Extracting Semantic Networks among Named Entities from Websites

Abstract. To enable machine processing of webpages, it is important to identify the relationships among named entities. Named entities, like, people, organizations, and places are important pieces of information that must be extracted. The scale of the web indicates that manual extraction is not feasible. We propose a system that automatically constructs a semantic network of named entities from webpages and stores them in a queryable repository. End-users can select a part of the semantic network using queries and explore the semantic network using a visual interface to understand the relationships among the entities of interest. We employ shallow parsing to rapidly extract the semantic network. For certain applications, where precision is important, we propose a machine-learning-based technique to extract the semantic relationships.

Keywords: Relationship Extraction, Semantic Network, Named Entity Extraction

1 Introduction

The number of pages on the World-Wide Web is ever increasing. Extracting information from this vast resource manually is an onerous task. End users can use search engines to perform keyword searches on a corpus of web documents. However, search does not satisfy the needs of an end-user who wants to explore the information available from the web. Suppose the end-user is interested in specific questions about a person, say, for a researcher’s work address, his or her research interests, and examine the network of collaborators of the researcher. A search for the person on the web returns several webpages that the end-user must navigate manually to cull the information required. Instead of using simple keyword searches, the end-user may be interested in exploring and identifying patterns in the information present in the set of documents. A simple, yet powerful, way to visualize the relationships among named entities in text on webpages is by extracting a semantic network among named entities in the pages and showing it to the end-user as a navigable graph. It would be nice to be able to derive information using a visual-analytics tool that allows the user to query information about named entities on the web and navigate a semantic network to determine the relationship of the named entity with other named entities using the information available from the World-Wide Web.

In this paper, we outline a system that (1) extracts named entities and their relationships from the textual information present in webpages, (2) stores the relationships in an OWL repository, (3) enables end-users to pose declarative queries, and (4) enables end-users to navigate and explore the semantic network.

The system can extract the primary textual content of webpages using techniques that we have reported before in [1]. The algorithm identifies and strips a webpage of non-textual content like navigation bars, banner advertisements, images, etc. leaving

---

1 We use Sesame, an open source RDF database (http://www.openrdf.org), as the OWL repository.
behind the textual part of the webpage. In this work, we assume that the webpage has already been cleaned of the non-textual information and we address the second part of the problem of automated extraction of information from the textual part of the webpages.

In this work, we use GATE[2] to derive named entities from text. GATE uses a gazetteer to derive location names. Our first algorithm involves shallow parsing of the text documents, comparing against gazetteers for person and place names, and identifying the noun or verb phrases occurring between two named entities as relationships. We use a simple set of heuristic rules and machine learning algorithms using text features to extract the relationships between entities as mentioned in the corpus. The relationships derived by our toolkit hold between two named entities and have two attributes: “spatial” and “temporal”. For example, from the statement, “John met Jack in London”, we derive the relationship “met” between “John” and “Jack” with a spatial attribute “London”. The problem of extraction is more complex when the sentences we parse are not as simple as the one mentioned above. We depend upon features such as “in <place>?” etc. to derive patterns using regular expressions or using a supervised learning approach using Support Vector Machines to derive the relationships among the named entities.

The extracted named entities and their relationships are stored in an OWL repository. The advantage of extracting and storing all the named entities and their relationships in the OWL repository is that they can be queried to select entities and relationships of interest to the end-user, and only the filtered semantic network presented to the end-user for exploration. In our work, we do not need the existence of an existing ontology to assist in our information extraction or semantic network construction.

The end-user can now visually navigate over the semantic network to identify the relationships between different people, explore which document a piece of information is extracted from, detect whether there are conflicting reports from different documents in the repository, and examine the spatial and temporal attributes of the different relationships depicted in the semantic network. Instead of having to look at all the documents, the end-user can easily select parts of the semantic network to explore by providing constraints on person and organization names of interest, desired locations, and time ranges. Such a selection facility drastically reduces the information that the end-user has to wade through. Therefore, deriving such a semantic network provides the information in a queryable, filterable, and scalable interface to large text repositories that are not entirely possible to process manually.

