Aggregating Semantic Annotators

Luying Chen, Stefano Ortona, Giorgio Orsi, and Michael Benedikt
Oxford University, UK
name.surname@cs.ox.ac.uk

ABSTRACT

A growing number of resources are available for enriching documents with *semantic annotations*. While originally focused on a few standard classes of annotations, the ecosystem of annotators is now becoming increasingly diverse. Although annotators often have very different vocabularies, with both high-level and specialist concepts, they also have many semantic interconnections. We will show that both the overlap and the diversity in annotator vocabularies motivate the need for *semantic annotation integration*: middleware that produces a unified annotation on top of diverse semantic annotators. On the one hand, the diversity of vocabulary allows applications to benefit from the much richer vocabulary available in an integrated vocabulary. On the other hand, we present evidence that the most widely-used annotators on the web suffer from serious accuracy deficiencies: the overlap in vocabularies from individual annotators allows an integrated annotator to boost accuracy by exploiting inter-annotator agreement and disagreement.

The integration of semantic annotations leads to new challenges, both compared to usual data integration scenarios and to standard aggregation of machine learning tools. We overview an approach to these challenges that performs *ontology-aware aggregation*. We introduce an approach that requires no training data, making use of ideas from database repair. We experimentally compare this with a supervised approach, which adapts maximal entropy Markov models to the setting of ontology-based annotations. We further experimentally compare both these approaches with respect to ontology-unaware supervised approaches, and to individual annotators.

1. INTRODUCTION

A growing number of resources are available for enriching documents with *semantic annotations* – annotations that label document snippets as referring to either certain entities (e.g. Barack Obama, the King of Spain) or to elements of particular semantic categories (e.g. actors, governmental organizations). While originally focused on a few high-level standard classes of annotations (e.g., people, places, organisations), the annotator ecosystem is becoming increasingly diverse. There are high-level and specialist concepts that are supported by multiple annotators and, although their vocabularies are often quite distinct, it is possible to recognise many semantic interconnections among them. There are annotators that deal with very common entity classes (e.g., Person), and those that focus on very specialized entities (e.g., Protein) to support applications domains such as biomedicine [10]. Furthermore, instead of being custom built "white-boxes" that can be directly tuned by developers, annotation is often available as a "black-box" service (e.g. OpenCalais, Zemanta), where the user does not have to implement his own annotator but, on the other hand, concept classes can be added or refined by the service host outside the control of clients.

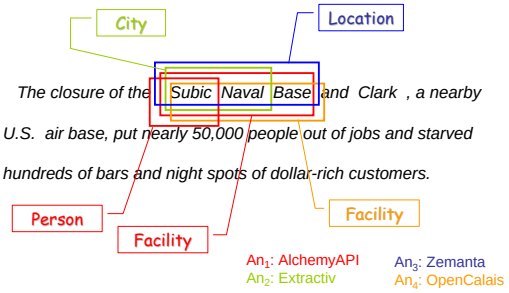


Figure 1: Entity Annotations

Consider the example displayed in Figure 1, 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 (i.e., Location, Person), as well as some more specific ones (i.e., Facility). Notice that the annotation of type City is only returned by Extractiv: indeed, several of the annotators simply do not have this concept in their vocabulary. 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. A second example is shown in Figure 2. Here we see several flavors of differences in annotator opinions. "*Nottingham Forest*" is labeled as a NaturalFeature by OpenCalais, and as a GeographicFeature by AlchemyAPI – clearly these outputs are compatible, since these two concepts represent the same meaning. On the other hand, the annotations Facility and Sports_team are clearly both incompatible with the prior classifications, as well as with each other.

We will show that both the overlap and the diversity in annotator vocabularies motivate the need for *semantic annotation integration*: middleware that produces a unified annotation on top of diverse semantic annotators. On the one hand, the diversity of vocabulary allows applications to benefit from the much richer vocabulary available in an integrated vocabulary. For example, recent work on web-scale wrapper induction (see, e.g., [14, 34]) uses se-

Proceedings of the VLDB Endowment, Vol. 6, No. 13

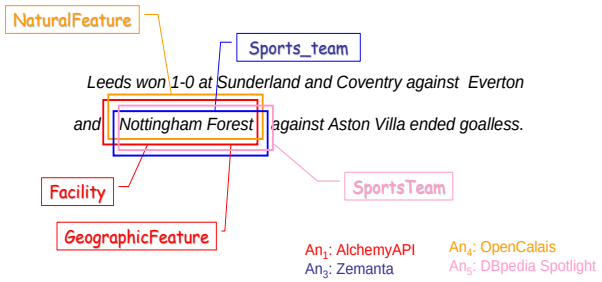


Figure 2: Conflicting and Re-enforcing Annotations

mantic annotations to lower the need for human supervision. On the other hand, annotation integration can also have an impact in terms of quality of the result. We will present experimental evidence (see Section 2) indicating that the most widely-used individual annotators suffer from serious accuracy deficiencies. The overlap in vocabularies from individual annotators could be exploited to allow an integrated annotator to boost accuracy by exploiting inter-annotator agreement and disagreement.

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). In many cases, Web-based annotators provide mappings of instances and concepts to a publicly-available knowledge base – for example DBpedia Spotlight links to the well-known DBpedia ontology. Even in cases where such mappings are not provided, ontological rules can often be conservatively inferred via commonsense reasoning (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 look for integration methods that are *ontology-aware* – making as much use as possible of semantic 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. 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.

We begin by describing an approach for on-line aggregation in the absence of training data. We present a *weighted repair* aggregation algorithm that assigns a score to annotations based on the explicit opinions of individual aggregators (an annotator tags a snippet with a class) implicit support (an annotator tags with a subclass), and implicit opposition (e.g. an annotator tags with a disjoint class). This method takes into account that annotators have distinct vocabularies, thus the lack of a certain annotation may not indicate lack of support. The weighted repair approach looks for an aggregation which is consistent w.r.t. the ontology. We show that for basic ontological constraints arising in the setting of concept reasoning, an optimal solution can be found efficiently.

An obvious comparison point would be a system that can benefit from off-line training in case training data is available. In this paper, we describe an approach that uses ideas from supervised learning. It uses a maximal entropy model, adapted to the setting of semantic annotation. Both our supervised and unsupervised approaches support overlapping or nested spans, an occurrence that is not uncommon in Web-based annotators.

We experimentally compare these approaches both with respect to each other and to a number of external baselines. Our results show clear benefits for aggregation of annotation, while also giving new insight into the behavior of Web-based semantic annotators.

Organization. Section 2 gives background on annotators in general, then describes the annotators we use in our examples along with their mapping to a common ontology. It then overviews the accuracy of the annotators, making use of a scoring model that is tied to the ontology. Section 3 describes our aggregation algorithms. Section 4 discusses the implementation of the algorithms, and gives an experimental evaluation. Section 5 overviews related work, while Section 6 gives conclusions.

2. BENCHMARKING ANNOTATORS

Semantic annotators take as input text documents and produce semantic annotations in a variety of forms. The most common ones are *concept* labels, e.g., for named entity recognition, *entity* referencing, e.g., for named-entity disambiguation, and *relation* (or role) labels, e.g., for relation extraction. Of these, the first is by far the most mature, and we focus on it in this work. Such an annotation consists of a contiguous segment of a document – namely, a *span* – along with a concept (or class, or *entity type*: we use these interchangeably throughout the paper), such as Person, Organization, or Artist. The example from Figure 1 shows five concept annotations. Three of them (AN_1:Facility, AN_4:Facility, and AN_3:Location) share the same span, with one of these (AN_3:Location) disagreeing with the other two on the concept name. Two others (AN_1:Person and AN_2:City) have different (but overlapping) spans.

