International Journal of Semantic Computing
© World Scientific Publishing Company

SEMANTIC DISAMBIGUATION AND LINKING OF QUANTITATIVE MENTIONS IN TEXTUAL CONTENT

MEHRNAZ GHASHGHAEI
University of New Brunswick,Canada

EBRAHIM BAGHERI
Laboratory for Systems, Software and Semantics (LS^3), Ryerson University,Canada

JOHN CUZZOLA
Laboratory for Systems, Software and Semantics (LS^3), Ryerson University,Canada

ALI A. GHORBANI
University of New Brunswick,Canada

ZEINAB NOORIAN
Laboratory for Systems, Software and Semantics (LS^3), Ryerson University,Canada

Semantic annotation techniques provide the basis for linking textual content with concepts in well grounded knowledge bases. In spite of their many application areas, current semantic annotation systems have some limitations. One of the prominent limitations of such systems is that none of the existing semantic annotator systems are able to identify and disambiguate quantitative (numerical) content. In textual documents such as Web pages, specially technical contents, there are many quantitative information such as product specifications that need to be semantically qualified. In this paper, we propose an approach for annotating quantitative values in short textual content. In our approach, we identify numeric values in the text and link them to an existing property in a knowledge base. Based on this mapping, we are then able to find the concept that the property is associated with, whereby identifying both the concept and the specific property of that concept that the numeric value belongs to. Results obtained from the developed gold standard dataset show that the proposed automated semantic annotation platform is quite effective in detecting and disambiguating numerical content, and connecting them to associated properties on the external knowledge base. Our experiments show that our proposed approach is able to reach an accuracy of over 70% for semantically annotating quantitative content.

Keywords: semantic web, automated annotation systems; qualitative content.

1. Introduction

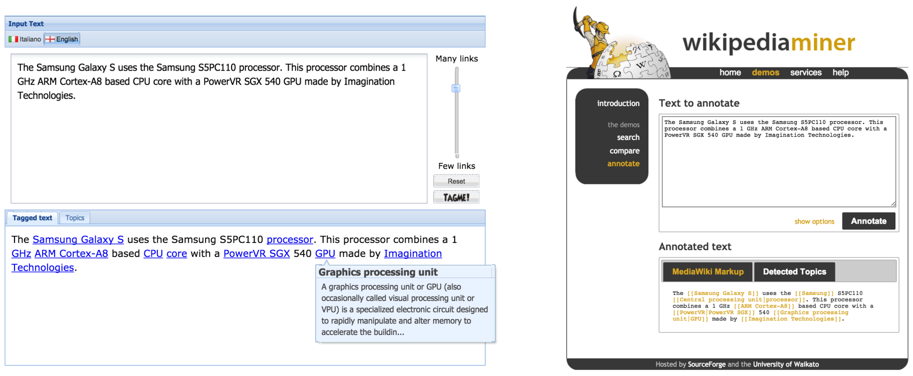
As more and more content is being disseminated on online platforms such as blogs, social media and microblogs, the need for better and more efficient techniques for organizing, searching and efficiently retrieving information is required. Techniques that benefit from well-grounded knowledge bases such as ontologies for the sake of information organization and retrieval have received attention in the recent years [1], which include open information extraction [2], ontology population and enrichment [3], and semantic tagging and annotation [4, 5], just to name a few. These techniques aim to identify and extract structured information from unstructured content. Automated semantic annotation systems are among such systems that enable the identification and labeling of instances of knowledge base concepts within text, whereby enriching textual documents with additional semantic information linked to external knowledge bases.

With the emergence of the linked open data initiative, many semantic annotator systems now benefit from the knowledge bases that are shared through this platform to spot, disambiguate and link semantic information within textual content [6]. Knowledge bases such as Freebase [7] and DBpedia [8] that sit at the core of the linked open data cloud have been used extensively for this purpose where their concepts are employed for semantically grounding textual content. Semantic annotator systems typically provide support for entity linking [9], suggestion of related but unobserved concepts, role assignment and detection of relevant semantic categories.

In spite of the growing adoption of semantic annotator systems, one of the major limitations that current annotators face concerns dealing with quantitative (numerical) textual content. In other words, none of the existing semantic annotator systems is able to semantically link or describe numerical content. Therefore, valuable information that are expressed in the form of numbers are largely ignored in the current semantic annotator systems; hence, they are neither exploited in the annotation process nor are they semantically linked for future use. Let us consider a sample short text describing a Samsung Galaxy S smart phone: “The Samsung Galaxy S uses the Samsung S5PC110 processor. This processor combines a 1 GHz ARM Cortex-A8 based CPU core with a PowerVR SGX 540 GPU made by Imagination Technologies.”. When processed by a state of the art semantic annotator system such as TagMe [10], the phrases ‘Samsung Galaxy S’, ‘ARM Cortex-A8’, ‘processor’, ‘Imagination Technologies’ and ‘CPU’ are detected and linked to their corresponding Wikipedia entities. However, none of the numerical values are detected for semantic annotation. This limitation prevents the correct interpretation of quantitative values within text, which can constitute a noticeable portion of text, e.g. see product specification Web pages.

Figure 1 show two state of the art semantic annotator systems: TagMe [10] and WikipediaMiner [11]. The figure illustrates the results of these systems for the above example sentence about “Samsung Galaxy S”. They both identify and link all concepts mentioned in the text. However, they are not able to infer the semantics of the mentioned numbers.

In this paper, we propose an approach for annotating quantitative values in a short text. In our work, we identify numeric values in text and not only link them to the most relevant property in the knowledge base but also find the best



The Samsung Galaxy S uses the Samsung S5PC110 processor. This processor combines a 1 GHz ARM Cortex-A8 based CPU core with a PowerVR SGX 540 GPU made by Imagination Technologies.

The Samsung Galaxy S uses the Samsung S5PC110 processor. This processor combines a 1 GHz ARM Cortex-A8 based CPU core with a PowerVR SGX 540 GPU made by Imagination Technologies.

Graphics processing unit
A graphics processing unit or GPU (also occasionally called visual processing unit or VPU) is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the buildin...

```
The Samsung Galaxy S uses the Samsung S5PC110 processor. This
processor combines a 1 GHz ARM Cortex-A8 based CPU core with a
PowerVR SGX 540 GPU made by Imagination Technologies.
```

```
The [[Samsung Galaxy S]] uses the [[Samsung]] S5PC110
[[Central processing unit|processor]]. This processor
combines a 1 GHz [[ARM Cortex-A8]] based CPU core with a
[[PowerVR|PowerVR SGX]] 540 [[Graphics processing
unit|GPU]] made by [[Imagination Technologies]].
```

Figure 1: TagMe semantic annotator (on the left), and WikipediaMiner semantic annotator (on the right). They identify and link concepts mentioned in the text, but they ignore all numerical content.

matching concept[a] that has the identified property. Therefore, our method enables the specification of a numeric value within the context of a concept by relating it with one of the properties of that concept. For instance, in the above example, our method is able to determine that 1 GHz is the value of the *frequency* property of the *ARM Cortex-A8* concept.

