Question Answering Based on Semantic Graphs

Lorand Dali
Jožef Stefan Institute
Jamova cesta 39
1000 Ljubljana, Slovenia
+386 1 477 3144
lorand.dali@ijs.si

Delia Rusu
Jožef Stefan Institute
Jamova cesta 39
1000 Ljubljana, Slovenia
+386 1 477 3144
delia.rusu@ijs.si

Blaž Fortuna
Jožef Stefan Institute
Jamova cesta 39
1000 Ljubljana, Slovenia
+386 1 477 3934
blaz.fortuna@ijs.si

Dunja Mladenić
Jožef Stefan Institute
Jamova cesta 39
1000 Ljubljana, Slovenia
+386 1 477 3377
dunja.mladenic@ijs.si

Marko Grobelnik
Jožef Stefan Institute
Jamova cesta 39
1000 Ljubljana, Slovenia
+386 1 477 3778
marko.grobelnik@ijs.si

ABSTRACT
In this paper we present a question answering system supported by semantic graphs. Aside from providing answers to natural language questions, the system offers explanations for these answers via a visual representation of documents, their associated list of facts described by subject – verb – object triplets, and their summaries. The triplets, automatically extracted from the Penn Treebank parse tree obtained for each sentence in the document collection, can be searched, and we have implemented a question answering system to serve as a natural language interface to this search. The vocabulary of questions is general because it is not limited to a specific domain, however the questions's grammatical structure is restricted to a predetermined template because our system can understand only a limited number of question types. The answers are retrieved from the set of facts, and they are supported by sentences and their corresponding document. The document overview, comprising the semantic representation of the document generated in the form of a semantic graph, the list of facts it contains and its automatically derived summary, offers an explanation to each answer. The extracted triplets are further refined by assigning the corresponding co referenced named entity, by resolving pronominal anaphors, as well as attaching the associated WordNet synset. The semantic graph belonging to the document is developed based on the enhanced triplets while the document summary is automatically generated from the semantic description of the document and the extracted facts.

WWW 2009, April 2009, Madrid, Spain.

Categories and Subject Descriptors
I.2.7 [**Natural Language Processing**]: *Text analysis*.
H.3.4 [**Systems and Software**]: *Question-answering (fact retrieval) systems.*

General Terms
Algorithms, Design

Keywords
Natural language processing, question answering, triplet, text mining, summarization, semantic graph.

1. INTRODUCTION
Providing structured and synthesized information has become increasingly important, and even more so if it concerns yielding answers to questions posed in natural language. The goal is not only to find a certain piece of information, but also to be able to easily scan through it, obtain the most relevant parts and links to the related information.

As a response to this challenge, we present an enhanced question answering system that integrates two important functionalities: providing answers to questions and browsing through the document that supports the answer. The questions follow a predetermined template, whereas the answers are yielded based on the previously extracted information, in the form of subject – verb – object triplets. Furthermore, the system retrieves the sentences that support these answers, as well as the documents containing the sentences. It integrates three possibilities of further exploring the relevant documents, which provide a document overview: by analyzing the list of facts (subject – verb – object triplets) extracted from the document, by visualizing the semantic representation of the document and by browsing the document summary.

Previous work has typically focused on one topic only (question answering, summarization, semantic representation and visualization of documents) and we see as advantage of the proposed system in combining these topics together.