Previous research has addressed the problem of named entity extraction from text documents[2], and the problem of extracting a few specific pre-defined relationships from text[3, 4]. Prior research has also attempted to augment and construct ontologies from text using “human language technologies” [5]. However, the extraction of all relationships between all named entities in a corpus and displaying the relationships among the named entities has not been attempted.

Extraction of information from text documents is a hard problem because context-sensitive natural language is not amenable to efficient and accurate automatic processing. Our system is a part of a multi-modal information integration system that allows for an integration of information from knowledge bases, text documents, and
Our mediated information integration system makes it possible to connect
the information in the text report to the existing background information.

The rest of this paper is organized as follows. In Section 2, we discuss some
related work.

2 Related Work

Katz has built the START Information Server[6] that can extract information
from natural language texts. The server can extract triples and ternary expressions of
the format <subject relation object> and stores them in a knowledge-base. Typically,
the relations are verbs. The system also contains rules that indicate the properties of
the relations. For example, an S-rule may be,

If <<subject verb object1> with object2>
Then <object2 verb object1>

Provided verb ε emotional-reaction class

The verb surprise is in the class of emotional-reaction verbs but the verb present is
not.

Their system can also allow for manual annotations to the text. Users’
queries are answered by matching the verb and the object values, if supplied, to the
facts stored in the knowledge base. The knowledge-base uses the S-rules and other
rules existing in the system to answer the queries.

Based on the START system, Katz et al. [7], have built the Omnibase system
that enables question answering over web information sources. START “translates
user queries into a structured request”. Omnibase allows queries that are triples
representing ternary relationships of the form <subject property value>. While this
system cannot handle higher order relationships and other more complex queries like,
“What is the best route between Boston and New York”, the authors claim that about
37% of TREC-9 and 47% of TREC-2001 questions from the QA track can be
represented as triples.

Our system is constructed using shallow parsing and simple techniques to
keep it scalable. The cost of the simplicity is that we may not be able to answer
sophisticated questions that Katz’ system allows. Our focus is instead on extracting a
few “special” relationships that are useful to an end-user, and to visualize the
extracted semantic network as a graph to allow the end-user to navigate and examine
the relationships between named entities mentioned in the text documents.
3.0 System Overview

Fig. 1. A system architecture for text-based integration, query answering, and semantic network construction using multi-format information.

In Figure 1, we show an architecture for a social network generator and a query answering system from text documents. Textual information extracted from text, images, audio and video can be stored in a repository in the Web Ontology Language (OWL) format\(^2\). The repository enables efficient retrieval via a declarative query language. Appropriate indexes are set up on the OWL repository to allow for efficient query processing. The question answering system is responsible for translating the user query to the vocabulary used while creating the data in the OWL repository. The social network constructor constructs a social network from the information available in the OWL repository. Note that the end-user may not want to visualize the entire social network. The end-user has the capability to indicate filters of interest, like a particular person, and configuration parameters like how many degrees of separation should be displayed to customize the part of the social semantic network that the end-user seeks.

\(^2\) In the rest of the report, we use the terms OWL repository and database interchangeably.
In this work, we have focused on the entity relationship extractor. We allow basic querying using the repository’s query engine. We have also interfaced our tool with a visualization toolkit (see Fig. 2) that enables the end-user to browse through the semantic network and filter parts of the semantic network that are interesting to the end-user (see screenshots in the appendices).

![Fig. 2. A snapshot of the semantic network visualization interface.](image)

4.0 Entity Relationship Extraction

The first step is to build an extractor that mines text documents to construct semantic networks from information available at the sentence-level in documents. Identification of semantic networks using links between websites and other analysis of documents using information extraction and statistical techniques has been performed before. However, such techniques fail to identify sentence-level information. For example, there may be one mention in one sentence among several hundred documents of an event that may be of interest to the end-user. “Mr. Gin Baden met Mr. Bal Zachary at Gondwanaland on February 21\(^{st}\) 2005 to discuss business matters.” If this is the only mention of this event in a corpus of 500 documents, then extracting macro-trends by analyzing the whole set of documents is of little use to discover this information.

In this work, we propose to build an information extraction tool that uses natural language processing and machine learning techniques to extract information from sentences appearing in text documents. More specifically, we seek to extract “relationships” among named entities.