Our work in this paper is primarily based on semantic annotation systems focused on addressing one of the major limitations of such systems: semantically annotating numerical content. We propose a two-step semantic annotation approach targeting quantitative content. Our main contributions can be enumerated as follows:

(1) We propose a property identification method that is able to detect properties from the knowledge base that can accurately describe quantitative content in the text.
(2) We further introduce a concept identification mechanism that is able to find the most relevant concept mentioned in the text which has the same properties as the ones identified in the first step.
(3) We build and introduce a gold standard dataset that includes sentences with quantitative values that can be used for evaluation purposes. By choosing different concept-property pairs from various domains, we collected descriptive sentences considering desired concept-property pairs. We then processed collected gold standard content with TagMe [10] semantic annotator, and stored the extracted concept in the gold standard dataset.

For evaluation purposes, we exploit a gold standard dataset consisting of short

[a]Also known as *entity*.

textual snippets that have at least one numerical value. We compare the obtained property and concept for each numerical value and compare them with the gold standard.The experimental results on the gold standard indicate an accuracy of 73% for predicting the correct property, and 72% for identifying the correct concept. The results further show that the performance of the concept identification method improves and reaches an accuracy of 87% if the property detection method would be able to correctly identify the relevant properties.

The rest of this paper is organized as follows. In Section 2, we review the background on different types of semantic annotation of textual content from manual annotators to fully automated ones. Section 3 is a detailed description of our proposed approach including the procedure for identifying the relevant entities and corresponding properties. We provide a running example to articulate each step in a detailed manner. The evaluation procedure, dataset and results are provided in Section 4 and finally Section 5 concludes the paper.

2. Related Work

The main purpose of Semantic Web [12] is to add formal structure and metadata to the web content for more efficient management and access [13]. Semantic annotation can be viewed as a task of assigning semantic description to entities. In this process, each entity in a given text is mapped to its corresponding entity in a knowledge base that properly reflects its semantics. When the entities are mapped to the knowledge base, their semantic description such as their type and their relations with other entities can easily be retrieved. Given different levels of human participation in the annotation process, semantic annotation tools are classified into three categories: manual, automated, and semi-automated [14].

In the following, we review the aspects of different types of annotation process and discuss their pros and cons in a detailed manner.

2.1. *Manual Annotation Tools*

In manual annotation, human users play the primary role in curating or providing the annotations through some form of interface. Having a predefined ontology, users annotate the web documents manually and produce annotated documents. Crawlers can be further used to generate a knowledge base (KB) based on the annotated documents.

In [15], Erdmann and et al. enhanced HTML pages with semantically relevant extensions to add semantics to the pages. Therefore, the annotation process is done by extending HTML anchor tags with spacial attributes. The resulting language is called HTML-A. Using this method, semantically meaningful metadata is added to web pages without changing the content of the documents. The semantic tags to HTML pages is embedded in such a way that browsers can still process them and at the same time crawlers e.g. Ontobroker [16]) can recognize the embedded tags in the pages.

In [17], the authors designed a manual annotation process dedicated to annotating specific websites about Transmissible Spongiform Encephaiopathy (TSE)[b]. In so doing, they extended HTML to a new knowledge representation language called Simple HTML Ontology Extensions (SHOE). Different from [15], the authors developed a tool to automatically convert user input to SHOE [17] syntax, helping knowledge providers to easily annotate web pages without much understanding on SHOE.

The main advantage of manual annotation is that human annotation is very fine-grained. However, manual annotation is an error-prone process as it highly depends on humans' knowledge in the domain, extensive training, personal motivation, and the sophistication of the ontologies used for annotation. Besides, manual annotation is an expensive process as it relies on human labor; thus it is not an efficient solution to annotate huge numbers of Web documents.

2.2. *Semi-automated Annotation Tools*

To overcome the shortcomings of manual annotation tools, semi-automated annotation was proposed. Semi-automated annotation platforms are primarily divided into two categories: *Pattern-based* approaches, and *Machine Learning-based* approaches [18]. Although this is the primarily categorization, there exist platforms that combines these approaches to benefit from the functionality of both approaches while curtailing their weaknesses.

Pattern-based platforms may either include pattern discovery mechanism or have their patterns manually defined. First, an initial set of entities are identified in the document. Then the initial entries are used to find rules based on the content of the entries. Discovering new rules leads to discovering new entities. This process repeats with an expanded set of entities recursively until no more entities can be discovered or the user stops it. From among the semi-automated annotation tools that utilizes a pattern-based method is Armadillo [19].This platform has a very generic architecture, which can be employed in various domains. The system starts with finding out where to search. Once it identifies the initial data, the data is passed to an oracle (such as a human or an IE system) for validation, which determines whether an entity is an instance of a concept in an ontology. This process continues until no more information can be discovered or the user decides to stop the process. A use-case implementation of Armadillo is the automated identification and extraction of information from Web pages such as the extraction of the list of staff from a Computer Science department webpage. In this application, the goal is to discover computer science department staff's names and their related information such as publications, homepage, and other personal data. First, they simply use a Named Entity Recognition (NER) system to find a potential set of names. Then

[b]TSE is a brain disease that causes abnormalities in the brain

Citeseer[c] and Unitrier[d] are used to determine whether the string represents a name or not (validation). At this step, there still might be some entities that are not desirable e.g. names of people that co-authored papers with people working in the department. In order to exclude those entities, personal web pages are consulted. This processes gives the system a reliable set of seed names. Afterwards, the system learns new names that are organized in HTML structure e.g. table or list. The repetitive process stops if no more names are found or the user stops it.