Natural language interfaces and search are popular research topics. Many of the previous approaches, like Aqualog [1] and QuestIO [2, 3] query structured data stored in ontologies. Aqualog has a restricted grammar and restricted vocabulary to which the query has to be compatible. QuestIO does not require a fixed grammatical structure of the question, but the words which it can handle are limited because of the dependency on an underlying ontology. Our system derives the answers only from unstructured text, which means that the things the user can ask about are not limited or domain specific. However the questions must be in fixed grammatical forms for our system to 'understand' them. TextRunner [4] is similar to our system in the way that it also consists of structured queries on unstructured text but the difference is that we also provide a natural language interface to the search. The Calais[1] system creates semantic metadata for user submitted documents. This metadata is in the form of named entities, facts and events. In the case of our system, named entities and facts represent the starting point; they are further refined by applying co reference resolution for named entities, anaphora resolution and semantic normalization based on WordNet [5] for facts. This process enables the construction of a semantic description of the document in the form of a semantic directed graph where the nodes are the subject and object triplet elements, and the link between them is determined by the verb. The initial document, its associated facts and semantic graph are then utilized to automatically generate a document summary. Powerset[2] enables search and discovery in Wikipedia and Freebase, by entering keywords, phases or simple questions. The search results contain aggregated information from several articles, as well as a list of facts related to people, places and things. What distinguishes our system from Powerset resides in the way we describe the answer: by a visual representation of the document in the form of a semantic graph and by the document summary, which is automatically extracted based on the document semantic graph.

The following section gives a brief system overview, while the remainder of the paper deals with describing the system components in more detail. We conclude by outlining the conclusions and future work.

2. SYSTEM OVERVIEW

At the highest level of abstraction, the presented system combines question answering, summarization and document visualization functionalities. The user obtains answers based on the facts previously extracted from text. Moreover, the sentences that support the answer, as well as the documents containing these sentences, are also retrieved. The relevant documents can be further explored with the aid of a document overview functionality that consists of a document summary, a semantic representation of the initial document and a list of facts extracted from the document.

[1] Calais web page: http://www.opencalais.com/

[2] Powerset web page: http://www.powerset.com/

Figure 1 describes a use case where a natural language question is posed. The system searches for possible answers to the question and, when found, each answer is linked to the sentences that support it and the document that contains these sentences. The system provides a document overview by retrieving the document semantic graph, the list of subject – verb – object facts and the automatically generated document summary of variable length that is set interactively by the user.

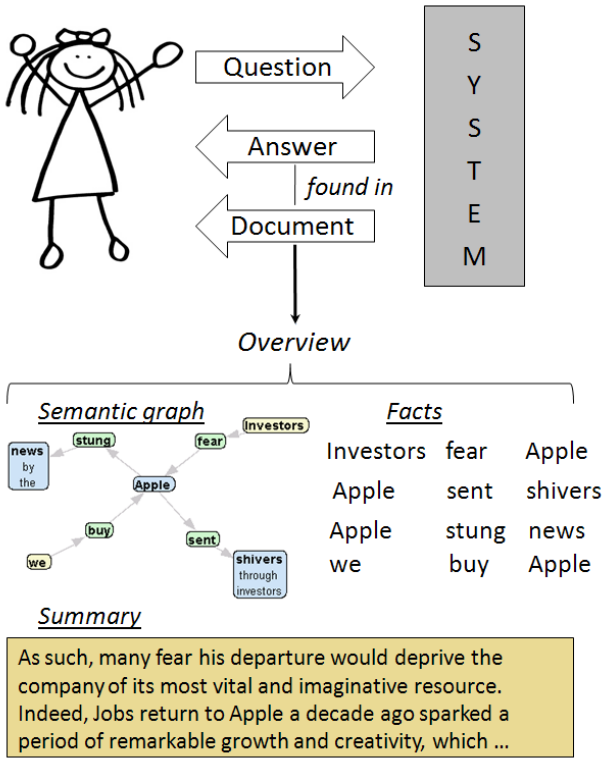

Figure 1. Illustration of the system functionality.

In Section 3 we describe the main system components in more detail, starting with the question answering functionality, explaining triplet extraction, query structure, question analysis and answer generation. Next, in Section 4, we describe the semantic graph generation process, and in Section 5 the document summarization technique.

3. QUESTION ANSWERING

The question answering system presented in this paper answers natural language questions based on the facts that are extracted from text. The facts are represented by subject-verb-object triplets. Indexing these triplets enables searching them by leaving any of their elements (subject, verb, object) undetermined. Figure 2 shows how the answer to the question *Where do tigers live?* is found.

