and semantic constraints have to be performed within QueryPlanner itself. The QueryExecutor fulfils a query plan by interacting with the AnnotationStore and the Knowledge base.

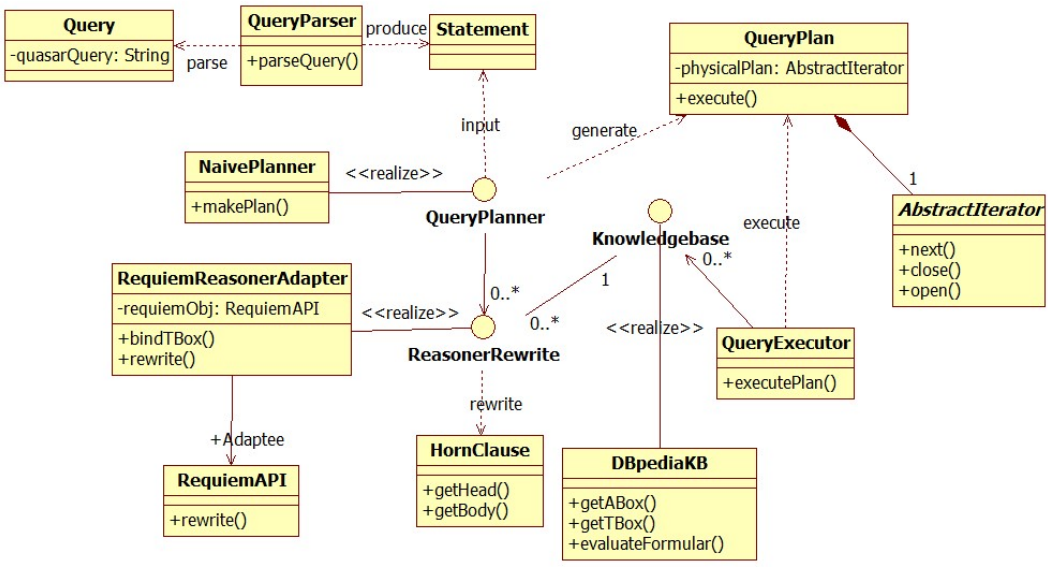


Figure 4.11: Class diagram fragment of online modules

The results returned from QueryExecutor generally contain annotation snippets represented by pointers encoding the locations within the source documents. The ResultComposer translates these pointers into concrete snippets within the document that are suitable for user navigation.

The modularity and maintainability of the system benefit from the interface-oriented framework shown in Figure 4.10 and Figure 4.11. More importantly, the framework increases the extensibility of the application. For example, it is flexible enough that any other annotators and rewriting-based reasoners can be employed as long as the common interfaces are implemented.

4.4.2 Implementation

The evaluation functions described in Section 4.3.2 are a naive way to implement the QUASAR language. They do not consider the storage and the existing database structure, and no optimized evaluation strategies are applied. We now turn to introducing in detail the system implementation which takes into account the above aspects. The QUASAR system is implemented in JAVA, with abstract interfaces for the components shown in Figure 4.9. The choices of these components used in our prototype are:

Snippet offset. According to the implicit hierarchy of a free-text document, we use a vector $\vec{V}$ to represent the offset, which is defined as $\vec{V} = \langle\ \#Corpus,\ \#Document,$

#Paragraph, *#Sentence*, *#Character* ⟩. Here "#" means the index number of a document component in an ordered collection. Figure 4.12 shows an example given the data model. The highlighted snippet "Isis" has the granularity character. The offset of the snippet is $\langle 2, 1, 2, 2, 1 \rangle$, indicating that the fragment starts from the first char in the second sentence of the second paragraph in the first document of the second corpus. The length of the snippet is four in characters.

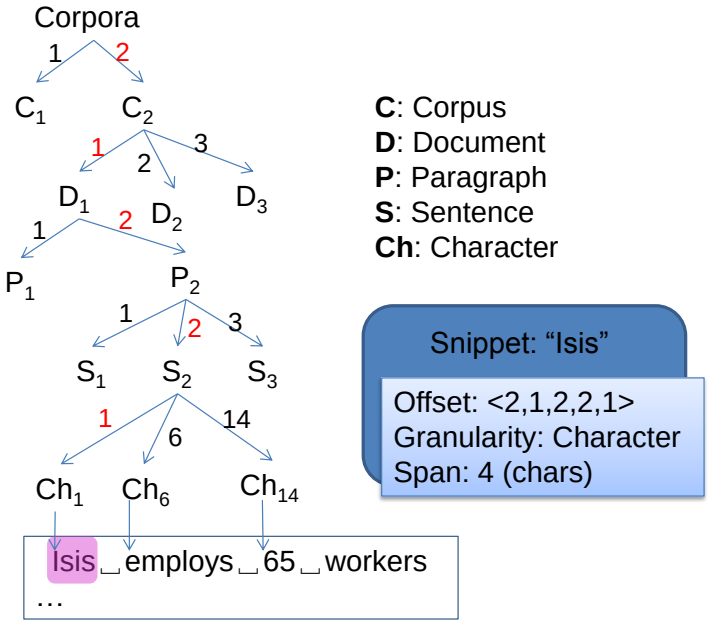


Figure 4.12: Example of encoding scheme of a snippet

Annotator. In the prototype, two text analysis APIs – OpenCalais and Alchemy are employed as the semantic annotators. OpenCalais performs entity and relation fact extraction, while Alchemy provides the support for entity and sentiment extraction.