Machine learning-based platforms employ probability and induction techniques in their annotation process. They use probability such as statistical models to predict the location of entities in documents, and induction to link the entities to concepts. One of the well-known applications of semi-automated annotation systems that uses machine learning methods is MnM [20]). MnM is a generic platform, which can be applied to any domain. There is basically five steps in this platform: 1) One of the features supported by MnM is ontology browsing, which enables users to choose a specific ontology model based on the application domain from a list of existing models on an ontology server. 2) the ontology is used to manually markup a training corpus. This process is exactly similar to what needs to be done with manual annotation tools. The User annotates a set of documents and then MnM inserts the appropriate SGML/XML tags into the document. 3) A machine learning algorithm is used to learn annotation rules from the marked up corpus. The authors have already integrated MnM with an IE tool called Amilcare that enriches the documents with XML tags. To annotate documents in a new domain Amilcare should be trained with a set of training documents. 4) Using a test corpus, the precision and recall of the trained rules is assessed. 5) If the annotation model meets the standards, it is used for marking up documents. At this point Amilcare has already induced the annotation rules. The system first receives a set of documents to annotate. These documents are then ran through Annie[e] for preprocessing. Then for annotation the system uses the Gate annotation schema[f].

As mentioned before, MnM integrates web-based ontology editing with semi-automated semantic markup. However, one of the main drawbacks with this system is that the annotation process is very time consuming. As an example, the authors spent over five years to annotate their web based newsletter.

Table 1 presents other semi-automated annotation systems along with their annotation methods. As indicated, most semi-automated systems use pattern-based approach with matching rules. PANKOW [21] is a pattern-based annotator, which also supports pattern discovery.

In semi-automated annotation, the system relies on human intervention at some point in the annotation process. Semi-automatic annotation is considered to be a

[c]`www.citeseer.com`
[d]`http://www.informatik.uni-trier.de/ley/db/`
[e]Shallow IE system in `www.gate.ac.uk`
[f]`www.gate.ac.uk`

Table 1: Semi-automated annotators and their annotating method.

Annotation system	**annotation method**
AeroDAML [22]	Pattern-based (rules matching)
KIM [23]	Pattern-based (rules matching)
MUSE [24]	Pattern-based (rules matching)
Ont-O-Mat: Amilcare [25]	Machine learning-based
Ont-O-Mat: PANKOW [21]	Pattern-based (Pattern discovery)
SemTag [26]	Pattern-based (rules matching)

milestone in transitioning to automatic annotation.

Since the contribution of this paper is proposing an automated annotation tool for quantitative data, we focus on this category and discuss it in more details in the following section.

2.3. *Automated Annotation Tools*

One of the open areas of knowledge extraction from natural language is semantic annotation. In this section, for the sake of brevity, we refer to semantic annotation tools as annotators. Existing annotators can basically support five different tasks: entity linking, identifying document topic, suggestion of related topics, and role assignment. Table 2 depicts some of the existing annotators and their supported tasks.

Automatic annotation is basically the task of extraction and disambiguation of mentioned entities in a given text. Annotators typically operate based on three main phases: detection of concept candidates, disambiguation, and pruning of results [4], which we briefly review in the following.

2.3.1. *Detection*

In the first phase, the annotator processes the given input text and picks out specific phrases from the text, called “mentions”, that can potentially refer to an existing concept with the source knowledge base. For each of the mentions, a set of candidate concepts are selected that are associated with that mention. Detection of mentions is also known as “spotting”. TagMe [10] has an Anchor Dictionary for this phase, and detects mentions by querying this dictionary. DBpedia Spotlight [28] also relies on a dictionary for spotting. In DBpedia Spotlight, a lexicon that associates multiple surface forms to a concept is used. Wikipedia Miner [11] uses pure text processing to find the spots and their candidates. It gathers all n-grams within text but only

Table 2: Some annotators and their supported tasks. The tasks that support relevance score are specified with RS in parenthesis.

Annotation tool	**Supported tasks**
TagMe [10, 27]	entity linking (RS)
DBpedia Spotlight [28]	entity linking (RS)
Wikipedia miner [11]	entity linking (RS), document topic
Alchemy API [g]	document topic (RS), entity linking (RS)
Open Calais [h]	document topic, entity linking (RS)
Denote [29]	entity linking (RS), suggestion of related topics (RS), role assignment
LUpedia [30]	entity linking (RS)

keeps those that have a high probability of linking in order to discard irrelevant phrases and stop words. In AIDA [31], a Named Entity Recognition (NER) tool is used. This NER tool identifies noun phrases that potentially denote named entities. Then YAGO2 is used to associate a candidate set to each potential named entity. In Illinois Wikifier [32] the authors perform pure text processing for entity spotting. They utilize an anchor-title index, computed by crawling Wikipedia, that maps each distinct hyperlink anchor text to its target Wikipedia titles. Since checking all substrings in the input text against the index is computationally inefficient, they only consider the expressions marked as named entities by a NER tagger, the noun-phrase chunks extracted by a publicly available shallow parser, and all sub-expressions of up to 5 tokens of the noun-phrase chunks. Then, for each mention, Wikipedia titles that are mapped to the mention (anchor text) are considered to be the candidate entities.

In our work, the detection phase starts with finding the numeric values in the input text. Assuming that we have the disambiguated mentions in the text, a set of candidate concepts are extracted. These concepts have the potential of having the most relevant property for the numeric value. Then from all properties of candidate concepts, a set of candidate properties are selected and associated with the spotted numeric value.

2.3.2. *Disambiguation*

Within the detection phase, a set of candidate concepts are identified. The objective of the disambiguation phase is then to select the concepts that most accurately highlight each mention's semantics, from among the concepts identified in the previous phase. There are generally four groups of work that perform disambiguation in annotators [4], namely popularity based, context based, collective disambiguation and graph-based techniques.

In the *popularity-based* approach the most frequently observed concept for a given mention is chosen. This method is usually combined with other approaches, since merely using this approach can lead to erroneous results. The reason is that the results do not consider context in which the mention appears and therefore largely ignore the main theme of the text. TagMe, Wikipedia Miner, AIDA, and Illinois Wikifier use the popularity-based approach combined with one of the following approaches for disambiguation.

Within the *context-based* approach, the context of the mention and the context of candidate concepts are compared. Context is typically modeled through bag-of-words and different distance measures [4]. Context-based approaches are used in DBpedia Spotlight, AIDA, and Illinois Wikifier for disambiguation.

The third type of disambiguation relies on *collective disambiguation*, where multiple mentions are disambiguated together. In this approach, target entities should be coherent and semantically related to each other. Many semantic annotation tools combine this approach with the popularity-based method such as TagMe, Wikipedia Miner, and Illinois Wikifier.

The final disambiguation approach is designed on a *graph-based* representation. In this approach, the extracted mentions and candidate concepts form the vertices of a graph. In this graph, the weighted edges between the mentions and candidate concepts represent the contextual similarity. On this basis, disambiguation is formulated as the task of finding a dense sub-graph in which each mention has exactly one edge. AIDA uses a graph-based approach for disambiguation.