Linguistic analysis of the question yields the query which has to be issued to the triplet search engine in order to get the answer. In general the query is organized as a tree whose leaves are triplets with one or more elements possibly undetermined. In our example the query triplets which have to be matched to the stored triplets

are (*tiger, live, ?*) but also (*tiger, inhabit, ?*) because *live* and *inhabit* are synonyms. Synonymy relations are given by WordNet. Finally *Sumatra* is found to be the answer to the question.

In the following sections the important parts of the question answering system (triplet extraction, indexing, query structure and linguistic analysis) are explained in more detail.

3.1 Triplets

The triplet is a representation of the information contained in a sentence. It consists of the subject, the verb and the object of the sentence it represents, and is the basic unit of data on which the question answering system is built. Section 4.2 gives a more detailed description of the triplet extraction process.

3.1.1 Search Engine

Triplets extracted from sentences are indexed for fast retrieval using Text Garden's library [6] search engine. The engine supports triplet pattern queries, for example: return all the triplets with *tiger* as subject and *live* as verb. Special attributes, which can be assigned to parts of triples, are also indexed and can be queried. Example of such attributes would be negations of verbs.

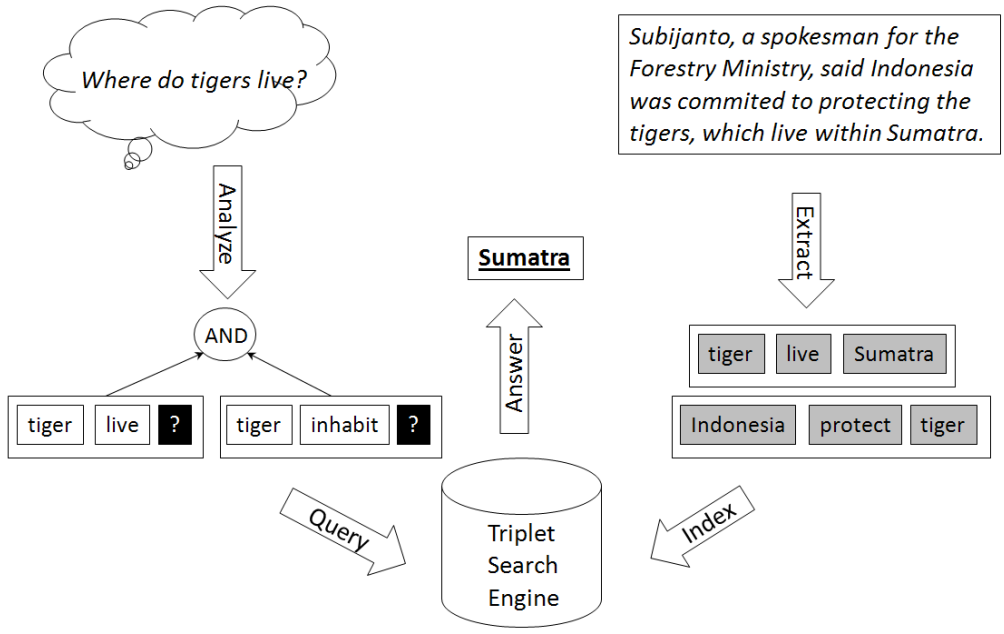

Figure 2. The way a question (*Where do tigers live?*) is answered by the system.

3.2 Queries

The query structure shown in Figure 3, is designed according to the composite pattern, and is meant to be easy to reuse and extend.

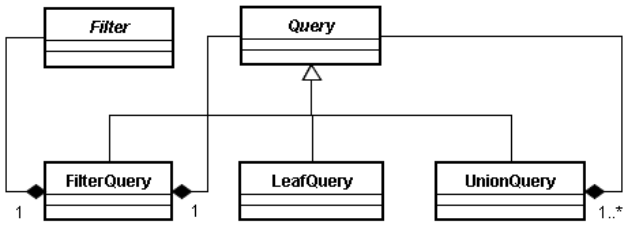

Figure 3 The query structure.