The state-of-the-art named entity extractors include software frameworks such as LingPipe [2] and Gate [23], and web service APIs such as OpenCalais, Alchemy, Extractiv, Spotlight and Zemanta. The service-based model has particular interest from the perspective of applications which want to exploit a wide vocabulary. Each annotator may focus on a particular subset of the concepts. They can make use of their own gazetteers and hand-tailored concept-specific heuristics, allowing these to remain proprietary, at the same time freeing applications from having to constantly upload or update whenever the annotator is tuned. On the other hand, the service-based model poses specific challenges to integrators, since the performance of the annotators, and even the vocabulary, may be highly dynamic.

Thus in our work we will target web service-based annotators, focusing on the task of named entity recognition, which is the most widely-supported function. OpenCalais, Extractiv and Spotlight return entities, including their position and entity type, while Alchemy and Zemanta simply return text anchors with their types. More importantly, the taxonomies of named entity types they exploit are quite distinct. OpenCalais deals with 39 types, no pair of which are in a subclass relationship. Most of these are quite general concepts such as Person and Organization. Spotlight and Zemanta adopt existing ontology resources (DBpedia and Freebase respectively) as the labeling vocabulary. Alchemy and Extractiv have taxonomies in which the deepest subclass hierarchy is quite short (usually no more than 2), including both abstract and specialised concepts.