In our work, disambiguation of a numeric value concerns the identification of the best matching property for that value from among the identified candidate properties in the detection phase. Our work is primarily based on the popularity-based approach. The selection of the best candidate is based on the cumulative distribution of values associated with each property in the knowledge base. The candidate property that has the closest distribution to the value observed in the given input text is selected.

2.3.3. *Pruning*

In this phase, the concepts that are irrelevant or marginally related to the topic of the input text are pruned. Some annotators such as AIDA perform this task in the disambiguation phase. However others such as DBpedia Spotlight perform it as a post-disambiguation phase.

In TagMe, pruning is based on the average value of each mention's link probability and the coherence between the selected concepts for all of the identified concepts. In DBpedia Spotlight pruning is based on a number of parameters that can be tuned by the user. Wikipedia Miner uses automated prunning similar to TagMe. It uses a topic detector to classify related and unrelated links in a document. Positive training instances for the classifier are the articles that were manually linked to an article in Wikipedia, while negative ones are those that were not. Features of these articles and the places where they were mentioned inform the classifier as to which mentions should or should not be linked. In our work, we do not perform pruning.

There are other areas of research that can be considered relevant to the theme of this paper including the work on ontology learning and knowledge base population. One of the state of the art automatic knowledge extraction tools is FRED [33]. This tool enables robust Ontology Learning and Population (OL&P) from natural language. Ontology learning is the task of acquiring a domain model from a given text and therefore involves parsing of natural language and extracting complex relations and concepts for the purpose of taxonomy induction. FRED does the OL&P task based on Discourse Representation Theory (DRT).

3. Theoretical Model

The overall objective of our work is to find a best describing property and its associated concept for a quantitative value in a short text. Our proposed model is divided into two steps: 1) *Property Identification*, which describes the method for finding the best property; and 2) *Concept Identification* ,which articulates the process to identify its corresponding concept.

In the following subsection, we explain each step in more details.

3.1. *Property Identification*

In order to find the most relevant property that accurately describes a numeric value[i], the first step is to identify the set of properties from the knowledge base that can potentially be related to that numeric value. Let us first provide a theoretic foundation for describing our work.

Definition 1. (Textual Snippet) Let textual snippet $T = [w_1...w_k]$ be a string where w_i $(1 \leq i \leq k)$ is a word. We define $T.dt = w_j \in D$ and $T.r = w_{j-1}$ s.t. w_{j-1} is a numeric value and $2 \leq j \leq k$, and D is the set of all possible datatypes. Further we define, $T.S$ to be the set of all concepts that are spotted in T.

According to this definition, our objective is to annotate $T.r$ with the most relevant property. For instance, for a textual snippet T such as "Motorola RAZR can support up to 64 MB", $T.dt$ is "MB" that represents the megabyte datatype and $T.r$ is "64". Furthermore, with the help of an automated semantic annotation system, one can find all the relevant concepts to T. For this example, $T.S$ is

[i]If numeric values are written in English words, we automatically convert them to numeric form before processing.

$\{Motorola\_Razr, Megabyte, Secure\_Digital\}$[j]. We rely on an existing annotator to provide the values for $T.S$. Now, our task is to find an appropriate property for the value "64" from the list of properties in our knowledge base (e.g. DBpedia).

Definition 2. (Knowledge Base) Let $KB = \{c_1, ..., c_n\}$ be a knowledge base, where c_i $(1 \leq i \leq n)$ is a concept and $c_i.P = \{(p_1^{c_i}, v_1^{c_i}), ..., (p_m^{c_i}, v_m^{c_i})\}$ where $(p_j^{c_i}, v_j^{c_i})(1 \leq j \leq m)$ represents a property-value pair for concept c_i.

For instance, for a concept such as "Motorola_A1000" in DBpedia, one can find a set of property-value pairs such as {(type, Device), (operatingsystem, "Symbian OS 7.0 + UIQ 2.1"), (storage, "24.0 megabyte"), ...}, among others. Based on Definitions 1 and 2, we formally specify the issue of property identification as follows:

Definition 3. (Property Identification) For a knowledge base $KB = \{c_1, ..., c_n\}$ and a textual snippet T, let $P_c = \{p|(p, v) \in c.P\}$ be the set of all properties for concept c. The set of all possible properties in our knowledge base is defined as $UP = \bigcup_{c \in KB} P_c$. The objective is to find the most relevant property $p \in UP$ for $T.r$.

In the context of the earlier example, our goal would be to find a relevant property for "64" which would in this case be "memory" or "storage". As the first step we select a set of concepts from the knowledge base such that they consist of appropriate properties for $T.r$.

Definition 4. (Candidate Concepts) For a textual snippet T, a Candidate concept set is defined as $C(T) = \{c|c \in KB, \exists(p, v) \in c.P\, s.t.\, v.dt = T.dt\}$ where $v.dt$ denotes the datatype for v.

According to this definition, a candidate concepts set will include all concepts that have at least one property with a value whose datatype is equivalent to $T.dt$. In our running example, concept "Motorola_A1000" would be in the candidate concept set, since it has the datatype "megabyte" in the value of one of its properties. In order to choose the best concepts from the members of the Candidate concepts set a ranking function is required. We rank the members of the Candidate concepts set based on their distance to the spots in $T.S$.

Definition 5. (Concept Distance) For concept c and textual snippet T, a distance function is defined as follows:

$$dist(c, T) = \sqrt{\sum_{s \in T.S} \left(\frac{\rho(s)}{r(c, s) + \beta} \right)^2} \tag{1}$$

where semantic relatedness of two concepts c_1 and c_2 is represented as $r(c_1, c_2)$[k]

[j]In our work, we employ DBpedia as the source knowledge base; hence, the complete URI for the concepts would be in the form of `http://dbpedia.org/resource/Motorola_Razr`.

[k]We benefit from TagMe Relatedness API for this purpose in our experiments.

and ρ is the function that returns the confidence score of the mentioned concept in the text (provided by the annotator). Also β is a very small constant for when $r(c, s) = 0$.

Table 3 shows a number of concepts and their distances to the spots in the context of the earlier example. We rank the concepts in the candidate concepts set using the distance function in Definition 5 and hypothesize that less distant concepts have a higher probability to include relevant properties for our purpose. Therefore, we select the top-k concepts from the candidate concept set, denoted by Top Concepts (TC)[l]. Based on the top-k concepts, the set of properties of concepts in TC that have a datatype equal to $T.dt$ will form a candidate property set defined as follows:

Table 3: Concept distances to the spots in $T.S$.