All query types which extend the abstract class Query return a set of triplets which match the given query. The simplest query type is the LeafQuery. It is the only one which queries the triplet search engine directly by sending it a subject, verb and object as described in the previous section. The other query types are used to combine triplet sets resulting from other queries. Such is the UnionQuery which has many queries and makes the union of all triplet sets returned by the queries it contains. The FilterQuery has a query of which results it passes through a Filter, and keeps only those that pass. The Filter is defined to filter out triplets that do not satisfy some desired properties, where the properties are defined over one of the triplet components and roughly correspond to some of the predefined types of questions (e.g., the verb is negated, object or subject is numeral).

Filter is the abstract class from which all filters inherit. Its role is to delete some elements from a set of triplets according to some conditions specific to each concrete filter. Some examples of filter types are: NegativeFilter (only triplets with negated verb), PrepositionFilter (only triplets where the object or subject has a preposition), NumeralFilter (only triplets where the object or subject is quantified by a numeral), AttributeFilter (only triplets of which a certain element has a certain attribute, like an adjective), NounFilter (enforces that a certain element be a noun), ReasonFilter (only triplets in whose sentence of occurrence a reason is expressed), TimeFilter (only triplets in whose sentence of occurrence a time is given), etc.

3.3 Question Analysis

The goal of question analysis is to determine what type of question is being asked, and to build a query whose result will be used to give the answer(s) to the question.

3.3.1 Question types

The following types of questions are currently supported by the system:

- Yes/No questions (*Do animals eat fruit?*),
- list questions (*What do animals eat?*),
- reason questions (*Why do animals eat fruit?*),
- quantity questions (*How much fruit do animals eat?*),
- location questions (*Where do animals eat?*) and
- time questions (*When do animals eat?*).

3.3.2 Linguistic analysis

```
QuestionTypes ← {YesNoQ, ListQ, WhyQ, QuantQ, LocQ,
                 TimeQ, UnknownQ}
Tags ←{SQ, NP, VP, SBARQ, WHNP, WHADVP,
       WHADJP}

function QUESTION_ANALYSIS(question)
returns: type, query
PARSE(question)
type ←UnknownQ
query ← NULL
if TN = SQ then
   if N1 = NP then
      type ← YesNoQ
      if N2 ← NP then query ← (N1, N0, N2)
                 else query ← (N1, N2, obj(N2))
if TN = SBARQ then
   if N0 = WHNP and N1 = SQ then
      type ← ListQ
      if N3= VP then query ← (?, N3, obj(N3))
      else if N4 = NP and N5 = VP then
         query ←(N4,N5,?)
   if N0 = WHADVP and N1 = SQ and N4 = NP then
      type ← WhyQ
      if N5 = NP then
         query ← reasonFilter((N4,N3,N5))
      if N5 = VP then
         if 'where' in question then
            type ← LocQ
            query ← prepositionFilter((N4, N5, ?))
         else if 'when' in question then
            type ← TimeQ
            query ← timeFilter((N4, N5, ?))
         else
            query ← reasonFilter(N4,N5,obj(N5))
   if N0 in {WHADJP, WHNP} and N1 in {S, SQ} then
      type ← QuantQ
      if N3 = VP then
         query ← numeralFilter((?,N3,?))
      if N3 = NP and N4 = VP
         query ← numeralFilter((?,N4,?))
```

Figure 4. Pseudo code for the question analysis rules.

The analysis starts with parsing the question to obtain a parse tree in the Penn Treebank [7] format. This has been done by using the OpenNLP[3] parser. Having the parse tree, the rules encoded in pseudocode in Figure 4 are applied to it. In the fragment of a Penn Treebank parse tree shown in Figure 5 the nodes are given names to help referencing them in the rules. The tags which appear in the pseudo code have the following meanings: SQ and SBARQ show interrogative clauses, NP is a noun phrase, VP is a verb phrase, WHNP, WHADVP and WHADJP are *Wh*-noun, adverb and adjective phrases respectively. This means that they contain a *Wh*-adverb like *who*, *why*, *how*, *what*, etc.