Vocabulary Alignment. As shown in the examples above, the annotators do not have a flat vocabulary – annotations that are syntactically distinct may have the same meaning, like NaturalFeature and GeographicFeature in Figure 2. Some annotators, such as OpenCalais, have mostly high-level concepts in their vocabulary – e.g. Organization; others such as Extractiv, have many much finer classifications – e.g. FinancialOrg, CommercialOrg. Clearly, these

```
<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 3: Example Snippet of Merged Ontology

should not be considered as conflicting within aggregation. To capture the relationships in the vocabulary, we will make use of an ontology Ω which states the relationships between types of entities. In this work, an ontology consists of a finite set of concepts (classes, types), including the special type $\perp$ (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 $C \sqcap D \sqsubseteq \perp$, 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 had 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 needed to be added. We believe that as these annotators mature, the use of standardized vocabularies will increase, making the use of rules as "hard facts", as we do throughout this work, appropriate. An effort in this direction is the Schema.org initiative. We are focusing in this paper 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 and by validating them against Schema.org knowledge base when applicable. 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 3 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 paper 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.

Dataset. In order to assess the performance of those individual annotators, we conducted experiments over four benchmark corpora: **(i)** the MUC7 dataset [3], which contains 300 newswire articles for evaluating Named Entity recognition, annotated with standard high-level concepts such as Person, Organization, Location and Money; **(ii)** the Reuters corpus (RCV1) [5], a general information retrieval corpus of English newswire documents, subdivided into topics; **(iii)** the corpus used by the FOX [1] entity extractor and consisting of 100 snippets of text from news headlines annotated with three concept types, namely Person, Location, and Organisation; **(iv)** the corpus used by the NETAGGER [31] entity extractor and consisting of text sourced from 20 web pages mostly about academics and annotated with the same three concepts above plus a MISC concept that we 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 topics e.g., *Biomolecular Computation* are not. This corpus also allowed us to see the performance of our own annotators in a very different domain from Reuters and MUC7. Note that for our supervised approach, MEMM (described in Section 3) we performed training/testing on representative samples of the gold-standards provided by the two datasets. 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 the texts by using the most specific concepts from the global vocabulary of the aggregated ontology discussed above. Details on the coverage of the testset are shown in Table 1. Our gold standard annotation of the Reuters corpus is available online from [6]. Note that for the other corpora, we directly use the original gold standard provided by the benchmarks.

Table 1: Description of Testing Corpora

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

Precision and Recall. We measure PRECISION and RECALL 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 [18] has defined semantic precision and recall measures in the context of ontology alignment – but the goal there is to assign static measures to an alignment, which approximate 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:

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

Evaluation Result of Individual Annotators. We tested individual annotators over the test 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 [6].

We present selected results in Table 2, where highlighted values represent the best performance. For each annotator AN, we display results for all of the MUC7, FOX, and NETAGGER concepts that are in the vocabulary of AN. For each corpus, we first mapped the response of individual annotators into the ones with the entity 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 with sufficient support, diversity in performance, and diversity within the ontology.

We see from the table that each annotator varied 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. 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. 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, was 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.

Conflicts of Individual Annotators. Given that 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 analyzed one coarse measure 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 "differing 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 3 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) occurred, and the total number of conflicts of each type. An annotated span is a span which contained at least one annotation.

Table 3: Conflicts Statistic 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
NETagger	1,493	1,486	1,195	179	83

We have also tracked conflicts in spans, e.g. annotators agreeing on a concept, but disagreeing on the span. Table 4 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 into 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).

Table 4: 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
NETagger	5,430	7,029	4,719	263	4,982

Although one might expect that conflicts are rare, the results show that they are extremely frequent. A chief source of conflict 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.

Overall, the results show both the need for annotator integration and the possibility of using conflict and agreement signals in constructing superior aggregate annotations. The results are *not* meant to indicate an intrinsic pattern of who is correct on which concepts. Results can vary as different datasets are used; further, these annotators are frequently modified, with the modifications impacting both their vocabularies and their scope. Thus our goal in the rest of the paper will not be to tune an aggregation algorithm to this particular dataset (e.g. by hard-coded rules capturing which annotator is seen to be reliable), but to come up with a general domain-independent methodology for aggregating annotator values, not relying on hand-tuned rules or weights.

3. AGGREGATION ALGORITHMS

Unsupervised aggregation via weighted repair. We introduce an aggregation approach that requires no supervision and no information on the reliability of aggregators.

Aggregation of conflicting data in the presence of integrity constraints is a long-standing topic in AI, semantic web, and database research. In the case where the input consists of a set of opinions from multiple "experts", there are a number of *judgement aggregation* methods that have been studied – for example, based on variations of *voting* [24]. These algorithms have been examined not

Table 2: Performance of individual annotators.

		MUC7							NETagger			Fox			Reuters				
		Date	Organization	Person	Location	Money	Percent	Time	Person	Location	Organization	Person	Location	Organization	Person	Location	Organization	Date	Movie
ALCHEMY	Prec.	.50	.88	.80	.90	-	-	-	.81	.63	**.74**	**1**	**.89**	**.79**	.75	**.92**	.80	.68	.42
	Rec.	-	.49	.55	.69	-	-	-	**.75**	**.52**	.45	**.92**	.89	.62	.55	.70	.47	-	.20
	F_1	-	.49	.65	.78	-	-	-	**.78**	**.57**	.56	**.96**	**.89**	**.70**	.64	.64	.60	.02	.27
SPOTLIGHT	Prec.	-	.63	.70	.76	-	-	-	.58	.30	.66	**1**	.75	.65	.71	.81	.58	-	.12
	Rec.	-	.40	.22	.46	-	-	-	.14	.40	.14	.40	.44	.33	.31	.57	.38	-	.29
	F_1	-	.49	.34	.57	-	-	-	.23	.34	.23	.57	.56	.43	.42	.67	.46	-	.17
EXTRACTIV	Prec.	**.91**	**.89**	.87	.80	**.98**	**1**	**.95**	.82	.46	.50	.80	.72	.69	.81	.75	.76	**.85**	**.54**
	Rec.	**.59**	.72	.71	**.78**	.68	**.89**	**.11**	.44	.48	.34	.73	.95	.65	.70	.76	.60	**.84**	.42
	F_1	**.71**	**.80**	.78	.79	.80	**.94**	**.21**	.57	.47	.41	.76	.82	.67	.75	.76	.67	**.85**	.47
OPENCALAIS	Prec.	.60	.85	**.97**	**.94**	.93	-	-	.83	**.64**	.69	.80	.87	.69	**.95**	.92	**.85**	.68	.51
	Rec.	-	**.73**	**.88**	.71	**.92**	-	-	.52	.33	**.69**	.87	**.90**	**.67**	**.87**	**.76**	**.69**	.01	**.53**
	F_1	.01	.79	**.92**	**.81**	**.93**	-	-	.64	.44	**.69**	.83	.89	.68	**.91**	**.83**	**.76**	.02	**.52**
ZEMANTA	Prec.	-	.70	.89	.72	-	-	-	**.92**	.36	.52	.62	.76	.67	.64	.83	.70	-	.45
	Rec.	-	.28	.07	.17	-	-	-	.10	.38	.07	.58	.75	.54	.16	.40	.23	-	.17
	F_1	-	.40	.13	.28	-	-	-	.17	.37	.13	.60	.76	.60	.26	.54	.35	-	.25

only from the point of view of guaranteeing consistency, but from the perspective of social choice theory – e.g. can they be considered to be fair to the parties involved ([22]). When, on the contrary, the input is seen as a single inconsistent dataset, a common approach in the database and semantic web community is based on *repair*: finding a "minimal" (in some metric) set of update operations to the initial data so that the updated data is consistent [33].