Concept	**Distance**
Theatre_of_War_(video_game)	73.09
Sony_Ericsson_C510	61.63
Motorola_A1000	7.39

Definition 6. (Candidate Properties) Assume TC is the set of top-k concepts based on the distance function in Equation 1. Candidate property set for TC is defined as $CP(TC) = \{p|c \in TC, (p, v) \in c.P, v\ is\ Numeric\ and\ v.dt$
$= T.dt\}$

In order to find the best related property from the candidate property set, for each of the properties in $CP(TC)$, a statistical analysis is done to see which property is more likely to have the numeric value $T.r$. To do so, we perform a statistical analysis over all observed values for each of the properties in CP(TC). In order to analyze the values of each property, we first build a set, called the Number Set.

Definition 7. (Number Set) For a textual snippet T and a given property p, Number Set is defined as $NS(p, T) = \{v_i|c \in KB, (p, v) \in c.P, r(v.dt, T.dt)$
$> \alpha\}$.

The Number Set represents the set of all the numerical values for a specific property observed in the knowledge base as long as the datatype for that value had a semantic similarity score of above threshold α[m] with the datatype of the value that we are annotating ($T.dt$). Based on the Number Set, we calculate the relevance prob-

[l]We set k to 10 in our experiments.
[m]In our experiments, we set α to 0.5.

ability for a given property through its Cumulative Distribution Function (CDF). In CDF, we assume that a Number Set has a Gaussian distribution.

Definition 8. (CDF) For a random variable R we have $Pr[R \leq T.r] \approx CDF(T.r)$. So, $Pr[T.r - \Delta T.r < R < T.r + \Delta T.r] = CDF(T.r + \Delta T.r) - CDF(T.r - \Delta T.r)$ where $\Delta T.r = T.r/100$. Therefore for a property p and a numeric value $T.r$, $Pr(p, T.r) = CDF(NS(p, T), T.r + \Delta T.r) - CDF(NS(p, T$
$), T.r - \Delta T.r)$.

The CDF for a property p and $T.r$ shows the probability of property p being the suitable representation for $T.r$. Table 4 shows a set of properties and their CDF values for the above example where the numeric value 64.0 was considered.

Table 4: The cumulative distance function for the properties.

Property	**CDF**
memory	0.004084854021963902
storage	1.0839821475783218E-4
size	1.316693881592279E-6

Based on the ranking provided through the CDF function, we are able to determine the best property that matches $T.r$. Algorithm 1 details the proposed approach to find the best property that describes a quantitative value mentioned in the input text. Lines 2-8 show how the candidate concepts set (C) is built. C is a subset of KB whose members (concepts) have a property value that includes the datatype of interest. After identifying candidate concepts, we find the top concepts (TC). TC is formed by taking the top-k members of C based on the ranking function in Definition 5 (line 9). Lines 10-16 show the process of forming the candidate properties set (CP). For every concept in the top concept set, all numeric-valued properties of the concepts that have a datatype close to $v.dt$ are chosen for CP. Finally, the property that has the highest probability of having $T.r$ as its value is identified as the property of interest (line 17).

Figure 2 shows the overall process of identifying the corresponding property of the numeric value in above example text. In the first step, the datatype MB in the input text is used to extract candidate concepts. Although these concepts may be far from our objective, they reduce the size of our search space. The next step is to narrow down candidate concepts. In the example, 8 related concepts are narrowed down to a set of 4 concepts. Then with narrowed concepts, candidate properties are extracted. Finally by doing a statistical analysis as explained before, the final property is identified.

Algorithm 1 IdentifyProperty(TexualSnippet T)

```
$C \leftarrow \emptyset$, $CP \leftarrow \emptyset$
for $c \in KB$ do
  for $(p, v) \in c.P$ do
    if $v$ is literal and $v.dt = T.dt$ then
      add $c$ to $C$
    end if
  end for
end for
TC $\leftarrow Top\_k(C, dist)$
for $c \in TC$ do
  for $(p, v) \in c.P$ do
    if $v$ is numeric and $r(v.dt, T.dt) > \alpha$ then
      add $p$ to $CP$
    end if
  end for
end for
return $\arg max_{p \in CP} Pr(p, T.r)$
```

3.2. *Concept Identification*

Now, given that the most relevant property for the numeric value is identified, in this phase, the objective is to find the most relevant concept mentioned in the text which either directly or through inference has the property identified in previous step. Let the identified property of the numeric value in our knowledge base be P. The objective is to find a subject for P with $T.r$ as its object. Note that the desired concept may or may not have the predicate (property) explicitly assigned to it. For example, Ford_XT_Falcon is a concept in the category of Ford_Falcon and it has the property "weight" in our knowledge base. However, Ford_Fairmont_(Australia) is in the category of Ford_Falcon but it does not have the property "weight". We are interested in all the concepts in $T.S$ that can potentially have P in one of their properties, which may not be direct but can be derived through hierarchical subclass inference.

Definition 9. (Candidate Mentions) For a textual snippet T and a property P, a candidate mentions set is defined as $CM(T, P) = \{c | c \in T.S, (P, v) \in c.P\}$.

In case candidate mentions set is empty (none of the mentioned concepts have the property P), we search for similar concepts in the knowledge base that have P as a property. A mention would be considered as a candidate, if there is at least a concept in the knowledge base that has the property P and is at least in one of the mention's categories as expressed in DBpedia's hierarchical concept categories. For example, Ford_XT_Falcon categories are Vehicles_introduced_in_1968,

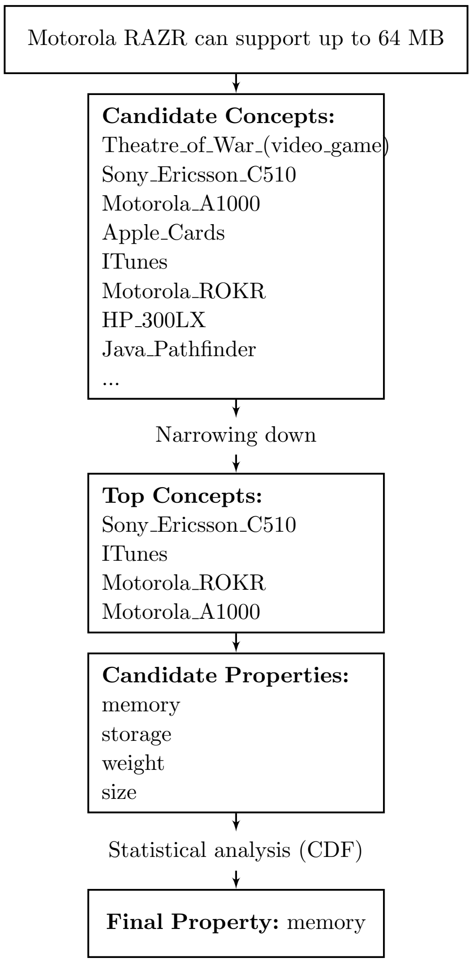


Figure 2: Processes of property identification. All concepts are DBpedia concepts. The complete URI for them is in the form of http://dbpedia.org/resource/Motorola_Razr.