Annotator wrapper. Tailored wrappers for OpenCalais and Alchemy are implemented to harvest the semantic annotations in diverse purposes including entity annotations, relation annotations and sentiment annotations. Jena RDF API [42] is employed to parse and explore annotation information from RDF-based responses of annotators.

Annotation store. Since annotation storage will always be a core component of the Quasar middleware, an annotation store has been built on top of BerkeleyDB JAVA Edition [65]. It is an embedded non-SQL persistence layer, providing flexible low level access primitives to annotation objects. Currently annotations are indexed by annotation predicates and positions measured by several granularities – e.g. by document, by paragraph, and by sentence.

Reasoner. Our API allows access to reasoning resources for determining whether a conjunctive query is derivable from a set of facts using axioms of a particular ontology.

Since the ontologies we deal with have fairly simple axioms (e.g.,DBpedia), currently the system is able to use a reasoner based on query-rewriting – REQUIEM [70]. For the ontologies we use, REQUIEM produces a union of conjunctive queries that can be applied to the KB.

Planner. In this prototype, we process queries with a naive planner which creates a query plan (a left-deep tree). When dealing with a join between the results from the annotation store and the KB, the planner simply assumes that the selectivity of the former is always higher than the latter. Thus we will first bind the annotation variables and then evaluate any of their ontology constraints by querying the KB. In addition, we maintain an in-memory cache to keep the results for the same bindings so that we are able to avoid duplicate queries over the KB. When dealing with a join which does not involve the KB, the planner decides the join order "greedily" according to the number of explicit constraints on annotation variables. More specifically, the outer table is the one with the largest number of selection conditions.

Knowledge bases. The KB could either be maintained internally or accessed externally (e.g. endpoints of SPARQL). The prototype imports the well-known cross-domain ontology – DBpedia as the background KB. The prototype uses a standard relational database, MYSQL v5.1.51, to encode fact triples as relations. For rewriting-based reasoners, the wrapped format of the rewritten query is thus translated into SQL for evaluation in MYSQL. Standard indexing approaches are applied to improve the performance.

Example corpus. The default corpus utilized for demonstrating sample queries comes from a collection of newswire documents. The corpus contains 186 articles (524KB in size) of CNN, BBC and ChinaDaily between 03/2010 and 09/2011 which cover diverse topics.

Empirical evaluation. The QUASAR query engine and the related persistence layer are set up on Windows XP SP3, Intel Core 2 Quad CPU,2.50 GHz and 3GB of RAM. The system is tested on DBpedia, importing the core ontology ABox of DBpedia 3.5.1, including 5,491,908 and 11,135,755 triples for concept assertions and role assertions respectively. Over 15,000 annotations are extracted from the sample corpus (around 18MB in size). Since the prototype is built upon third-party data storage engines, the query performance depends on these components as well as on our own optimization. In Figure 4.13, we report the preliminary numbers of scalability testing on the sample queries in Section 4.3. Since there is no up-to-date benchmark dataset available for the scalability testing, we synthetically generated datasets at the scale-factors of 5, 10, 15 and 20 over the original dataset.

For queries Q_{begin} and Q^1_{place} which do not make use of the ontology, we execute them over BerkeleyDB with indexes. For reasoning-involved queries, in fact, further optimization can be done by materializing the ABox of the KB, avoiding the blow-up in query-rewriting. Indeed, DBpedia performs such materialization for the subsumption hierarchy already which we can take advantage of. Thus for sample query Q_{city}, we conduct experiments in the following three different scenarios: (1) using an in-memory cache and materialized DBpedia, (2) using an in-memory cache and non-materialized DBpedia and (3) using non-materialized DBpedia and no cache (denoted as Q_{city}, Q'_{city} and Q''_{city} respectively).

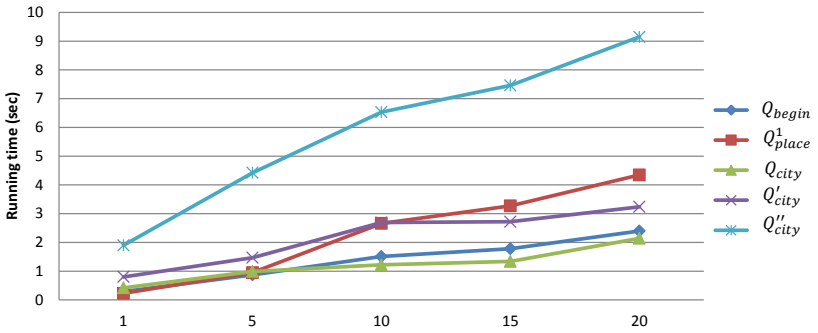


Figure 4.13: Time performance

4.4.3 The user Interface

In this section, we describe the graphical user interface of Quasar. A screenshot of the GUI is shown in Figure 4.14. The GUI allows a user to either choose a predefined query from a collection of samples in panel (B) or compose a new one from scratch in (A). She can query all the annotated corpora registered in the system's corpora directory (see (E)). She can browse in (F) the vocabulary of the KB and configure the corresponding information, e.g., the URL of the TBox. A console in panel (G) prints out the operating messages for the users.

The central tabbed pane (D) visualizes the returned query answers. The GUI provides users with four modes to preview the set of resulting annotations returned by our query engine: a "Plain list" view, "Group by label" view, "Group by document" view, and "Group by instance" view. For any of these "grouping modes", the group names with results are shown along with the total number of results per group. By clicking on each group entry, the user can further preview the annotation sublist of the corresponding group. Here, each annotation is represented as a highlighted snippet, along with a window giving its context.