Our solution in the unsupervised setting will use ideas from both voting and repair. Assume for the moment a single span $\hat{s}$ that is tagged by several annotators. In this case, concepts can be identified with atomic propositions, and the ontology relationships can be considered as propositional constraints – e.g. if concepts C and D are disjoint, our logical theory includes the constraint $C \rightarrow \neg D$. Thus we can translate the ontology Ω to a propositional theory T_Ω. One component of our solution is a voting-based *AtomicScore* function $\mathsf{AtomicScore}(C)$, assigning an integer to each concept C. This represents the degree of support for or opposition to C by annotators. We will always consider an annotator to support a concept C if it tags the span $\hat{s}$ with a subclass of C – each such supporter will add to $\mathsf{AtomicScore}(C)$. $\mathsf{AtomicScore}(C)$ will drop for each annotator that tags $\hat{s}$ with a class disjoint from C. We allow in addition the possibility of "opposition by omission" – $\mathsf{AtomicScore}(C)$ drops when an annotator fails to tag $\hat{s}$ with a superclass of C that is in its vocabulary. Such a penalty is justified under the assumption that annotators have high recall. The general form of our scoring function allows the possibility of tuning the assumptions about both precision and recall, as well as support for more general concept constraints than simply subsumption and disjointness:

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

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. Lastly $\mathsf{SupportWeight}_{\mathsf{A},D}$ 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. Note that a term in the sum above will only be present if the corresponding type of axiom is asserted (or inferred) in the ontology. The above sum considers two ways for an annotator can "oppose" a concept C, by tagging with a concept disjoint from C or by omitting to tag with (a superclass of) C in its vocabulary. If our ontology language was rich enough to have (e.g.) rules saying that C and D partition all entities, we would also have an additional term representing "support by omission": an annotator omitting D is supporting C. Since we have no such rules in our example ontology, we do not examine this setting.

We will discuss ways of setting the weights further below. A straightforward approach is to assume annotators have low recall, and thus annotator A failing to tag with a concept D gives no signal that the entity is not in class D – this corresponds to setting each $\mathsf{OmitWeight}_{\mathsf{A},D}$ to 0, eliminating two of the terms above. In our experiments with wide-spectrum web-based annotators we find that recall is indeed low, and hence we utilize this approach in our implementation (see Section 4 for details). However, for domain-specific annotators, other approaches, such as those described below based on bootstrapping, may be appropriate.

Another assumption would be that the precision does not vary either across annotators or across concepts – hence $\mathsf{SupportWeight}_{\mathsf{A},D}$ can be set uniformly to 1. We will refer to this as *simple atomic scoring*, and under this policy $\mathsf{AtomicScore}(C)$ simplifies to:

$$\Sigma_{D\sqsubseteq C\in\Omega}\{\mathsf{A} \mid \mathsf{A}\text{ annotates with }D\} - \Sigma_{D\sqcap C=\bot\in\Omega}\{\mathsf{A} \mid \mathsf{A}\text{ annotates with }D\}$$

EXAMPLE 1. *Assume the simple atomic scoring model, and consider a span annotated by annotator A_1 with City, annotator A_2 with Country, and annotator A_3 with NaturalFeature. Since these concepts are disjoint in the ontology, under the simple scoring model each of City, Country, and NaturalFeature will have AtomicScore -1, since each is supported by one annotator and opposed by 2. The least common ancestor of these three concepts, namely Location, will have AtomicScore 3.*

Given the atomic scores, a *score combination function* is responsible for calculating a boolean combination of concepts that is consistent with the ontology, based on the scores of constituent literals. For the combination function we propose a declarative approach, denoted as *weighted repair* (WR). We begin with the union of all concepts returned by any annotator, which can be considered as a

conjunction of concepts σ_{Init}. A repair operation Op is either a deletion of a concept occurring as a conjunct within σ_{Init} or an insertion of a concept that is absent from σ_{Init}. The application of Op to σ_{Init} is 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 (the superclasses could also be inserted, but this will have no impact on the calculation below). A set of repairs is *internally-consistent* if no two operations conflict: 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 to be the result of applying the Op_i in any order – one can easily check that this gives a unique result. A repair set is *non-redundant* if we do not delete or insert two concepts in a subclass relation. We call an internally-consistent, non-redundant repair set S such that $S(\sigma_{\mathsf{Init}})$ is consistent with T_Ω a *solution*.

EXAMPLE 2. *Let us return to Example 1. σ_{Init} is City $\wedge$ Country $\wedge$ NaturalFeature. The repair set $\{Del(City), Del(Country)\}$ is internally-consistent and non-redundant, and its application to σ_{Init} gives the formula NaturalFeature $\wedge \neg$Country $\wedge \neg$City. Since this is consistent with the ontology, the repair set is a solution. The repair set $\{Del(City), Del(Country), Del(NaturalFeature)\}$ is another solution, yielding the formula $\neg$NaturalFeature $\wedge \neg$Country $\wedge \neg$City.*

Our goal is to find a solution S such that its *aggregate score* is maximal among all solutions, where the aggregate score of 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 a concept C incurs the penalty $\mathsf{AtomicScore}(C)$, while an insertion of a concept C incurs the negative of $\mathsf{AtomicScore}(C)$ as a penalty.

There can be many repairs that achieve the maximal score, and it is natural to impose other criteria to prune them: 1. Given two solutions with the same score and different numbers of repairs, we prefer the smaller one. 2. Given solution $S_1 = S' \cup \{\mathsf{Ins}(C_1)\}$ $S_2 = S' \cup \{\mathsf{Ins}(C_2)\}$, with C_2 a subclass of C_1, we prefer S_2 – that is we prefer those that insert deeper (that is, more specific) classes. Note that, unlike in other database repair contexts, repair operations can have positive score (net benefit), negative score (net penalty), or zero score (cost neutral, but perhaps needed to gain consistency).

We can solve this optimization problem by a reduction to integer linear programming (ILP). We have variables $X_{\mathsf{Del},C}$ for each concept C that can be deleted, with a value of 1 denoting deletion and 0 denoting no deletion. Similarly we have variables $X_{\mathsf{Ins},C}$ for each concept that can be inserted. Each disjointness or subsumption relation in the ontology corresponds to an integer constraint; e.g. for concepts $B, C \in \Omega$ with $B \in \sigma_{\mathsf{Init}}$ and $C \notin \sigma_{\mathsf{Init}}$, a disjointness constraint $B \sqcap C = \bot$ implies an integer inequality $X_{\mathsf{Ins},C} \geq \Sigma_{B' \preceq B} X_{\mathsf{Del},B'}$ – that is, if we insert C we must delete a subclass of B. The scoring policy described above is easily translation to maximization of a corresponding linear objective function.

EXAMPLE 3. *Consider an annotated span with 5 different concepts $C = \{$Commercial_Org, Financial_Org, Educational_Org, Government_Org, Organization_Other$\}$, where all the concepts are mutually disjoint and in subclass relationship with the concept Organization. Moreover Organization is the disjoint union of all the concepts in C. The set of concepts involved is $L = C \cup \{$Organization$\}$. For each concept l involved we have two variables, $X_{Del,l}$ and $X_{Ins,l}$ to determine whether a concept must be deleted or inserted in the final solution. Since each concept in C is already in the initial solution and cannot be inserted, we have $X_{Ins,l} = 0$ $\forall l \in C$. On the other hand, Organization can only be inserted, therefore $X_{Del,organization} = 0$. Note that Organization is the only concept in the example eligible to be inserted in the final solution. For each pair of disjoint concepts in the initial solution we add an integer constraint to ensure that at least one of them will be deleted, that is $X_{Del,l} + X_{Del,h} \geq 1$ $\forall l, h \in C \mid l \sqcap h \sqsubseteq \bot$. Then we add a constraint to ensure that Organization is the disjoint union of classes in C, that is $|C| - \Sigma_{l \in C} X_{Del,l} = X_{Ins,organization}$. Eventually we try to maximize*

$$k * (\Sigma_{l \in L} AtomicScore(l) * X_{Ins,l} - \Sigma_{l \in L} AtomicScore(l) * X_{Del,l})$$