Cars_of_Australia, and Ford_Falcon. In order to identify the related concepts based on shared categories, we define the Related Concepts set as follows:

Definition 10. (Related Concepts) For a concept s and a property P, related concepts set is defined as $RC(s, P) = \{c|c \in O, (P, v) \in c.P, cat(c) \cap cat(s) \neq \emptyset\}$ where *cat* is a function that returns the set of all DBpedia categories for a concept.

Algorithm 2 shows the procedure for identifying the best concept for P. First, if the candidate mentions set is not empty, the concept in CM with the highest confidence (ρ) is selected (lines 1-3). Otherwise, we try to find other related concepts

Algorithm 2 IdentifyConcept(TexualSnippet T, Property P)

```
if $CM(T, P)$ not empty then
  return $\arg max_{c \in CM(T,P)} \rho(c)$
end if
$CM \leftarrow \emptyset$
for $s \in T.S$ do
  if $RC(s, P)$ not empty then
    add $s$ to $CM$
  end if
end for
return $\arg max_{c \in CM} \rho(c)$
```

to each mention that have the property P. If such a concept is found, it will be added to CM (lines 5-9). CM is populated based on the related concepts. Finally, the best concept for property P is the one with the highest confidence value (ρ) in CM (line 10).

As an example in the earlier text “Motorola RAZR can support up to 64 MB”, “memory” was selected as the best property for 64. Based on this identified property, there is only one concept in T.S = $\{Motorola\_Razr, Megabyte, secure\_Digital\}$, i.e. Motorola_Razr, that has “memory” as property. Therefore, Motorola_Razr would be the selected concept. In case there are more than one mentions that have the identified property, the one with the highest confidence is selected.

Now let us suppose that in the above example “storage” was selected instead of “memory”. Then, in this case, candidate properties set would be empty, because none of the members of T.S has the “storage” property. Therefore, we need to consider the related concepts to the concepts in T.S. Here, there is only one concept, i.e., Motorola_Razr in T.S, which has a non-empty related concepts set. This is because we are able to find some concepts such as Motorola_Rokr that share a common DBpedia category with Motorola_Razr, i.e. Motorola_mobile_phones, that at the same time consist of the “storage” property. Therefore, our proposed algorithm identifies Motorola_Razr as the concept and “storage” as the property for the numeric value 64.

Figure 3 shows the concept identification process in a flowchart. It starts with searching for the identified property in the properties of spotted concepts. If such a concept is found, that would be our final concept. However, if it is not found, we look for a concept in the spotted concepts for which there is at least a concept in KB that is similar to it and has the identified property.

4. Experimental Results

In this section, we discuss the evaluation of our work. First we explain our dataset and its curation process. We then go through the experimental setup including the

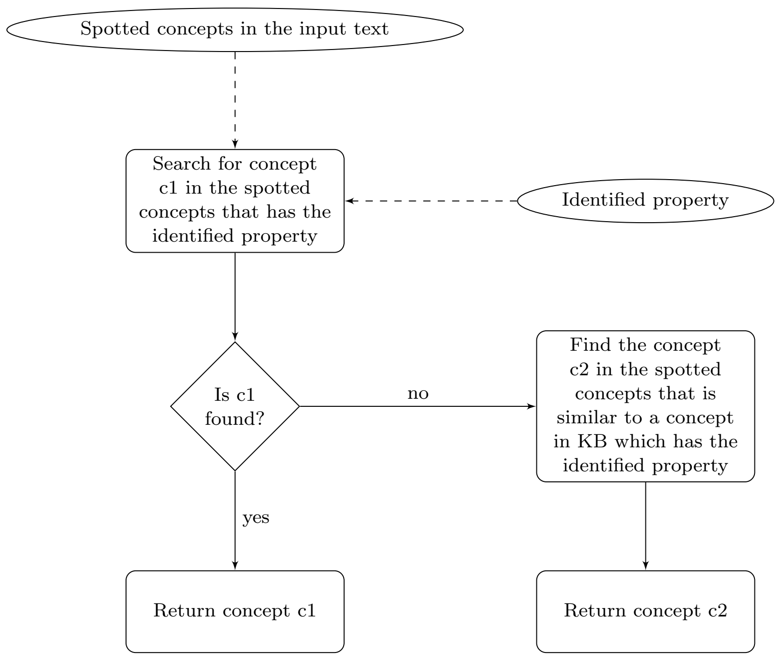

Spotted concepts in the input text
Search for concept c1 in the spotted concepts that has the identified property
Identified property
Is c1 found?
no
Find the concept c2 in the spotted concepts that is similar to a concept in KB which has the identified property
yes
Return concept c1
Return concept c2

Figure 3: Flowchart of concept identification.

description of the machine used for running the experiments, specification of the knowledge base, and database. At the end, we report the results.

4.1. *A description of the Dataset*

In order to evaluate our work, we first developed a gold standard dataset that includes sentences that have quantitative values. Existing datasets that are used for evaluating semantic annotator systems were not suitable as they do not provide gold standard annotations for numeric values. They are mostly developed to evaluate typical annotation systems and they barely include numerical content.

Therefore, we recruited a group of ten Computer Science graduate students at MSc and PhD levels, all of whom had experience in working with semantic annotator systems before, to collect and annotate the gold standard dataset. The recruited graduate students were given a set of suggested concept-property pairs and were asked to collect descriptive sentences about each concept-property pair such that the sentences included quantitative content describing the desired property of the desired concept. Since our knowledge base (DBpedia) does not contain much numerical information about concepts, we provided the participants the suggested concept-property pairs to make sure that the collected gold standard would consist of concepts that exist in the knowledge base. Since the recruited graduate students were given a set of suggested concept-property pairs, there were no overlap between the sentences they collected. The concept-property pairs were chosen so that they cover various domains including electronics, motor vehicles, movies & music, geographical locations, famous people and food. As a final step, all the collected gold standard content were processed by the TagMe semantic annotator and the extracted concepts were stored in the gold standard.