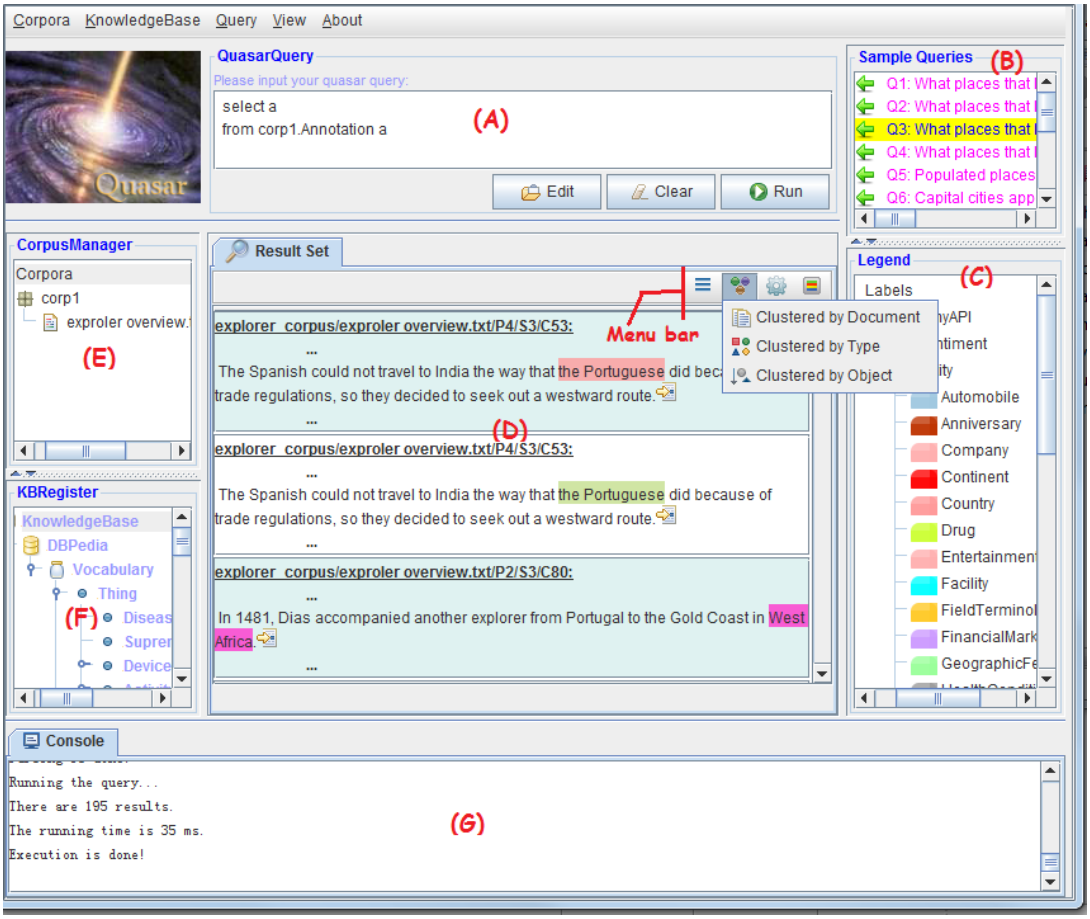


Figure 4.14: GUI of QUASAR

From the "Legend" pane (C) , the user can explore the annotation tasks and corresponding vocabularies supported by each annotator. For queries with *OntologyFilter*, the user can browse the results of the rewritten query produced by the reasoner (as shown in Figure 4.15 and Figure 4.16) .

Filter: subClassOf(?A,Place)

AnnotationPredicate	Annotator
Place	-
Park	-
WorldHeritageSite	-
GeographicFeature	AlchemyAPI
Region	OpenCalais & AlchemyAPI
NaturalFeature	OpenCalais
PopulatedPlace	-
WineRegion	-
HistoricPlace	-
River	-

Figure 4.15: Example of concept filter

Filter: Q(?x) <- Politician(?y), birthPlace(?y,?x)

Q =

KB Constraints
Q(?x) <- Politician(?y), birthPlace(?y,?x)
Q(?x) <- Deputy(?y), birthPlace(?y,?x)
Q(?x) <- VicePrimeMinister(?y), birthPlace(?y,?x)
Q(?x) <- Governor(?y), birthPlace(?y,?x)
Q(?x) <- Congressman(?y), birthPlace(?y,?x)
Q(?x) <- Mayor(?y), birthPlace(?y,?x)
Q(?x) <- MemberOfParliament(?y), birthPlace(?y,?x)
Q(?x) <- President(?y), birthPlace(?y,?x)
Q(?x) <- Chancellor(?y), birthPlace(?y,?x)
Q(?x) <- Senator(?y), birthPlace(?y,?x)
Q(?x) <- VicePresident(?y), birthPlace(?y,?x)

Figure 4.16: Example of instance filter

When the user selects an item from the list of annotations, the system shows the whole