$$-(\Sigma_{l \in L} X_{Del,l} + X_{Ins,l})$$

where k is the total number of concepts involved + 1.

This optimization problem requires (in particular) checking consistency of various boolean combinations with respect to the propositional theory T_Ω. For rich enough ontological rules (enough to give an arbitrary propositional theory), is NP-hard, and the use of a robust ILP solver is needed. In our implementation of the ILP-based approach, we used IBM's ILOG CPLEX as a solver, coupled with OWLIM-Lite to query the ontology. For each span a collection of a few hundred constraints is produced. We can prune these by considering only super-classes of concepts that are returned by one of the annotators on the span; given that the depth of the concept hierarchy is in single digits, this results in a drastic reduction in the number of variables. We also generate constraints corresponding to disjointness axioms from the top of the ontology hierarchy downward, allowing us to suppress redundant disjointness constraints, further reducing the constraint set. Note that the ILP reduction also gives an NP upper bound for the decision problem corresponding to finding an optimal repair (checking the existence of a repair with a given score).

However, in the special case of disjointness and subsumption constraints only, we can find at least one optimal solution tractably, without appeal to ILP:

THEOREM 1. *Assuming only disjointness and subsumption constraints, an annotation set corresponding to some optimal repair set can be achieved by inserting every non-annotated concept of positive AtomicScore, deleting every concept of negative AtomicScore and deleting some concepts with score* 0.

The proof relies on the following simple observation about the AtomicScore function, which can be proven simply by unwinding the definitions:

CLAIM 2. *Given two disjoint concepts C_1 and C_2 at most one can have a positive AtomicScore, and if one does, the other must have negative AtomicScore.*

The proof of Theorem 1 now follows easily: it suffices to consider any repair set S_0 that yields a boolean combination containing all concepts with positive AtomicScore, and which deletes all concepts with AtomicScore ≤ 0. Using the claim above we see that S_0 leaves the database consistent. Now consider S an arbitrary non-redundant repair set of maximal score, whose application yields a consistent formula. First, suppose the application of S deletes a concept C of positive AtomicScore. Thus S contains $\mathsf{Del}(C')$ for C' a superclass of C. Consider the repair S' formed by removing $\mathsf{Del}(C')$ from S. It is easy to see, using the definition of AtomicScore, that $\mathsf{AtomicScore}(C')$ must also be positive. If S' were inconsistent, this means that C' was disjoint from some class D either inserted or left intact by S'. But using the claim we see that any such D that was inserted must have had negative AtomicScore, as would any D left intact. Thus a modification to S by deleting or not inserting any

such D, as appropriate, would result in a repair set that yields consistency and has higher score. Now suppose that the application of S fails to insert a C of positive AtomicScore. We can argue, as above, that the application $S \cup \{\mathsf{Ins}(C)\}$ gives a consistent formula. Finally, we consider a concept C of negative AtomicScore. Clearly adding $\mathsf{Del}(C)$ to S does not lose consistency, and $S \cup \{\mathsf{Del}(C)\}$ will have higher score than S; thus if S has optimal score its application must already delete C.

Theorem 1 gives a simple algorithm for finding an annotation that results from an optimal repair set: start with inserting every non-annotated concept of positive AtomicScore, deleting all concepts that are disjoint from such concepts, and deleting all concepts with score ≤ 0. Applying this to the initial formula, one gets a formula that is consistent, since it cannot contain disjoint concepts with AtomicScore 0, and also cannot contain disjoint concepts where one has positive AtomicScore. By the theorem above, this gives an optimal result.

EXAMPLE 4. *Returning to Example 1, an optimal score can be obtained by a repair set that deletes all 3 base annotations and inserts Location. This matches the intuition that the preponderance of the evidence points against each specific annotation, while supporting their common superclass Location.*

We return to the question of the settings for $\mathsf{SupportWeight}_{\mathsf{A},D}$ and $\mathsf{OmitWeight}_{\mathsf{A},D}$, beyond that given in the simple atomic scoring model. We have also experimented with a *bootstrapping approach* setting all the weights to be initially uniform at 1, and then iterating:

$$\mathsf{SupportWeight}^{k+1}_{\mathsf{A},D} = AVG_{\mathsf{Support}(\mathsf{A},D,s)}\ \mathsf{NormAtomicScore}^{k}(D)$$

$$\mathsf{OmitWeight}^{k+1}_{\mathsf{A},D} = AVG_{\mathsf{Omit}(\mathsf{A},D,s)}\ (1 - \mathsf{NormAtomicScore}^{k}(D))$$

where $\mathsf{Support}(\mathsf{A},D,s)$ indicates that s is a span which A tags with D, $\mathsf{Omit}(\mathsf{A},D,s)$ indicates that A has D in its vocabulary but omitted it on s, and $\mathsf{NormAtomicScore}^{k}(D)$ is a version of the $\mathsf{AtomicScore}(D)$ calculated based on $\mathsf{SupportWeight}^{k}$ and $\mathsf{OmitWeight}^{k}$, but normalized to be in $[0,1]$ (e.g. by shifting the scores and dividing by their maximum). The intuition is that if A has high precision for D, spans that it annotates with D should get high support in round k, and hence $\mathsf{NormAtomicScore}^{k}(D)$ should be be close to 1, making its SupportWeight high in round $k+1$. Similarly, if A has high recall for D, spans for which it omits D should get low support in round k, which will make $\mathsf{OmitWeight}^{k+1}$ high.

The set of documents over which the spans are calculated obviously will have a large impact on the results. We have tested the bootstrapping approach using the entire test corpus from Section 2. We have found that it works well for very general concepts, such as Person, Place, and Organization, where examples are plentiful, but hurts on a more specific (and less frequent) classes – increasing the Micro-average F-score but decreasing the Macro-average F-score (see Section 4 for the definitions of these).

Dealing with overlapping spans. The procedure described above assumes that annotations are related only if they have the same span, i.e., they insist on the same tokens of text. The repair is therefore computed one span at a time. On the other hand, annotations often have nested or overlapping spans, as in Figure 1. We therefore consider a more careful definition of annotation span that takes into account overlaps.

Overlaps: Let T be a document represented as a sequence of tokens $T = \langle t_1, \ldots, t_n \rangle$ and $\underline{\mathsf{a}}$ an annotation of a sequence $\langle t_i, \ldots, t_j \rangle$, $1 \leq i < j \leq n$, of tokens of T provided by an annotator AN_i using a label l_{ω_i} from its local vocabulary $\mathcal{L}_{\omega_i}$. We denote by $\mathsf{type}(\underline{\mathsf{a}})$ the label l_Ω from the vocabulary $\mathcal{L}_\Omega$ of the global ontology to which l_{ω_i} is mapped.

We also denote by $\mathsf{intv}(\underline{\mathsf{a}})$ the interval $[i, j)$ of tokens covered by $\underline{\mathsf{a}}$ in D. Given two annotations $\underline{\mathsf{a}}_1$, $\underline{\mathsf{a}}_2$ we say that they are *disjoint* iff $\mathsf{intv}(\underline{\mathsf{a}}_1) \cap \mathsf{intv}(\underline{\mathsf{a}}_2) = \emptyset$, they *overlap* iff $\mathsf{intv}(\underline{\mathsf{a}}_1) \cap \mathsf{intv}(\underline{\mathsf{a}}_2) \neq \emptyset$, in particular they *coincide* iff $\mathsf{intv}(\underline{\mathsf{a}}_1) = \mathsf{intv}(\underline{\mathsf{a}}_2)$. Moreover, we say that $\underline{\mathsf{a}}_1$ is *contained* in $\underline{\mathsf{a}}_2$ iff $\mathsf{intv}(\underline{\mathsf{a}}_1) \subseteq \mathsf{intv}(\underline{\mathsf{a}}_2)$.

Orthogonal spans. A way of dealing with overlapping spans is to consider distinct spans as mutually independent, i.e., annotations conflict or support one another only if their spans coincide. This obviously leads to a loss of precision in the aggregation, since overlapping spans do often refer to the same entity.

Coalesced spans. A completely opposite strategy is to take the connected component of spans under the overlap relation, considering each component as a span having all of the annotations associated with its component. We found that in practice these connected components are quite small – at most four tokens. But considering a disjoint pair of annotations connected in this way as being in conflict can lower the recall: e.g. in the case of nested annotations such as *"Sweden Central Bank"* where the whole string is annotated as Organization, e.g., by OpenCalais, and the string *"Sweden"* is annotated as Country, e.g., by AlchemyAPI.