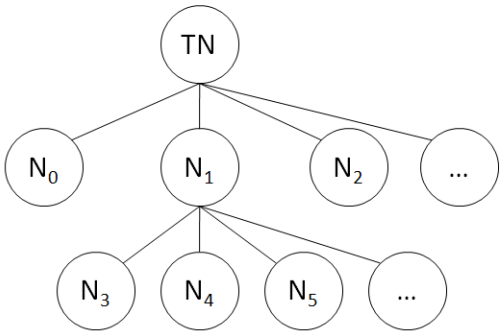

Figure 5. A parse tree fragment.

The function *obj*(V) returns all the objects of the verb V, and *filter*((Subj,Verb,Obj)) is the result of applying a certain filter to the query. It should be noted that a query is in general not only a LeafQuery with a filter as is apparent from the pseudo code, but it has the full structure described in the previous section, a fact that has been omitted in order to simplify the presentation of the rules. Another thing, not detailed here, is how the system, using WordNet, normalizes words and extends queries with synonyms and hyponyms. Also AttributeFilters are often used to match the adjectives in the question.

3.4 Answer Generation

As explained in the section about queries, the result of a query is a set of triplets.

If the question is a Yes/No question, then the resulting set of triplets is split into two groups. One where the polarity of the verb (that is, if it is negated or not) matches the polarity of the verb in the question; this will be the group that supports the answer *Yes*. The other group consists of those triplets where the polarity of the verb is the opposite of the polarity of the verb in the question; this will be the group that supports the answer *No*. Both answers are justified by the supporting sentences of the triplets of their group.

If the question was a list question, a quantity question, or a location question, then the answer will consist of many items. The items are taken from the set of triplets returned by the query, by knowing from question analysis which element of the triplets contains the answer to the question. The relevant elements of the triplets are grouped and ordered decreasing by frequency. Each group obtained is an item in the final answer.

[3] OpenNLP web page: http://opennlp.sourceforge.net/

If the question was a reason question or a time question, then no clear answer is given by the system. Instead the sentences which contain the triplets returned by the query are given as an answer.

4. SEMANTIC GRAPHS

As already described, once the system has found some answers, an explanation is provided for each of them, in the form of facts, sentences and documents it was derived from. As proposed in [8] a document can be presented by its associated semantic graph, thus providing an overview of its content. The graph is obtained after processing the input document and passing it through a series of sequential operations composing a pipeline (see Figure 6):

- *Text preprocessing*: splitting the original document into sentences;
- *Named entity extraction*, followed by named entity co reference resolution;
- *Triplet extraction* based on the Stanford Parser;
- *Triplet enhancement* by solving pronominal anaphors and assigning for each triplet its WordNet synset;
- *Triplet merger into a semantic graph* of the document.

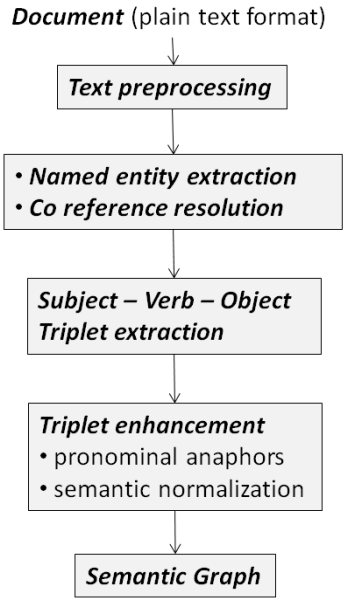

Figure 6. The semantic graph generation pipeline.

In what follows, we are going to further detail the aforementioned pipeline components as proposed in [9].