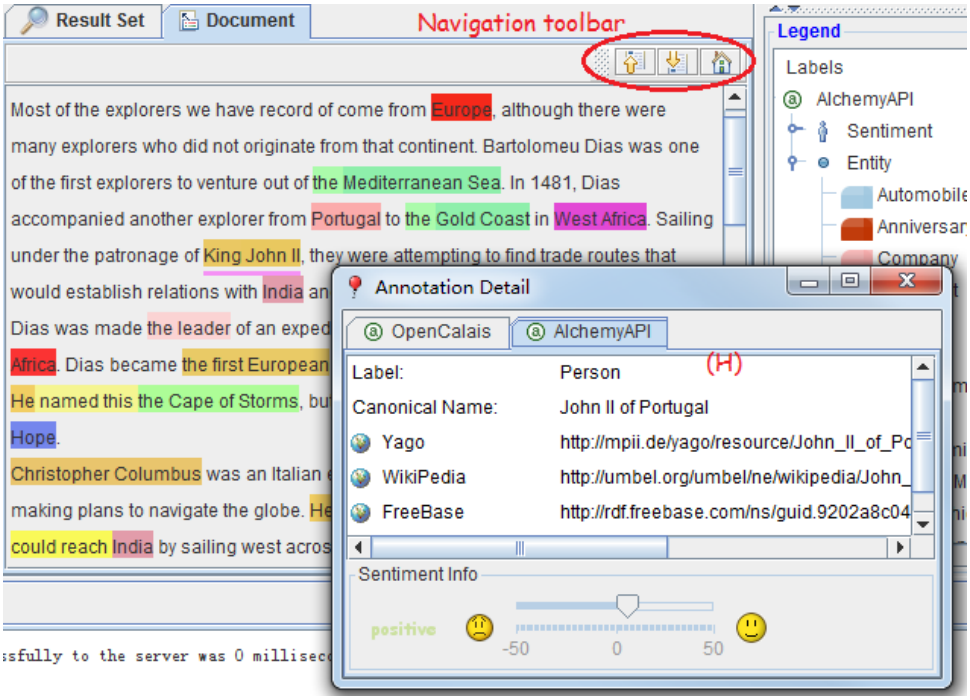


Figure 4.17: Navigating annotations

text region (the document by default) together with the filtered annotations it contains (see Figure 4.17 as an example). The user can navigate the annotations by pressing the *next* or *previous* buttons in the toolbar. If she is interested in one particular highlighted snippet, by mousing over the region she will see a description of all the corresponding annotations associated with the snippet, including annotators, annotation types, and the participating entities from both annotators and the KB (if applicable) (see (H) in Figure 4.17). The user can click the entity URI links to browse more information from external Linked Data and Web assets such as DBpedia and YAGO.

4.5 Summary

QUASAR is a "proof-of-concept" system in devising a rich querying environment which enhances structured querying on documents with access to annotation structure and reasoners. The goal is to allow melding traditional structured querying with ontology and annotation access. We proposed the corresponding query language, describing the syntax and semantics. A prototype system was implemented, which shows how the interface to such a system might look, and how it can allow users to specify their information needs with greater precision.

4.6 Related work

In contrast to QUASAR, there has been considerable activity in the DB and IR research communities that take keyword querying as a starting point and enhance it with some support for "semantics" – e.g. making use of a knowledge base. Many of these maintain the use of a keyword-based query interface but make use of entity annotations. Others make limited extensions to keyword queries; for example, Ilyas and Pound's QUICK system [71] provides support for queries that supplement keywords with structured entity/relationship annotations. As with keyword queries, they do not use a "hard" boolean semantics for the language – instead the query processor looks first for entities in a knowledge base that match the annotations; it then uses these entities to search for relevant documents.

Other work targets enhancing keyword search by exploiting the output of semantic annotators. The KIM platform [49] supports access to semantically annotated documents, but with no full query language for accessing the annotation and document structure in tandem. The DOCQS system of Zhou, Cheng, and Chang [100] provides a language for combining keyword search with matching of entities produced by an entity extractor. Their language does not support access to a reasoner. On the other hand, their query languages do support more powerful structure manipulation operations than ours does, such as aggregation and grouping.

Chapter 5

Conclusions and Future work

Semantic annotation has gained tremendous popularity for more than a decade, since it lays the foundation for bridging the gap between text and useful information. Focusing on integrating and querying semantic annotations, this thesis includes the following work:

- We have benchmarked the state-of-the-art semantic annotators through quantitative comparisons, providing a deep insight into the ecosystem of modern semantic annotators. The benchmarking result reveals that these annotators suffer from wide disparities in aspects such as the coverage and annotation quality, which evidences the need to integrate the semantic annotators.

- We have presented a supervised approach based on Maximum Entropy Markov Model for knowledge-aware integration of entity annotators. As for an internal comparison, we also proposed an unsupervised approach based on a knowledge-aware "voting" scheme. Both approaches consider each annotator as a black box and do not assume any prior knowledge about their performance or competence w.r.t. a certain set of entities. The overall superior performance against individual annotators and external competitors shows clearly that our approaches can benefit the quality of resulting annotation.

- We have also explored a set of supervised alternatives to MEMM, considering many different aspects including: keeping multiple answers, keeping quantitative data, using additional features of context and employing hybrid models. We observed from the empirical evaluation that these methods showed their advantages in different scenarios. The work has provided users with a broader perspective of applying MEMM-based approaches to reconciling semantic annotations.

- We have implemented the ROSeAnn system, as well as the corresponding JAVA APIs and web service APIs for end users or applications to make use of func-

tionalities of ROSeAnn. Our framework was designed with extensibility, offering developers the flexibility to extend the core components.

- We have proposed a structured query language for querying the implicit and explicit knowledge of semantically-enriched documents. The benefit stems from its ability to filter query results based on the presence of annotations in diverse annotation vocabularies, where the filtering specification takes advantage of the multiple kinds of relationships within an annotated document.

- We have implemented the QUASAR prototype system, which is the first step towards devising a rich querying environment which enhances structured querying on documents with access to the annotation structure and reasoners.