We use the term “relationships” loosely to indicate any connection between a named entity like an organization or a person to another person, organization, place, or time. For example, in the example shown above, we extract a relationship between Mr. Gin Baden and Mr. Bal Zachary. The relationship is minimally identified as “met” and the temporal and spatial attributes of the relationship are set to “Gondwanaland”, and “February, 21\(^{st}\) 2005”. Relationships may not have their spatial and temporal attributes set when the sentence does not contain any spatial and temporal attributes, e.g., in the sentence “Mr. G met Mr. B”. Some relationships may also not have a second named entity, but only have one named entity and a spatial attribute. For example, “Dokan, Inc. is located in Baltimore, MD.” has a relationship named “locatedIn”, the named entity is “Dokan, Inc.”, and the spatial attribute is...
“Baltimore, MD”. This extracted information is stored in a database. Additionally, along with the information, we seek to store which document(s) it was derived from.

4.1 Levels of Relationship Extraction

In this work, we attempt to extract information at two levels:

1. Rapid Relationship Extraction: At this level, we want to extract relationships from sentences containing any references to people, organizations, place and time. A relationship has at least two named entities, one of which is a person or organization and the other is any of person, organization, place, and time. The verb group or the noun group (in case of verbs like “is”, “was”) connecting the named entities is extracted from the sentence and stored. For example, from the sentence “Zerkin had asked Moussaoui if he believed his defense team is in a conspiracy to kill him.” we extract the relationship “had asked” between the entities “Zerkin” and “Moussaoui”. This “triple” is then represented in OWL and stored in the database. Because, in this method, we rapidly extract the named entities and their relationships as expressed by verb or noun groups, but do not attempt to derive the deeper semantics of the sentences, we refer to this method as “rapid information extraction”.

2. Precise Relationship Extraction: At this level, we do not just want to extract snippets of code and refer to them as relationships. Instead, we want to extract precise pre-defined relationships from the text documents. For example, one piece of information we want to extract is whether a person was actually present at a location (instead of, say, just speaking about it). Let us call this relationship: “visited”. For example, in the sentence, “John went back to his house in Jerusalem”, we have a positive example, where John was actually present at the location. Rapid information extraction would extract the relationship as “went back to”, but this relation would not be stored under the relationship “visited”. A negative example would be, “John likes the city Jerusalem.” In this case, there is no evidence that John actually visited the city. We only extract the relationship, “visited” between “John” and the spatial attribute, “Jerusalem”, if there is a positive example in the text. The semantic relationships extracted in this case are more precise and are an aggregation of relationships that are semantically similar. We generate these relationships using supervised learning techniques.

4.1.1 Comparison of the Two Levels of Relationship Extraction

Information extraction at both granularities is necessary. The rapid relationship extraction crudely identifies all relationships extracted from sentences in the corpus of documents and that is stored in a database. Extracting information from text documents and inserting them in a database introduces some structure to the data. This process is useful for the following reasons: (a) the data can be queried using the repository’s query engine that supports more sophisticated query processing rather than querying using search keywords, (b) no information regarding named entities and their relationships are lost because all named entities and their relationships are recorded as triples in the repository, and (c) rare events are easier to identify using a query.

The advantage of the rapid relationship extraction is that all named entities and their relationships are now stored in a database and can be queried declaratively.
There is substantial savings from the perspective of the end-user because the end-user does not have to browse through the entire dataset. However, storing the rapidly extracted information and retrieving this information is not enough. Natural language is characterized by its semantic heterogeneity. There are problems handling synonyms, homonyms, hyponyms, and other relationships between words. For example, a query on the created database for relationship “visited” should also, perhaps, return the entities linked by the relationship “toured”. Therefore, when the relationship extracted rapidly, the semantic heterogeneity must be handled before answering end-user queries.

Precise relationship extraction requires identifying the types of relationships that are most essential to extract. Then, we must build machine-learning based classifiers that can be trained to classify sentences to determine whether the sentences contain the relationships or not. The disadvantage of this method is that the interesting relationships have to be identified when the system is being set up and the end-user can only choose from among a set of interesting relationships that the builder of the system allows. The advantage of this method is that this method of relationship generation is typically more precise than the rapid relationship extraction. Besides, the (semantically) same relationship is not represented as multiple relationships because the sentences used different verb or noun groups to refer to them.