4.2. *Experimental Setup*

The developed gold standard dataset consists of 165 separate entries[n],each of which has a query text and a property-concept pair is associated with it. In the whole dataset, there are 1,225 unique concepts extracted by TagMe. In each instance, there are 9.85 mentioned concepts on average. Each of the entries was selected such that TagMe can find at least one spot for that entry.

With regards to DBpedia, in our experiments, we used DBpedia 3.8. locally installed on a MongoDB server and specifically exploited the "properties" collection. The "properties" collection has over 130 million subject-predicate-object triples. One of our observations when working with DBpedia was that although DBpedia is a great source of information, it does not provide substantial reliable numeric data. In other words, many of the properties that need to have numeric values are missing or have incorrect or too generic datatypes associated with them. Given DBpedia does not enforce a schema, we believe one of the areas that can be improved on this knowledge base is with regards to the quantitative values.

The experiments were run on a machine with 3.20 GHz CPU and 8 GB RAM. For each entry test in our dataset, we ran the system for the entry query text. As output we get a property-concept pair from the system. In the next step, we compare the result of the system with the entry property-concept pair in order to match the concept-property pair output with the ones in gold standard.

Table 5 shows some sample entries and the corresponding concept and properties that were identified. In this table the mentioned entities are the spotted concepts extracted by TagMe. The predicted property and concept are those identified by our method for the highlighted numeric value in that dataset entry. As an example, in the first entry, fuelCapacity (`http://dbpedia.org/property/fuelCapacity`) is identified as the best property and Honda_Gyro (`http://dbpedia.org/resource/Honda_Gyro`) as the best concept for the numeric value 5.0L in the entry.

[n]The dataset is publicly available at `http://ls3.rnet.ryerson.ca/people/mehrnaz/dataset.xlsx`.

Table 5: Some instances of dataset and their results.

Dataset Entry	**Spotted Concepts by TagMe**	**Predicted Property**	**Predicted Concept**
The Honda Gyro is a family of small, three-wheeled, single-occupant vehicles sold primarily in Japan, and often used for delivery or express service with **5.0 L** fuel capacity.	Honda_Express, Japan, Fuel, Family, Engine_displacement, Tricycle, Single-occupant_vehicle, Honda_Gyro, Delivery_(commerce), Service_(economics)	fuelCapacity	Honda_Gyro
Before participating in the reality show, Sibuja Kaliraman weighed **90 kg**, and knew nothing about "presentation".	Reality_television, Kilogram, Sonika_Kaliraman, GMTV, Nothing	weight	Sonika_Kaliraman
Maple syrup is a syrup usually made from the xylem sap of sugar maple, red maple, or black maple trees, although it can also be made from other maple species. It contains **0.1 g** fat every 100 grams of the syrup.	Maple_syrup, Canadian_dollar, Xylem, Doepfer_A-100, Acer_nigrum, Gram, Fat, Acer_saccharum, Plant_sap, Species, Acer_rubrum, Maple	fat	Maple_syrup
The Boy Who Could Fly is a **114-minute** movie about an autistic boy who dreams of flying touches everyone he meets, including a new family who has moved in after their father dies.	Ontario_Highway_114, Dream, The_Boy_Who_Could_Fly, Autism, Minute, Film, Flight, Family, Death, Everyone_(film), Christian_Shephard, Father	runtime	The_Boy_Who_Could_Fly

4.3. *Results*

The goal of our proposed model is to identify the correct concept and property for each of the quantitative values in the dataset entries. The experiments on the gold standard shows an accuracy of 73% for predicting the correct property and 72% for identifying the correct concept. It should be noted that given concept identification is dependent on the performance of the property detection method in our work, when the property was correctly identified, in 87% of the cases the concept was identified accurately as well.

Table 6 shows the accuracies based on the domains of the entries. As mentioned before, concept accuracy is dependant on whether the property is identified correctly. Therefore, we report the third accuracy for concept identification with the correct identified properties (forth column). The number of entries vary for each domain, since for different domains there are different amount of numerical information on DBpedia. For instance, there are huge amount of information in electronics domain, including cellphones, computers, electronic devices, radio stations. On the other hand, we cannot find much information in the military equipment domain within DBpedia.

Table 6: Accuracies based on entries domain.

Domain	**Property accuracy**	**Concept accuracy**	**Concept accuracy (correct properties)**
Celebrities	92.3	92.3	100
Electronics	82.9	77.1	87.9
Food	33.3	60.0	60.0
Geography	41.7	33.3	60.0
Military equipment	66.7	58.3	75.0
Motor vehicle	67.9	75.0	89.5
Movies & music	86.7	80.0	92.3
Overall	72.7	72.1	86.7

5. Conclusion and Future Work

In this paper, we have proposed a technique for semantically annotating quantitative values in textual content. In our work, we identify a property and a concept for the numeric value in the text. In other words, our method finds the property in the knowledge base that the number in the text is the value of and afterward it associates this pair to a concept mentioned in the text. This concept, property, and numeric value form a concept-property-object statement. This statement is in machine-readable format and adds valuable metadata to the text.

We have evaluated our work with a gold standard and we were able to get accuracies of over 70% for both property and concept prediction. To the best of our knowledge, our work is among the first to consider the semantic annotation of numerical values and connecting them to appropriate properties on an external knowledge base such as DBpedia.

While we reach overall accuracies (property and concept) of over 70% on the gold standard, there is one main limitation for our work that we will be addressing in the future work: The core assumption of our work is that a numeric value is proceeded by a unit measure (datatype), e.g. 5.0 L. However, in many real world cases such a unit measure is non-existent after a numeric value. We are interested in predicting the unit measure of a numeric value based on its context. We intend to use Google search for predicting the datatypes by simply using the input text as query text. If the number in the input text actually represents one of the concepts' properties, Google search finds the datatype associated with the number in most of the cases at least in one of its returned results.

One of the areas that we plan to investigate to further improve the performance of our work is to contextualize the consideration of properties with DBpedia categories. In our work, we did not consider DBpedia categories in statistical analysis part of property identification. Narrowing down statistical data to only relevant ones would increase property identification accuracy and subsequently increasing concept identification. One of the examples that can show how this can improve accuracies is the property "length" in DBpedia. "length" can have very diverse values based on the category of its subject, eg. length of a river, length of a car, length of an electronic chip, among others. We intend to filter property values based of the category of their subject and the topic of the input text, and then use them as statistical data in property identification.