Future Work

Annotation reconciliation. One limitation of ROSeAnn is that it performs reconciliation without distinguishing whether an annotation has strong or weak support for correctness, but each of our aggregators can be adapted to give a measure of confidence. For example, for MEMM we can return the probability of the given annotation at each span. This is conceptually straightforward, since it is a particular kind of query on the probabilistic annotation generated by the Markov sequence. We are currently investigating a general approach to efficiently query Markov sequences, tuned to the case of annotations. We are also preparing a study of the adaptation of other supervised learning methods, such as CRFs, which avoid the inherent "label-bias" problem of MEMM , for the annotation reconciliation setting, along with a comparison of the resulting adapted method to MEMM.

Another interesting direction is to progressively reduce the dependency on the annotator's documentation when creating the mapping between the recognized concept classes and the merged ontology. This can be addressed either by schema/ontology matching techniques or by ontology learning approaches. For now, the reference implementation of WR and MEMM called ROSeAnn simply reports newly discovered classes, leaving to the programmer the task of updating the mapping to the merged ontology.

We have focused on concept annotation here, since it is the most widely-supported function within current annotators. Our methods can be extended to do entity concept annotation reconciliation jointly with reconciliation of *entity instance disambiguation* – e.g. by considering instances as bottom-level elements of an ontology. Again, ROSeAnn simply reports the disambiguation information provided by the annotators. A more

difficult task is *relation extraction*, which is also emerging in service-based annotators: it currently has some support from OpenCalais and Extractiv. This is important from the point of view of information extraction, since it identifies part-whole relationships that are needed to piece together a tuple or an object. Our focus on concept annotation allows us to use a very simple kind of reasoning – based on subsumption and disjointness. More costly reasoning would be needed for relations, since the ontology may have richer constraints (e.g. foreign keys, uniqueness constraints) available at the relationship level.

Furthermore, apart from the natural applications in entity extraction systems, we are currently considering to employ roseann in other application domains. For example, in the setting of DIADEM [21] whose goal is to transform unstructured web information into highly structured data without human supervision, roseann can be used to assist the analysis of wrapper induction. The ability of roseann to integrate multiple annotators is of paramount importance for reducing the disadvantages of domain-specific annotators. Also, the ability of roseann to annotate different sections of the DOM with different annotator pools, together with its reconciliation ability, reduces the noise in the annotations for annotation-driven wrapper inducers.

Querying semantic annotations. The expressiveness of the QUASAR query language needs to be enhanced, for example, by being able to deal with "dynamically-generated ABoxes". A dynamic ABox could be one which is composed on-the-fly by a query from the document's annotations within a scope or region. Consider again the annotated document in Figure 4.1. A query could ask for entity instances x such that(i) x is an explorer and (ii) from $\{\mathcal{KB}, \mathcal{A}\}$, where the ABox $\mathcal{A}$ is composed of document's annotations within the first paragraph, we can derive that x was born in a municipality of Portugal. Such queries require an extension of the query language to introduce extra variables and expressions for dynamic ABoxes, where sub-queries need to be handled.

Another drawback of QUASAR is that it can only support fixed structure for free text, such as paragraphs and sentences. Thus, QUASAR can not be applied as-is to structured or semi-structured documents such as XML. Note that it is a limitation of the implementation, not the general model. Thus, another direction to improving QUASAR is to extend the system to support structured documents.

As an annotation management and querying system, QUASAR should have the ability to explicitly distinguish the reliability of annotations that are populated by annotators from those known to be reliable (e.g., the assertions in a KB). Two possible solutions are:(i) incorporate some reconciliation strategies (e.g., as proposed in Chapter 3) to manage the uncertainty transparently to end users; (ii) allow explicit specification of the constraints about the reliability in the query language (e.g., only return answers with confidence scores above a given threshold).

In terms of the implementation, query optimization has not been seriously considered in the QUASAR prototype. For example, the join order is fixed in the prototype. We always treat the results of the structured part as the outer table when performing the joins between annotation results and KB results, assuming that the volume of the annotation store is usually much smaller than a KB. However, in practice, the selectivity of sub-queries can vary from one query to another. Thus it requires a query optimizer with "smarter" strategies to determine the best query plan for evaluation.

References

[1] Alchemyapi. `www.alchemyapi.com/api/entity/`, checked 2014/06/11.

[2] Enrique Alfonseca and Suresh Manandhar. An unsupervised method for general named entity recognition and automated concept discovery. In *Proceedings of the 1st International Conference on General WordNet*, GWC '02, Mysore, India, 2002.

[3] Javed A. Aslam and Mark Montague. Models for metasearch. In *Proceedings of the 27th International Conference on Very Large Data Bases*, SIGIR '01, pages 276–284, New York, NY, USA, 2001. ACM.

[4] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2670–2676, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[5] Roberto Basili, Alessandro Moschitti, Maria Teresa Pazienza, and Fabio Mossimo Zanzotto. Personalizing web publishing via information extraction. *IEEE Intelligent Systems*, 18(1):62–70, 2003.

[6] Robert Baumgartner, Sergio Flesca, and Georg Gottlob. Visual web information extraction with lixto. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 119–128, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[7] Paul Buitelaar and Melanie Siegel. Ontology-based information extraction with soba. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, LREC '06, pages 2321–2324, Genoa, Italy, 2006. ELRA-ELDA.

[8] Rafael Carreira, Sónia Carneiro, Rui Pereira, Miguel Rocha, Isabel Rocha, Eugénio Ferreira, and Analia Lourenço. Semantic annotation of biological concepts interplaying microbial cellular responses. *BMC Bioinformatics*, 12:1–10, 2011.

[9] Soumen Chakrabarti, Kriti Puniyani, and Sujatha Das. Optimizing scoring functions and indexes for proximity search in type-annotated corpora. In *Proceedings of the 15th International Conference on World Wide Web*, WWW '06, pages 717–726, New York, NY, USA, 2006. ACM.