Hybrid spans. To circumvent the above problem, we introduce a hybrid annotation span defined as follows. Given an ontology Ω, we define two annotations $\underline{\mathsf{a}}_1$, $\underline{\mathsf{a}}_2$ (with $\underline{\mathsf{a}}_1 \neq \underline{\mathsf{a}}_2$) as belonging to the same annotation span if either:

1. $\underline{\mathsf{a}}_1$ is contained into $\underline{\mathsf{a}}_2$ (resp. $\underline{\mathsf{a}}_2$ is contained into $\underline{\mathsf{a}}_1$) and it is not the case that $\Omega \models \mathsf{type}(\underline{\mathsf{a}}_1) \sqcap \mathsf{type}(\underline{\mathsf{a}}_2) \sqsubseteq \bot$.
2. $\underline{\mathsf{a}}_1$ overlaps with $\underline{\mathsf{a}}_2$ and neither $\underline{\mathsf{a}}_1$ nor $\underline{\mathsf{a}}_2$ is contained into the other.
3. $\underline{\mathsf{a}}_1$ coincides with $\underline{\mathsf{a}}_2$.

Condition 1 relates overlapping annotations that are possibly redundant. This is the case, e.g., when one annotation is contained in another with types in superclass relationship. Condition 2 captures the case of annotations with ambiguous spans, i.e., they overlap on one or more tokens but neither contains the other. Note that we purposely excluded those cases of (strictly) contained annotations with logically conflicting types.

Consider again the annotations in the example of Figure 1. According to the definition of hybrid spans, AN_1:Facility and AN_4:Facility belong to the same annotation span, due to Condition 1. The same applies to AN_2:City and AN_3:Location, since City $\sqsubseteq$ Location. On the other hand, AN_1:Person and AN_3:Location do not belong to the same annotation span, because $\Omega \models$ Person $\sqcap$ City $\sqsubseteq \bot$. The same applies to AN_1:City and AN_2:Facility. Therefore, for this example, the repair procedure will operate on AN_1:Facility, AN_4:Facility, AN_3:Location, and AN_2:City, leaving out AN_1:Person. Notice that, for the example of Figure 2, the repair procedure will operate on all annotations due to Condition 3.

Our experiments have shown that the use of hybrid spans increases the recall but lowers the precision w.r.t. coalesced spans. However, the change in performance is below 1% and the overall F-score remains practically unchanged ($\pm 0.1\%$) on all datasets. In the following we will stick to hybrid spans since containment among annotations is very frequent in practice.

Comparison with supervised aggregation. In Section 4 we benchmark our repair-based approach against several baselines, including all freely-available annotation aggregators. But a second natural comparison point is against what can be accomplished with supervision. Currently we know of only two approaches (i.e., FOX and NERD) to supervised aggregation of Web-based annotators. In FOX a single web annotator is considered together with standalone annotators and aggregated via neural networks, while in NERD the

details of the machine learning technique used for aggregation is not explicitly stated. We have thus implemented our own supervised approach to see to what extent one can leverage the existence of training data to find patterns that indicate (in)correctness of annotations. Can supervision be used not just to look for patterns in reliability of annotators (e.g. from tracking the recall or precision of annotators on concepts), but to detect and exploit more complex patterns in the behaviour of annotators?

We build on maximal entropy models, which have gained wide popularity for a variety of individual extraction tasks in recent years [8]. In our case, we wish to use a sequential model, which will allow us to reconcile span boundaries jointly with concept names, which also allows span boundary differences to serve as clues. Also, this model is general enough to detect a wide class of behavioural patterns within annotators. When non-sequential approaches have been applied to aggregation of annotations (e.g. the neural networks used in [1]) they have ignored span boundary issues. We thus use a *Maximal Entropy Markov Model* (MEMM) [29], which combines features of maximal entropy models with the sequential approach of a Markov model. A MEMM has a set of states, an input and output alphabet, and a probabilistic transition and emission functions that capture the system dynamics when a new input item is consumed. 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.

Applied to this setting, we let $\mathcal{AN} = \{\text{AN}_1, \text{AN}_2, \cdots\}$ be the collection of base annotators, $\mathsf{Classes}(\Omega)$ be the concept classes in ontology Ω, and D be a document represented as a sequence of tokens $D = \langle t_1, \ldots, t_n \rangle$. By adopting the standard begin/in/out (B-I-O) encoding scheme, let $L(\Omega) = \{B\_, I\_\} \times \mathsf{Con}(\mathsf{Classes}(\Omega))$, 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 the beginning of a snippet for an entity that is a member of each element of S, while (I,S) indicates that the position is inside of such a snippet. We let $\mathsf{States}(\Omega)$ be the power set of $L(\Omega)$. Elements of $\mathsf{States}(\Omega)$ can be mapped to a boolean combination in the obvious way: conjoining all the concepts in the list and dropping the set of decorations. We identify the empty subset as "*Other*" meaning that a token is out of any of the classes in Ω.

The input alphabet for observations about tokens, denoted O, consists of the *evidence vectors*, that correspond to a token t – the labelings of the different base annotators. We denote an evidence vector w.r.t. token t as $\vec{Ev}_t = \{l^t_{\text{AN}} | \text{AN} \in \mathcal{AN}\}$, where $l^t_{\text{AN}} \in \mathsf{States}(\Omega)$. Note that we do not assume that the individual annotator opinions are consistent. In our prototype, the output alphabet was identical to the state space, and we denote both by $\mathcal{S}$ below. We can easily extend to finer-grained state spaces (e.g. distinguishing between sections of an annotation).

As in any Markov model, the key problem is to define the transition probabilities: given two states l, l' along with an input symbol o, define the probability of transiting from l to l' 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 with their expectation over a sample distribution. Such a conditional probability must have the form:

$$P_l(l'|o) = \frac{1}{Z(l,o)} exp(\sum \lambda^l_i f_i(l', o))$$

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. $\lambda^l = \{\lambda_1, \lambda_2, \cdots, \lambda_m\}$ are the corresponding weight parameters learned in a training phase of source state l, and $Z(l,o)$ is a normalization factor for the weights, which is calculated using the formula:

$$Z(l,o) = \sum_{l' \in \mathcal{S}} exp(\sum \lambda^l_i f_i(l', o))$$

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*) [29] algorithm that iteratively updates the parameters based on expectations defined over their current values.

Thus the transition probability calculation is straightforward once the feature functions are defined and the corresponding weights are trained. Our most fundamental features are based on annotator opinions. In order to deal with *data sparsity* issues, we construct features based on *decomposed evidence* on 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 π_A is the projection on a set of annotators A. So that we can capture very fine-grained patterns, we allow these features to be specific to a particular piece of evidence and target state. For example, the following feature function 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*:

$$f_1(l', o) = \begin{cases} 1, \text{ if } l_{\mathsf{OpenCalais}} = B\_Facility, l_{\mathsf{AlchemyAPI}} = B\_Person \\ \quad \text{and } l' = B\_Person \text{ ;} \\ 0, \text{ otherwise.} \end{cases}$$

We train separate Maximum Entropy models via the *GIS* algorithm for each source state l with its corresponding training set $\mathcal{T}^l$. $\mathcal{T}^l$ consists of token events, of which the previous token label is l, as $\mathcal{T}^l = \{\langle \pi_{\text{AN}_i}(\vec{Ev}^{t_k}), \pi_{\text{AN}_i,\text{AN}_j}(\vec{Ev}^{t_k}), label_{t_k} \rangle | label_{t_{k-1}} = l$ and $\text{AN}_i, \text{AN}_j \in \mathcal{AN}$ and $\text{AN}_i \neq \text{AN}_j\}$.

We then process the input evidence sequence by constructing a *Markov sequence*, after which we decode using the standard Viterbi algorithm, calculating the most likely annotation sequence according to the model and returning the output annotations. More details on the implementation of MEMM are deferred to [6].

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. The model will enforce that an output annotation is always well-formed (e.g. that is does not begin a new span before closing an old one), and thus that the token-string produced by MEMM will correspond to an annotation of the document, which can be returned as the final output.

We utilize the library OpenNLP-MAXENT-3.0.0 as the core solution for the Maximum Entropy framework. This is a mature JAVA package for training and running maximum entropy models. The toolkit OpenNLP Tools [4] is used to deal with common NLP tasks such as tokenization and sentence segmentation.