Another interesting direction for future work is the selection of a knowledge base (KB). As mentioned before, DBpedia does not include much numerical information about concepts and also in many cases the information provided is inaccurate. Therefore, we can work on either improving the quality of DBpedia numeric data or finding another alternative KB. A notable alternative is Freebase, a graph-based knowledge base used to structure general human knowledge, which is now a part of the Google Knowledge Graph.

Bibliography

[1] Myongho Yi. Information organization and retrieval using a topic maps-based ontology: results of a task-based evaluation. *Journal of the American Society for Information Science and Technology*, 59(12):1898–1911, 2008.

[2] Fei Wu and Daniel S Weld. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118–127. Association for Computational Linguistics, 2010.

[3] Georgios Petasis, Vangelis Karkaletsis, Georgios Paliouras, Anastasia Krithara, and Elias Zavitsanos. Ontology population and enrichment: State of the art. In *Knowledge-driven multimedia information extraction and ontology evolution*, pages 134–166. Springer-Verlag, 2011.

[4] Jelena Jovanovic, Ebrahim Bagheri, John Cuzzola, Dragan Gasevic, Zoran Jeremic, and Reza Bashash. Automated semantic tagging of textual content. *IT Professional*, 16(6):38–46, 2014.

[5] Marco Cornolti, Paolo Ferragina, and Massimiliano Ciaramita. A framework for benchmarking entity-annotation systems. In *Proceedings of the 22nd international conference on World Wide Web*, pages 249–260. International World Wide Web Conferences Steering Committee, 2013.

[6] Liyang Yu. Linked open data. In *A Developer's Guide to the Semantic Web*, pages 409–466. Springer, 2011.

[7] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM, 2008.

[8] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. *Dbpedia: A nucleus for a web of open data.* Springer, 2007.

[9] Thomas Lin, Oren Etzioni, et al. Entity linking at web scale. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 84–88. Association for Computational Linguistics, 2012.

[10] Paolo Ferragina and Ugo Scaiella. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1625–1628. ACM, 2010.

[11] David Milne and Ian H Witten. An open-source toolkit for mining wikipedia. *Artificial Intelligence*, 194:222–239, 2013.

[12] Tim Berners-Lee, James Hendler, Ora Lassila, et al. The semantic web. *Scientific american*, 284(5):28–37, 2001.

[13] Liyang Yu. A web of data: Toward the idea of the semantic web. In *A Developer's Guide to the Semantic Web*, pages 1–18. Springer, 2011.

[14] Victoria Uren, Philipp Cimiano, Jose Iria, Siegfried Handschuh, Maria Vargas-Vera, Enrico Motta, and Fabio Ciravegna. Semantic annotation for knowledge management: Requirements and a survey of the state of the art. *Web Semantics: science, services and agents on the World Wide Web*, 4(1):14–28, 2006.

[15] Michael Erdmann, Alexander Maedche, H-P Schnurr, and Steffen Staab. From manual to semi-automatic semantic annotation: About ontology-based text annotation tools. In *Proceedings of the COLING-2000 Workshop on Semantic Annotation and Intelligent Content*, pages 79–85. Association for Computational Linguistics, 2000.

[16] Stefan Decker, Michael Erdmann, Dieter Fensel, and Rudi Studer. *Ontobroker: Ontology based access to distributed and semi-structured information.* Springer, 1999.

[17] Jeff Heflin, James Hendler, and Sean Luke. Applying ontology to the web: A case study. *Engineering Applications of Bio-Inspired Artificial Neural Networks*, pages

715–724, 1999.

[18] Lawrence Reeve and Hyoil Han. Survey of semantic annotation platforms. In *Proceedings of the 2005 ACM symposium on Applied computing*, pages 1634–1638. ACM, 2005.

[19] Alexiei Dingli, Fabio Ciravegna, and Yorick Wilks. Automatic semantic annotation using unsupervised information extraction and integration. In *Proceedings of SemAnnot 2003 Workshop*, 2003.

[20] Maria Vargas-Vera, Enrico Motta, John Domingue, Mattia Lanzoni, Arthur Stutt, and Fabio Ciravegna. Mnm: Ontology driven semi-automatic and automatic support for semantic markup. In *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, pages 379–391. Springer, 2002.

[21] Philipp Cimiano, Siegfried Handschuh, and Steffen Staab. Towards the self-annotating web. In *Proceedings of the 13th international conference on World Wide Web*, pages 462–471. ACM, 2004.

[22] Paul A Kogut and William S Holmes III. Aerodaml: Applying information extraction to generate daml annotations from web pages. In *Semannot@ K-CAP 2001*, 2001.

[23] Borislav Popov, Atanas Kiryakov, Angel Kirilov, Dimitar Manov, Damyan Ognyanoff, and Miroslav Goranov. Kim–semantic annotation platform. In *The Semantic Web-ISWC 2003*, pages 834–849. Springer, 2003.

[24] Diana Maynard. Multi-source and multilingual information extraction. *Expert Update*, 6(3):11–16, 2003.

[25] Siegfried Handschuh, Steffen Staab, and Fabio Ciravegna. S-cream—semi-automatic creation of metadata. In *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, pages 358–372. Springer, 2002.

[26] Stephen Dill, Nadav Eiron, David Gibson, Daniel Gruhl, R Guha, Anant Jhingran, Tapas Kanungo, Sridhar Rajagopalan, Andrew Tomkins, John A Tomlin, et al. Semtag and seeker: Bootstrapping the semantic web via automated semantic annotation. In *Proceedings of the 12th international conference on World Wide Web*, pages 178–186. ACM, 2003.

[27] Paolo Ferragina and Ugo Scaiella. Fast and accurate annotation of short texts with wikipedia pages. *arXiv preprint arXiv:1006.3498*, 2010.

[28] 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*, pages 1–8. ACM, 2011.

[29] John Cuzzola, Zoran Jeremic, Ebrahim Bagheri, Dragan Gasevic, Jelena Jovanovic, and Reza Bashash. Semantic tagging with linked open data. In *CSWS*, pages 52–53. Citeseer, 2013.

[30] Pavel Mihaylov. D4.5 integration of advanced modules in the annotation framework. 2011. Deliverable of the NoTube FP7 EU project (project no. 231761), http://notube3.files.wordpress.com/2012/01/notube_d4-5-integration-of-advanced-modules-in-annotationframework-vm33.pdf.

[31] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 782–792. Association for Computational Linguistics, 2011.

[32] Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1375–1384. Association for Computational Linguistics, 2011.

[33] Valentina Presutti, Francesco Draicchio, and Aldo Gangemi. Knowledge extraction based on discourse representation theory and linguistic frames. In *Knowledge Engineering and Knowledge Management*, pages 114–129. Springer, 2012.