[10] Luying Chen, Michael Benedikt, and Evgeny Kharlamov. Quasar: Querying annotation, structure, and reasoning. In *Proceedings of the 15th International Conference on Extending Database Technology*, EDBT '12, pages 618–621, New York, NY, USA, 2012. ACM.

[11] Luying Chen, Stefano Ortona, Giorgio Orsi, and Michael Benedikt. Aggregating semantic annotators. *Proceedings of the VLDB Endowment*, 6(13):1486–1497, August 2013.

[12] Luying Chen, Stefano Ortona, Giorgio Orsi, and Michael Benedikt. Roseann: Reconciling opinions of semantic annotators. *Proceedings of the VLDB Endowment*, 6(12):1238–1241, August 2013.

[13] Tao Cheng, Xifeng Yan, and Kevin Chen-Chuan Chang. Supporting entity search: a large-scale prototype search engine. In *Proceedings of the 26th ACM SIGMOD International Conference on Management of data*, SIGMOD '07, pages 1144–1146, New York, NY, USA, 2007. ACM.

[14] Hai Leong Chieu and Hwee Tou Ng. Named entity recognition with a maximum entropy approach. In *Proceedings of the 7th Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 160–163, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[15] Philipp Cimiano, Siegfried Handschuh, and Steffen Staab. Towards the self-annotating web. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 462–467, New York, NY, USA, 2004. ACM.

[16] Fabio Ciravegna and Alberto Lavelli. Learningpinocchio: Adaptive information extraction for real world applications. *Natural Language Engineering*, 10(2):145–165, June 2004.

[17] Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 109–118, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[18] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. GATE: an architecture for development of robust HLT applications. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 168–175, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[19] James R. Curran and Stephen Clark. Language independent ner using a maximum entropy tagger. In *Proceedings of the 7th Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 164–167, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[20] Nilesh N. Dalvi, Ravi Kumar, and Mohamed A. Soliman. Automatic wrappers for large scale web extraction. *Proceedings of the VLDB Endowment*, 4(4):219–230, 2011.

[21] Diadem. `http://diadem.cs.ox.ac.uk`, checked 2014/06/11.

[22] George R. Doddington, Alexis Mitchell, Mark A. Przybocki, Lance A. Ramshaw, Stephanie Strassel, and Ralph M. Weischedel. The automatic content extraction (ace) program - tasks, data, and evaluation. In *Proceedings of the 4th Language Resources and Evaluation Conference*, LREC '04, Lisbon, Portugal, 2004. European Language Resources Association.

[23] Xin Luna Dong, Laure Berti-Equille, and Divesh Srivastava. Integrating conflicting data: the role of source dependence. *Proceedings of the VLDB Endowment*, 2(1):550–561, 2009.

[24] Deborah Duong, James Venuto, Ben Goertzel, Ryan Richardson, Shawn Bohner, and Edward Fox. Support vector machines to weight voters in a voting system of entity extractors. In *Proceedings of the International Joint Conference on Neural Networks*, IJCNN '06, pages 1226–1230. IEEE, 2006.

[25] Marieke Van Erp and RaphaÃńl Troncy. Learning with the Web: spotting named entities on the intersection of NERD and machine learning. In *Proceedings of the Concept Extraction Challenge at the Workshop on 'Making Sense of Microposts'*, pages 27–30, 2013.

[26] Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Web-scale information extraction in knowitall: (preliminary results). In *Proceedings of the*

13th International Conference on World Wide Web, WWW '04, pages 100–110, New York, NY, USA, 2004. ACM.

[27] Jérôme Euzenat. Semantic precision and recall for ontology alignment evaluation. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI '07, pages 348–353, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[28] Extractiv. `http://extractiv.com/index.html`, checked 2013/04/11.

[29] Ronald Fagin, Benny Kimelfeld, Yunyao Li, Sriram Raghavan, and Shivakumar Vaithyanathan. Understanding queries in a search database system. In *Proceedings of the 29th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '10, pages 273–284, New York, NY, USA, 2010. ACM.

[30] David Ferrucci and Adam Lally. Building an example application with the unstructured information management architecture. *IBM Systems Journal*, 43(3):455–475, July 2004.

[31] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[32] Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. Named entity recognition through classifier combination. In *Proceedings of the 7th conference on Natural Language Learning at HLT-NAACL 2003*, CONLL '03, pages 168–171, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[33] Fox. `http://ontowiki.net/Projects/FOX?v=4e5`, checked 2014/06/11.

[34] Rob Gaizauskas, George Demetriou, Pete Artymiuk, and Peter Willett. Protein structures and information extraction from biological texts: the PASTA system. *Journal of Bioinformatics*, 19(1):135–143, 2003.

[35] Alban Galland, Serge Abiteboul, Amélie Marian, and Pierre Senellart. Corroborating information from disagreeing views. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 131–140, New York, NY, USA, 2010. ACM.

[36] Murthy Ganapathibhotla and Bing Liu. Mining opinions in comparative sentences. In *Proceedings of the 22nd International Conference on Computational Linguistics*, COLING '08, pages 241–248, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[37] Rahul Gupta and Sunita Sarawagi. Creating probabilistic databases from information extraction models. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, VLDB '06, pages 965–976. VLDB Endowment, 2006.

[38] Stephan Hartmann, Gabriella Pigozzi, and Jan Sprenger. Reliable methods of judgement aggregation. *Journal of Logic and Computation*, 20(2):603–617, 2010.