As the performance of predicting the most likely annotation sequence is highly sensitive to the state space of the MEMM, we perform some post-processing on the Markov sequence resulting from the product of the Maximum Entropy model with the input sequence, removing transitions that are (heuristically) unlikely to be relevant. For example, 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.

We use a map to index only the transitions with nonzero probability, taking advantage of the sparsity of the transition matrix. We further exploit the fact that our Markov model does not consider interactions across sentences; thus we can run Viterbi in parallel over each sentence.

4. EXPERIMENTS

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 state-of-the art aggregators such as FOX [1] and NERD [32].

Individual vs aggregated. We evaluated our aggregation methods using the same datasets and experimental setting which have been described in Section 2. 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 5 reports the 10-fold cross-validation performance 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. We give averages of the precision, recall, and F-score, measured in the ontology-aware way given in Section 2. We provide: a *macro-average* that first averages 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 testset).

Our first observation is that all three 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 NETagger and Reuters corpus respectively. However, it is worth noting that their vocabularies represent only 18% of all concepts in the gold standard. On average, 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 5% 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); 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 its resilience to sparsity in the dataset. We see from the example of PhoneNumber discussed above that MEMM performs quite poorly when training data is sparse, while WR has score comparable with individual annotators.

Comparative evaluation. We compared the performance of our aggregation methods against the FOX [1] and NERD [32] NER aggregators (discussed in Section 5) on all datasets.

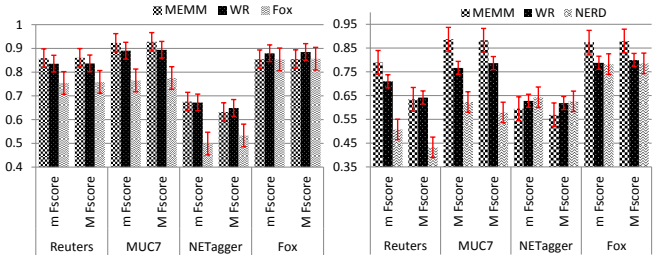
Although our focus is on Web-based annotators, the comparative evaluation required the use of non web annotators used by FOX, namely the STANFORDNER and Illinois NETAGGER, as well as other web-based annotators such as SAPLO, WIKIMETA, LUPEDIA, and YQL. Individual results about these annotators have not been included in this paper due to space reasons but can be examined online at [6]. Figure 4 summarizes the results and clearly shows 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, NETagger, and Reuters datasets. An interesting case is the NETagger corpus that suffers from high data sparsity. In fact, not only does NERD show a better behavior, but also MEMM is often outperformed by WR on this dataset. On the other hand, the difference in performance is less evident on the Fox corpus where FOX and MEMM behave very similarly, but they are often outperformed by WR due again to data-sparsity issues in the dataset.

The full suite of test results, together with all the text resources and annotated gold standards is available online at [6].

Performance and Scalability. The last set of experiments studies the amount of resources required to compute a solution for the

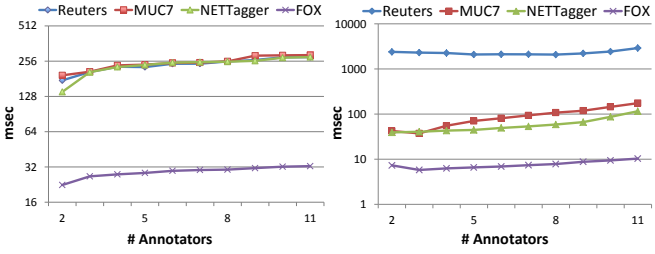
Table 5: Individual annotators vs aggregators.

	MUC7						NETagger						Web						Reuters					
	micro			macro			micro			macro			micro			macro			micro			macro		
	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1
ALCHEMY	.86	.45	.59	.76	.44	.55	**.78**	.60	**.67**	.73	.57	**.64**	**.90**	.81	.85	**.89**	.81	.85	.80	.59	.68	.62	.44	.51
Baseline	.44	.84	.57	.55	.84	.66	.40	**.79**	.53	.41	**.75**	.53	.35	**.90**	.50	.36	**.91**	.51	.43	**.89**	.58	.43	**.75**	.54
WR	.73	.84	.78	.77	.83	.80	.51	.78	.61	.53	.72	.60	.71	.88	.79	.72	.89	.79	.68	.82	.75	.61	.64	.63
MEMM	**.95**	**.88**	**.92**	**.95**	**.87**	**.91**	.77	.55	.64	**.74**	.52	.60	.89	.88	**.88**	.89	.88	**.88**	**.83**	.79	**.81**	**.70**	.60	**.64**
SPOTLIGHT	.69	.37	.49	.70	.36	.48	.30	.15	.19	.30	.17	.20	.77	.38	.50	.77	.38	.50	.66	.45	.53	.45	.32	.37
Baseline	.40	.91	.56	.43	.91	.59	.40	**.79**	.53	.41	**.75**	.53	.35	**.90**	.50	.36	**.91**	.51	.36	**.88**	.52	.38	**.66**	.48
WR	.70	.91	.79	.73	.91	.81	.51	.78	.61	.53	.72	**.60**	.71	.88	.79	.72	.89	.79	.64	.83	.72	.54	.56	.55
MEMM	**.96**	**.94**	**.95**	**.96**	**.94**	**.95**	**.77**	.55	**.64**	**.74**	.52	.60	**.89**	.88	**.88**	**.89**	.88	**.88**	**.84**	.78	**.81**	**.61**	.52	**.56**
EXTRACTIV	.87	.68	.76	.92	.64	.75	.56	.35	.40	.50	.35	.41	.73	.79	.76	.74	.78	.76	.71	.69	.70	.65	.62	.63
Baseline	.45	.82	.58	.73	.75	.74	.40	**.79**	.53	.41	**.75**	.53	.35	**.90**	.50	.36	**.92**	.51	.52	**.81**	.63	.62	**.72**	.67
WR	.73	.82	.77	.86	.75	.80	.51	.78	.61	.53	.72	**.60**	.71	.88	.79	.72	.89	.79	.72	.76	.74	.69	.65	**.67**
MEMM	**.95**	**.85**	**.90**	**.94**	**.76**	**.85**	**.77**	.55	**.64**	**.74**	.52	.60	**.89**	.88	**.88**	**.89**	.88	**.88**	**.83**	.79	**.81**	**.70**	.63	.66
OPENCALAIS	.91	.61	.73	.85	.65	.73	.73	.51	.60	.60	.45	.51	.80	.82	.81	.79	.82	.80	.80	.72	.76	**.69**	.59	**.64**
Baseline	.45	.85	.58	.62	.85	.72	.40	**.79**	.53	.41	**.75**	.53	.35	**.90**	.50	.36	**.91**	.51	.49	**.86**	.62	.48	**.77**	.59
WR	.73	.84	.78	.81	.85	.83	.51	.78	.61	.53	.72	**.60**	.71	.88	.79	72	.89	.80	.66	.79	.72	.60	.67	.63
MEMM	**.95**	**.88**	**.92**	**.95**	**.88**	**.92**	**.77**	.55	**.64**	**.74**	.52	.60	**.89**	.88	**.88**	**.89**	.88	**.88**	**.80**	.72	**.76**	.69	.55	.61
ZEMANTA	.72	.20	.31	.76	.18	.29	.47	.14	.20	.37	.15	.19	.71	.64	.67	.69	.63	.65	.69	.29	.41	.51	.21	.29
Baseline	.41	.91	.56	.43	.91	.59	.40	**.79**	.53	.41	**.75**	.53	.35	**.90**	.50	.36	**.91**	.51	.42	**.87**	.56	.40	**.60**	.48
WR	.70	.91	.79	.73	.91	.81	.51	.78	.61	.53	.72	**.60**	.71	.88	.78	.72	.89	.79	.67	.80	.73	.53	.53	.49
MEMM	**.96**	**.93**	**.95**	**.96**	**.94**	**.95**	**.77**	.55	**.64**	**.74**	.51	.60	**.89**	.88	**.88**	**.89**	.88	**.88**	**.83**	.76	**.79**	**.59**	.49	**.54**



(a) vs FOX (b) vs NERD

Figure 4: Comparative Evaluation.



(a) WR (b) MEMM

Figure 5: Aggregation Performance.

integration. Figure 5 plots 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. WR is substantially more performant than MEMM. 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, and that is considerably higher in the MUC7, NETagger 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 1-2 seconds while on the other corpora, when fewer concepts are involved, is much lower and MEMM's performance is comparable with WR.

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 both 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 about 60 seconds to annotate a medium-sized (i.e., 3-4 Kb) document. Also, both MEMM and WR have negligible memory consumption when compared with the size of the raw annotations.

Summary. The finding of our experimental evaluations can be summarized as follows:

1. Aggregating opinions of individual annotators is always beneficial in terms of recall. In terms of overall accuracy, aggregation shows consistent benefit except few cases where the majority of annotators contribute wrong annotations.
2. Although supervised aggregation delivers overall higher accuracy, unsupervised aggregation is to be preferred when no or sparse training data is available.
3. Both our aggregation techniques consistently outperform existing annotation aggregation approaches.
4. Aggregation does not impact the scalability of the performance of annotation tasks, since it is always dominated by the annotation time of the individual annotators.

5. RELATED WORK

Our work 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.

Named Entity Recognition. Research on Named Entity Recognition (NER) systems can be currently grouped into three strands, according to the sources of information that are leveraged in order to produce textual annotations. *Language-driven* approaches (e.g. KnowItAll [17]) represent the tradition in NER and exploit knowledge about the natural language to derive linguistic patterns – typically in the form of rules – that, once applied to the text, extract instances of the entities of interest. *Knowledge-driven* systems (e.g. LearningPinocchio [13], PANKOW [12], KIM [27]) use unstructured, i.e.,

gazetted, or structured, i.e., ontological, background knowledge to locate the entities of interest in a text document.