4.1 Named Entity Extraction

The term *named entities* refers to names of people, locations and organizations, yielding semantic information from the input text. For named entity recognition we consider *GATE*[4] (*General Architecture for Text Engineering*), which was used as a toolkit for natural language processing. For people we also store their gender, whereas for locations we differentiate between names of cities and of countries, respectively. This enables co reference resolution, which implies identifying terms that refer to the same entity. It is achieved through consolidating named entities, using text analysis and matching methods. We match entities where one surface form is completely included in the other (for example *Anna Smith* and *Anna Maria Smith*), one sufrace form is the abbreviation of the other (for example *ISWC* and *International Semantic Web Conference*), or there is a combination of the two situations described above (for example *A. Smith* and *Anna Smith*).

Figure 7 represents an excerpt of a document with two annotated named entities and their corresponding co reference (we eliminate stop words when resolving co references).

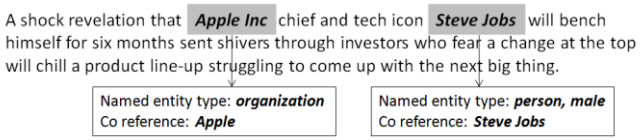

Figure 7. A document excerpt with two annotated named entities (an organization and a person).

4.2 Triplet Extraction

We envisage the "core" of a sentence as a *triplet* consisting of the *subject*, *verb* and *object* elements and assume that it contains enough information to express the message of a sentence. The usefulness of triplets resides in the fact that it is much easier to process them instead of dealing with very complex sentences as a whole.

Triplets are extracted from each sentence independently, without taking text outside of the sentence into account. We apply the algorithm for obtaining triplets from a Penn Treebank parser output described in [10], and employ the statistical *Stanford Parser*[5] as well as the *OpenNLP* parser, in the case of question answering. The extraction is performed based on pure syntactic analysis of sentences. The rules are built by hand, and use the shape of the parse tree to decide which triplets to extract.

Figure 8 shows a triplet (Apple Inc – sent – shivers) extracted from the sentence listed in Figure 7. Aside from the main triplet elements (subject, verb, object), the image also depicts the object attributes (through investors) – these are the words which are linked to the object in the parse tree.

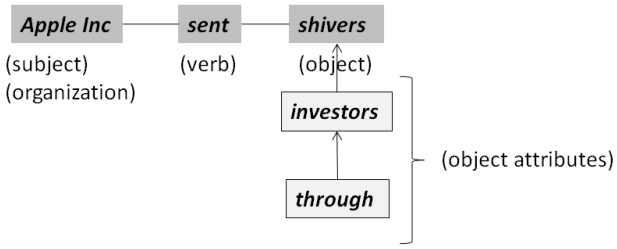

Figure 8. A triplet (Apple Inc - sent - shivers) extracted from the sentence listed in Figure 7.

[4] GATE web page: http://gate.ac.uk/

[5] The Stanford Parser web page: http://nlp.stanford.edu/software/lex-parser.shtml

4.3 Triplet Enhancement and Semantic Graph Generation

The *semantic graph* is utilized in order to represent the document's semantic structure. Our approach is based on the reasearch presented in [8] and further developed in [9]. While in [8] semantic graph generation was relying on the propriatory NLPWin linguistic tool [11] for deep syntactic analysis and pronominal reference resolution, we take advantage of the co referenced named entities as well as the triplets extracted from the Penn Treebank parse tree and derive rules for pronominal anaphora resolution and graph generation.

Triplets are enhanced by first resolving anaphors for a subset of pronouns: {*I, he, she, it, they}*, and their objective, reflexive and possessive forms, as well as the relative pronoun *who*. For solving this task, triplets are linked to their corresponding co referenced named entity (if there exists one). In the previous example, the subject element (Apple Inc) would be linked to the co referenced named entity (Apple). Furthermore, we search throughout the document for possible candidates to replace the pronoun. The candidates receive scores, based on a series of antecedent indicators (or preferences) [9]: *givenness, lexical reiteration, referential distance, indicating verbs* and *collocation pattern preference.*

Secondly, triplets are assigned their corresponding WordNet synset. This is a mandatory step preceding the semantic graph generation, as it enables us to merge triplet elements which belong to the same WordNet synset, and thus share a similar meaning. Hence we augment the compactness of the graphical representation, and enable various triplets to be linked based on a synonymy relationship. We obtain a directed semantic graph, the direction being from the subject node to the object node, and the connecting link is represented by the verb.