[39] Guoping Hu, Jingjing Liu, Hang Li, Yunbo Cao, Jian-Yun Nie, and Jianfeng Gao. A supervised learning approach to entity search. In *Information Retrieval Technology*, volume 4182 of *Lecture Notes in Computer Science*, pages 54–66. Springer Berlin Heidelberg, 2006.

[40] Minqing Hu and Bing Liu. Mining opinion features in customer reviews. In *Proceedings of the 19th National Conference on Artifical Intelligence*, AAAI'04, pages 755–760. AAAI Press, 2004.

[41] T. S. Jayram, Rajasekar Krishnamurthy, Sriram Raghavan, Shivakumar Vaithyanathan, and Huaiyu Zhu. Avatar information extraction system. *IEEE Data Engineering Bulletin*, 29:40–48, 2006.

[42] JenaAPI. `http://jena.apache.org/`, checked 2013/04/11.

[43] Sham Kakade, Yee Whye Teh, and Sam T. Roweis. An alternate objective function for markovian fields. In *Proceedings of the 19th International Conference on Machine Learning*, ICML '02, pages 275–282, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[44] Nanda Kambhatla. Minority vote: at-least-n voting improves recall for extracting relations. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, COLING-ACL '06, pages 460–466, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[45] Jaap Kamps, Shlomo Geva, and Andrew Trotman. Report on the sigir 2008 workshop on focused retrieval. *SIGIR Forum*, 42:59–65, November 2008.

[46] Vangelis Karkaletsis, Constantine D. Spyropoulos, Claire Grover, Maria Teresa Pazienza, José Coch, and Dimitris Souflis. A platform for cross-lingual, domain and user adaptive web information extraction. In *Proceedings of the 16th European Conference on Aartifical Intelligence*, ECAI '04, pages 725–729. IOS Press, 2004.

[47] Kim Platform. `https://www.ontotext.com/kim`, checked 2014/06/11.

[48] Benny Kimelfeld and Christopher Ré. Transducing markov sequences. In *Proceedings of the 29th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '10, pages 15–26, New York, NY, USA, 2010. ACM.

[49] Atanas Kiryakov, Borislav Popov, Ivan Terziev, Dimitar Manov, and Damyan Ognyanoff. Semantic annotation, indexing, and retrieval. *Journal of Web Semantics*, 2(1):49 – 79, 2004.

[50] Zornitsa Kozareva, Oscar FerrÃąndez, Andres Montoyo, Rafael MuÃśoz, O SuÃąrez, and Sistemas InformÃątics. Combining data-driven systems for improving named entity recognition. *Data and Knowledge Engineering*, 61(3):449–466, 2007.

[51] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[52] Yang Li, Chi Wang, Fangqiu Han, Jiawei Han, Dan Roth, and Xifeng Yan. Mining evidences for named entity disambiguation. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 1070–1078, New York, NY, USA, 2013. ACM.

[53] LingPipe. `http://alias-i.com/lingpipe/`, checked 2014/06/11.

[54] Feifan Liu, Deana Pennell, Fei Liu, and Yang Liu. Unsupervised approaches for automatic keyword extraction using meeting transcripts. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 620–628, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[55] Linked Open Data. `http://linkeddata.org/`, checked 2013/04/11.

[56] Lupedia. `http://lupedia.ontotext.com/`, checked 2013/04/11.

[57] Liping Ma and John Shepherd. Information extraction using two-phase pattern discovery. In *Proceedings of the 27th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, pages 534–535âŰň, New York, NY, USA, 2004. ACM.

[58] Alexander Maedche, Günter Neumann, and Steffen Staab. Bootstrapping an ontology-based information extraction system. In *Studies in Fuzziness and Soft Computing, Intelligent Exploration of the Web*, pages 345–359. Physica-Verlag GmbH, Heidelberg, Germany, Germany, 2003.

[59] Hassan H. Malik, Vikas S. Bhardwaj, and Huascar Fiorletta. Accurate information extraction for quantitative financial events. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 2497–2500, New York, NY, USA, 2011. ACM.

[60] Andrew McCallum, Dayne Freitag, and Fernando C. N. Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the 17th International Conference on Machine Learning*, ICML '00, pages 591–598, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[61] Pablo N. Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. DBpedia Spotlight: shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*, I-Semantics '11, pages 1–8, New York, NY, USA, 2011. ACM.

[62] Eirinaios Michelakis, Rajasekar Krishnamurthy, Peter J. Haas, and Shivakumar Vaithyanathan. Uncertainty management in rule-based information extraction systems. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, pages 101–114, New York, NY, USA, 2009. ACM.

[63] George A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, November 1995.

[64] Muc7. `http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2001T02`, checked 2014/06/11.

[65] Michael A. Olson, Keith Bostic, and Margo Seltzer. Berkeley db. In *Proceedings of the annual conference on USENIX Annual Technical Conference*, ATEC '99, pages 43–43, CA, USA, 1999. USENIX Association Berkeley.

[66] Opencalais. `http://www.opencalais.com/`, checked 2013/04/11.

[67] OpenNLP-MAXENT-3.0.0. http://maxent.sourceforge.net/about.html, checked 2014/06/11.

[68] OpenNLP Tools. http://opennlp.apache.org/index.html, checked 2014/06/11.

[69] Stefano Ortona, Luying Chen, and Giorgio Orsi. Roseann: Taming online semantic annotators. In *Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web Companion*, WWW Companion '14, pages 591–594, 2014.

[70] Héctor Pérez-Urbina, Ian Horrocks, and Boris Motik. Practical Aspects of Query Rewriting for OWL 2. In *Proceedings of the 5th International Workshop on OWL: Experiences and Directions*, OWLED '09, 2009.