Combining annotators. Many previous approaches addressing the integration of multiple NER systems have been based on variations of voting mechanisms [28]. As also noticed in [35], 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 [35] the weights are determined using an exponential model borrowed from meta information retrieval systems [7]. 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. Our bootstrapping procedure described in Section 3 also works at a fine granularity, producing weights for each annotator-concept pair, ensuring that these peculiarities are considered in the computation. Biasing is also often complemented with thresholding [26], 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 [1], support vector machines [16], classifier *stacking* [38, 39, 20], and conditional random fields (CRFs) [35]. 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 [32] – 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 [37], 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 wide range of variants for both CRFs and Maximal Entropy Markov models (e.g. in the objective function used in training [25]), and so the trade-off between them, even in the absence of semantic considerations, is quite complex. In Section 4, we compared our approach experimentally to the main learning-based approach which was freely available, FOX [1]. We could not find an implementation of [20], which won a CoNLL competition, but we note that it performed experiments on a Reuters corpus similar to the one we tested on, reporting F-scores that are very similar to the macro- and micro- F-scores 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. [30] 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. Unlike either of our approaches, semantic relationships within the vocabulary of annotations are not considered. Our unsupervised method is close in spirit to classic judgement aggregation techniques [24], 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.

Relation to other areas. 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 [15, 21] and produce a *truth-finding* model. For example, iterative algorithms within the PageRank/Hits family are applied in Galland et. al. [21] to determine trustworthiness. Here trust/reliability/dependence is naturally attributed globally to a web source, 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 [36] 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 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 discovering them.

Our repair-based aggregation is close in spirit to consistent reasoning for inconsistent ontologies [33]. However, such techniques cannot be straightforwardly applied to our setting where inconsistent assertions can be stated multiple times and the logical consistency also relates to the way annotations are contained into each other. We consider scoring of repairs due to the support of the facts they remove. Scoring of repairs also arises for numerical integrity constraints [9, 19]. Unlike in our case repairs can only have non-negative scores, and this impacts the technical development of the solution dramatically.

6. CONCLUSIONS & FUTURE WORK

We have presented a set of approaches for knowledge-aware integration of entity extractors that consider each entity extractor as a black box and do not assume any prior knowledge about their performance or competence w.r.t. a certain set of entities.

These approaches have been proposed as alternatives, depending on the application scenario. But of course it will often occur that one approach or the other is more appropriate on a per-concept basis. Thus we are currently investigating how to combine these methods. We currently produce an aggregate annotation 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 querying Markov sequences, tuned to the case of annotations. We are also preparing a study of the adaptation of other supervised learning methods, such as conditional random fields, 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 [11] 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. Our methods can be extended to do entity 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 object. Our focus on concept annotation allowed 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.

7. REFERENCES

[1] FOX. http://ontowiki.net/Projects/FOX?v=4e5.

[2] LingPipe. http://alias-i.com/lingpipe/.

[3] MUC7. http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2001T02.

[4] OpenNLP Tools. http://opennlp.apache.org/index.html.

[5] Reuters. http://about.reuters.com/researchandstandards/corpus/index.asp.

[6] ROSeAnn. http://diadem.cs.ox.ac.uk/roseann.

[7] J. A. Aslam and M. Montague. Models for metasearch. In *SIGIR*, pages 276–284, 2001.

[8] O. Bender, F. J. Och, and H. Ney. Maximum entropy models for named entity recognition. In *CoNLL*, pages 148–151, 2003.

[9] L. Bertossi, L. Bravo, E. Franconi, and A. Lopatenko. The complexity and approximation of fixing numerical attributes in databases under integrity constraints. *Inf. Sys.*, 33:407–434, 2008.

[10] R. Carreira, S. Carneiro, R. Pereira, M. Rocha, I. Rocha, E. Ferreira, and A. Lourenço. Semantic annotation of biological concepts interplaying microbial cellular responses. *BMC Bioinformatics*, 12:1–10, 2011.

[11] L. Chen, S. Ortona, G. Orsi, and M. Benedikt. ROSeAnn: Reconciling opinions of semantic annotators. *PVLDB*, To appear 2013.

[12] P. Cimiano, S. Handschuh, and S. Staab. Towards the self-annotating web. In *WWW*, pages 462–471, 2004.

[13] F. Ciravegna and A. Lavelli. LearningPinocchio: Adaptive information extraction for real world applications. *Nat. Lang. Eng.*, 10(2):145–165, 2004.

[14] N. Dalvi, R. Kumar, and M. Soliman. Automatic wrappers for large scale web extraction. *PVLDB*, 4(4):219–230, 2011.

[15] X. L. Dong, L. Berti-Equille, and D. Srivastava. Integrating conflicting data: the role of source dependence. *PVLDB*, 2(1):550–561, 2009.

[16] D. Duong, J. Venuto, B. Goertzel, R. Richardson, S. Bohner, and E. Fox. Support vector machines to weight voters in a voting system of entity extractors. In *IJCNN*, pages 1226–1230, 2006.

[17] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Web-scale information extraction in KnowItAll. In *WWW*, pages 100–110, 2004.

[18] J. Euzenat. Semantic precision and recall for ontology alignment evaluation. In *IJCAI*, pages 348–353, 2007.

[19] S. Flesca, F. Furfaro, and F. Parisi. Querying and repairing inconsistent numerical databases. *ACM Trans. Database Syst.*, 35(2):14:1–14:50, May 2010.

[20] R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. Named entity recognition through classifier combination. In *CoNLL*, pages 168–171, 2003.

[21] A. Galland, S. Abiteboul, A. Marian, and P. Senellart. Corroborating information from disagreeing views. In *WSDM*, pages 131–140, 2010.

[22] D. Grossi and G. Pigozzi. Introduction to judgment aggregation. In *Lectures on Logic and Computation*, pages 160–209. 2012.

[23] H. Cunningham et. al. *Text Processing with GATE (Version 6)*. U. Sheffield Dept. of CS, 2011.

[24] S. Hartmann, G. Pigozzi, and J. Sprenger. Reliable methods of judgement aggregation. *J. Log. Comp.*, 20(2):603–617, 2010.

[25] S. Kakade, Y. W. Teh, and S. T. Roweis. An alternate objective function for markovian fields. In *ICML*, pages 275–282, 2002.

[26] N. Kambhatla. Minority vote: at-least-n voting improves recall for extracting relations. In *COLING*, pages 460–466, 2006.

[27] A. Kiryakov, B. Popov, I. Terziev, D. Manov, and D. Ognyanoff. Semantic annotation, indexing, and retrieval. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2(1):49–79, 2004.

[28] Z. Kozareva, O. Ferrández, A. Montoyo, R. Muñoz, A. Suárez, and J. Gómez. Combining data-driven systems for improving named entity recognition. *Data Knowl. Eng.*, 61(3):449–466, 2007.

[29] A. McCallum, D. Freitag, and F. C. Pereira. Maximum entropy markov models for information extraction and segmentation. In *ICML*, pages 591–598, 2000.

[30] E. Michelakis, R. Krishnamurthy, P. J. Haas, and S. Vaithyanathan. Uncertainty management in rule-based information extraction systems. In *SIGMOD*, pages 101–114, 2009.

[31] L. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. In *CoNLL*, pages 147–155, 2009.

[32] G. Rizzo and R. Troncy. Nerd: A framework for unifying named entity recognition and disambiguation extraction tools. In *EACL*, pages 73–76, 2012.

[33] R. Rosati. On the complexity of dealing with inconsistency in description logic ontologies. In *IJCAI*, pages 1057–1062, 2011.

[34] P. Senellart, A. Mittal, D. Muschick, R. Gilleron, and M. Tommasi. Automatic wrapper induction from hidden-web sources with domain knowledge. In *WIDM*, pages 9–16, 2008.

[35] L. Si, T. Kanungo, and X. Huang. Boosting performance of bio-entity recognition by combining results from multiple systems. In *BIOKDD*, pages 76–83, 2005.

[36] F. M. Suchanek, M. Sozio, and G. Weikum. SOFIE: A self-organizing framework for information extraction. In *WWW*, pages 631–640, 2009.

[37] M. van Erp, G. Rizzo, and R. Troncy. Learning with the Web: Spotting named entities on the intersection of NERD and machine learning. In *MSM*, 2013.

[38] H. Wang and T. Zhao. Identifying named entities in biomedical text based on stacked generalization. In *WCICA*, pages 160–164, 2008.

[39] D. Wu, G. Ngai, and M. Carpuat. A stacked, voted, stacked model for named entity recognition. In *CoNLL*, pages 200–203, 2003.