5.0 Our Algorithms

Named entity extraction systems can tag documents to identify occurrences of named persons, organizations, and places, and in some cases temporal information.

5.1 Rule-Based Sentence Parsing and Entity Relationship Detection

This component of our system is essentially a collection of grammatical rules that are used to parse sentences and extract information from them.

Currently, our extraction tool is developed using the GATE system. The GATE system is a rule based natural language processing tool and provides a Java framework. We use GATE to extract named entities, find coreference between nouns and pronouns, and do shallow parsing (Part-Of-Speech, Noun Chunk, and Verb Group tagging). Based on the information provided by GATE, we perform a deep parsing. The objective of deep parsing is use the syntactic information extracted from shallow parsing to identify dependency relationships using a Subject-Verb-Object (SVO) structure in each sentence. We use the SVO structure to represent the rapidly extracted relationships. For precise relationships, we use predefined relation names to replace the verb in SVO structure. Finally, all the extracted information will be put together as an ontology. Every named entity (subject or object) will become a concept. The relationships between entities will be represented as properties between two concepts. We use the Web Ontology Language (OWL) to represent ontologies.
In order to process sentences correctly, we first identify the structure of sentence. We use a shallow parser, Minipar\(^3\), to create a parse tree. Based on the parse tree and the information provided by GATE, we can construct the SVO structure easily. If there are more than one subjects or objects, we associate the relation to each distinct pair of subject and object.

For example, Fig. 4 shows the parse tree with named entity tagging for a sentence: “John went to Philadelphia and met his best friend Sean.” Start from the root node “went”, the first relation (John, located in, Philadelphia) can be easily extracted by finding they are both connected by the verb “went”. The second relation (John, met, Sean) can be extracted since Minipar has identify John is the subject of the second verb “met” and both person are connected by the verb met.

We evaluate the performance of our rule-based relation extraction system with IEEE VAST 2006 Contest dataset. The VAST dataset consist of 1182 fictitious news stories. We randomly select 26 stories and perform the rule-based extraction. We demonstrate the scalability of our method in Figure 5. The results in Fig. 5 show the running time for our extraction algorithm is quadratic as the file length and number of sentences increase.

---

\(^3\) [http://www.cs.ualberta.ca/~lindek/minipar.htm](http://www.cs.ualberta.ca/~lindek/minipar.htm)
**Rule Example: Handling Non-Informative Verb Forms**

Apart from parse tree, we use other rules to assist the relationship extraction. For example, when non-informative verbs like “is”, “was”, etc. are present, we extract the noun phrase following the verb, if any, to denote the relationship. In a sentence, “John is a resident of State College”, we extract the relationship “resident” between the entities “John” and the location attribute “State College”, instead of extracting a relation “is”.

### 5.2 Precise Relationship Detection Using Support Vector Machines

As pointed out above, for precise relationship detection, we use a supervised learning technique to build a classifier for each relationship we intend to detect. We use an SVM (SVM-Light) to perform the classification. Let us consider the precise relationship “located” for persons. Sentences that indicate some person is located in some place are marked as a positive example. The set of features we use are the bag of words that appear between the entities.

We also intend to empirically evaluate feature selection based on measures of information content of the words and determine if careful feature selection has an effect on the precision of the extraction.

Besides bag of words kernel, we also use the tree kernels proposed in [4, 9, 10]. Tree kernels utilize the syntax structure of sentences, therefore, they usually outperform bag of words kernel in NLP tasks [9]. We use parse trees generated by Minipar and augment each node with a list of features. The parse trees generated by Minipar are dependency trees. Therefore, we implement the tree kernels described in

---

4 The machine-learning based relationship detector has not yet been implemented, but has just been designed.
The features we use include PoS tag, entity type, chunk tag, relation augment, and Wordnet hypernyms. We also intend to experience combining tree kernels and bag of words kernel to improve accuracy of relation extraction.

Fig. 5. Elapsed time in milliseconds for rule-based relation extraction.

References