Figure 9 presents a semantic sub-graph of a text excerpt.

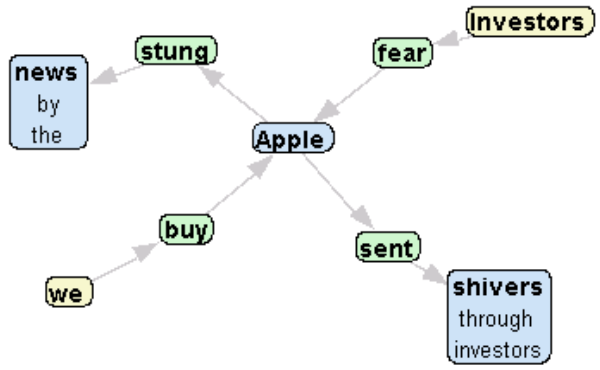

Figure 9. A semantic sub-graph of a text excerpt.

The semantic graph generation system components were evaluated by comparing their output with the one of similar systems [9]. The evaluation was performed on a subset of the Reuters RCV1 [12] data set. For co reference resolution, the comparison was made with GATE's co reference resolver; our co reference module performed about 13% better than GATE. In the case of anaphora resolution, we compared the outcome of our system with two baselines that considered the closest named entity as a pronoun replacement, and one baseline also took gender information into account, whereas the other did not. We obtained good results in the case of the masculine pronoun *he.*

5. DOCUMENT SUMMARIES

The second form of obtaining a document overview is through its associated summary. This is automatically obtained starting from the initial document and its corresponding semantic representation. The document summary will be composed of sentences from the initial text, preserving the sentence ordering. For example, Figure 10 illustrates a three sentence long summary for a Reuters article[6], using our summarization system.

> As such, many fear his departure would deprive the company of its most vital and imaginative resource.
>
> Indeed, Jobs return to Apple a decade ago sparked a period of remarkable growth and creativity, which helped keep Apple one step ahead of rivals in the brutally competitive consumer technology sector.
>
> Jobs said he would step aside until June to deal with his health issues, which turned out to be "more complex than originally thought.

Figure 10. A three sentence long summary for a Reuters article.

The technique involves training a linear SVM classifier to determine those triplets that are useful for extracting sentences which will later compose the summary. The features employed for learning are represented by *linguistic*, *document* and *graph* attributes associated with the triplet elements [9]. The summarization process, described in Figure 11, starts with the original document and its semantic graph. The three types of features abovementioned are then retrieved. Further, the sentences are classified with the linear SVM and the document summary is obtained. Its sentences are labeled with SVM scores and ordered based on these scores in a decreasing manner.

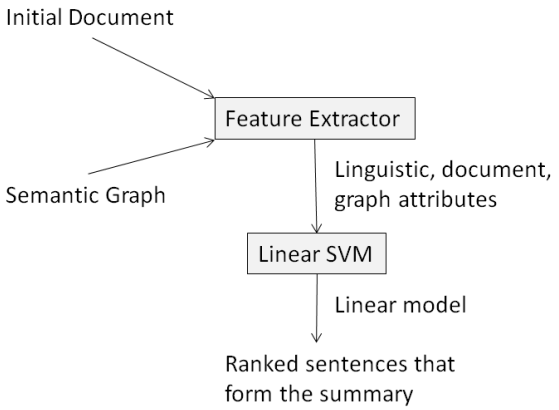

Figure 11. The summarization process.

[6] A link to the Reuters article: http://www.reuters.com/article/technologyNews/idUSTRE50E17320090115 (15/01/2009)

For the evaluation of the document summary, we utilize the *DUC (Document Understanding Conferences)*[7] datasets from 2002 and 2007, respectively, and compare the results with the ones obtained in the 2007 update task [9].