[71] Jeffrey Pound, Ihab F. Ilyas, and Grant Weddell. Quick: expressive and flexible search over knowledge bases and text collections. *Proceedings of the VLDB Endowment*, 3(1-2):1573–1576, September 2010.

[72] Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the 13th Conference on Computational Natural Language Learning*, CoNLL '09, pages 147–155, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[73] Frank Reichartz, Hannes Korte, and Gerhard Paass. Semantic relation extraction with kernels over typed dependency trees. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 773–782, New York, NY, USA, 2010. ACM.

[74] Giuseppe Rizzo and Raphaël Troncy. Nerd: a framework for unifying named entity recognition and disambiguation extraction tools. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 73–76, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[75] Riccardo Rosati. On the complexity of dealing with inconsistency in description logic ontologies. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence - Volume Volume Two*, IJCAI '11, pages 1057–1062. AAAI Press, 2011.

[76] ROSeAnn. http://diadem.cs.ox.ac.uk/roseann, checked 2014/06/11.

[77] Saplo. http://saplo.com/, checked 2013/04/11.

[78] Sunita Sarawagi. Scalable information extraction and integration. In *Proceedings of the 13th International Conference on Management of Data*, COMAD '06, page 249. Tata McGraw-Hill Publishing Company Limited, 2006.

[79] Pierre Senellart, Avin Mittal, Daniel Muschick, Rémi Gilleron, and Marc Tommasi. Automatic wrapper induction from hidden-web sources with domain knowledge. In *Proceedings of the 10th ACM Workshop on Web Information and Data Management*, WIDM '08, pages 9–16, New York, NY, USA, 2008. ACM.

[80] B. Settles. ABNER: An open source tool for automatically tagging genes, proteins, and other entity names in text. *Bioinformatics*, 21(14):3191–3192, 2005.

[81] Warren Shen, AnHai Doan, Jeffrey F. Naughton, and Raghu Ramakrishnan. Declarative information extraction using datalog with embedded extraction predicates. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB '07, pages 1033–1044. VLDB Endowment, 2007.

[82] Luo Si, Tapas Kanungo, and Xiangji Huang. Boosting performance of bio-entity recognition by combining results from multiple systems. In *Proceedings of the 5th International Workshop on Bioinformatics*, BIOKDD '05, pages 76–83, New York, NY, USA, 2005. ACM.

[83] Joaquim Ferreira da Silva, João Mexia, Carlos Agra Coelho, and José Gabriel Pereira Lopes. Document clustering and cluster topic extraction in multilingual corpora. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, ICDM '01, pages 513–520, Washington, DC, USA, 2001. IEEE Computer Society.

[84] Andrew Smith and Miles Osborne. Using gazetteers in discriminative information extraction. In *Proceedings of the 10th Conference on Computational Natural Language Learning*, CoNLL-X '06, pages 133–140, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[85] Rohini Srihari and Wei Li. A question answering system supported by information extraction. In *Proceedings of the 6th Conference on Applied Natural Language Processing*, ANLC '00, pages 166–172, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[86] Rohini Srihari, Cheng Niu, and Wei Li. A hybrid approach for named entity and sub-type tagging. In *Proceedings of the 6th Conference on Applied Natural Language*

Processing, ANLC '00, pages 247–254, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[87] Fabian M. Suchanek, Mauro Sozio, and Gerhard Weikum. Sofie: a self-organizing framework for information extraction. In *Proceedings of the 18th International Conference on World wide web*, WWW '09, pages 631–640, New York, NY, USA, 2009. ACM.

[88] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: language-independent named entity recognition. In *Proceedings of the 7th Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 142–147, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[89] Kees van Deemter and Rodger Kibble. On coreferring: coreference in muc and related annotation schemes. *Computational Linguistics*, 26:629–637, 1995.

[90] Virtuoso. `http://virtuoso.openlinksw.com`, checked 2014/06/11.

[91] Daisy Zhe Wang, Michael J. Franklin, Minos Garofalakis, and Joseph M. Hellerstein. Querying probabilistic information extraction. *Proceedings of the VLDB Endowment*, 3(1-2):1057–1067, September 2010.

[92] Haochang Wang and Tiejun Zhao. Identifying named entities in biomedical text based on stacked generalization. In *Proceedings of the 7th World Congress on Intelligent Control and Automation*, WCICA '08, pages 160–164, 2008.

[93] WebTable. `http://windowsutilities.com/webtable.html`, checked 2014/06/11.

[94] Wikimeta. `http://www.wikimeta.com/`, checked 2013/04/11.

[95] Dekai Wu, Grace Ngai, and Marine Carpuat. A stacked, voted, stacked model for named entity recognition. In *Proceedings of the 7th Conference on Natural Language Learning*, CONLL '03, pages 200–203, 2003.

[96] YahooQL. `https://developer.yahoo.com/contentanalysis/`, checked 2013/04/11.

[97] Yulan Yan, Naoaki Okazaki, Yutaka Matsuo, Zhenglu Yang, and Mitsuru Ishizuka. Unsupervised relation extraction by mining wikipedia texts using information from the web. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of*

the AFNLP, ACL '09, pages 1021–1029, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[98] Zemanta. `http://developer.zemanta.com/`, checked 2014/06/11.

[99] GuoDong Zhou and Jian Su. Named entity recognition using an hmm-based chunk tagger. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 473–480, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[100] Mianwei Zhou, Tao Cheng, and Kevin Chen-Chuan Chang. Docqs: A prototype system for supporting data-oriented content query. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 1211–1214, New York, NY, USA, 2010. ACM.