6. CONCLUSIONS AND FUTURE WORK

In this paper we presented an enhanced question answering system where the document that supports the answer can be further described by its visual representation in the form of a semantic graph, by its automatically generated summary and by a list of facts which stand for subject – verb – object triplets. Each of the system components were detailed, starting with the question answering technique which requires questions to follow a predetermined template and searches through a set of facts automatically extracted from a document collection, followed by the semantic graph generation pipeline and concluding with the document summarization process. The questions' parse trees undergo a linguistic analysis to determine the type of the question and to translate it to a query for the triplet search engine. Question answering is not domain specific, so the variety of words to use in the question is not limited. For the linguistic analysis to be successful it is required that the question has a predefined grammatical structure.

Regarding future improvements, we aim at extending the system by adding several components such as another named entity recognizer module, as well as a new triplet extraction module. For further improving the document overview functionality, we intend to integrate external resources that would refine the semantic representation, as well as the document summary. Future improvements to the question answering module could be extending the search to look for answers in ontologies like the ones available in the open linked data. Also question analysis should be improved to relax the requirements that the questions have a predetermined form. Different indexing approaches can be tried to increase the speed of retrieval, and also a key word search which the system can use as a last resort, has to be implemented.

7. ACKNOWLEDGMENTS

This work was supported by the Slovenian Research Agency and the IST Programme of the EC SMART (IST-033917) and PASCAL2 (IST-NoE-216886).

[7] DUC web site: http://duc.nist.gov/

8. REFERENCES

[1] Lopez, V., Uren, V., Motta, E. and Pasin, M. 2007. AquaLog: An ontology-driven question answering system for organizational semantic intranets. Journal of Web Semantics, 72-105, Elsevier.

[2] Damljanovic, D., Tablan, V. and Bontcheva, K. 2008. A text-based query interface to owl ontologies. In the 6th Language Resources and Evaluation Conference (LREC), (Marrakech, Morocco, ELRA, May 2008).

[3] Tablan, V., Damljanovic, D. and Bontcheva, K. 2008. A natural language query interface to structured information. In Proceedings of the 5h European Semantic Web Conference ESWC 2008, (Tenerife, Spain, June, 2008).

[4] Banko, M. and Etzioni, O. 2008. The Tradeoffs Between Open and Traditional Relation Extraction. In Proceedings of the Association for Computational Linguistics, Columbus, Ohio.

[5] Fellbaum, Ch. 1998. WordNet: An Electronic Lexical Database. MIT Press.

[6] Grobelnik, M. and Mladenić, D. 2009. Text Garden. Springer.

[7] Marcus, M. P., Santorini, B.and Marcinkiewicz, M. A. Building a large annotated corpus of English: the Penn Treebank.

[8] Leskovec, J., Grobelnik, M. and Milic-Frayling, N. 2004. Learning Sub-structures of Document Semantic Graphs for Document Summarization. Workshop on Link Analysis and Group Detection (LinkKDD) at KDD 2004 (Seattle, USA, August 22 – 25, 2004).

[9] Rusu, D., Fortuna, B., Grobelnik, M. and Mladenić, D. 2009. Semantic Graphs Derived From Triplets With Application In Document Summarization. Informatica Journal.

[10] Rusu, D., Dali, L., Fortuna, B., Grobelnik, M. and Mladenić, D. 2007. Triplet Extraction from Sentences. In Proceedings of the 10th International Multiconference "Information Society - IS 2007" (Ljubljana, Slovenia, October 8 – 12, 2007). 218 – 222.

[11] Corston-Oliver, S.H. and Dolan, B. 1999. Less is more: eliminating index terms from subordinate clauses. In Proceedings of the 37th Conference on Association for Computational Linguistics, College Park, Maryland.

[12] Lewis, D. D., Yang, Y., Rose, T. G., Li, F. 2004. RCV1: A New Benchmark Collection for Text Categorization Research. 2004, Journal of Machine Learning Research, Vol. 5.