Mining Web Snippets to Answer List Questions

Alejandro Figueroa **Günter Neumann**

Deutsches Forschungszentrum für Künstliche Intelligenz - DFKI,
Stuhlsatzenhausweg 3, D - 66123,
Saarbrücken, Germany
Email: {figueroa|neumann}@dfki.de

Abstract

This paper presents `ListWebQA`, a question answering system that is aimed specifically at extracting answers to list questions exclusively from *web snippets*. Answers are identified in *web snippets* by means of their semantic and syntactic similarities. Initial results show that they are a promising source of answers to list questions.

Keywords: Web Mining, Question Answering, List Questions, Distinct Answers.

1 Introduction

In recent years, search engines have markedly improved their power of indexing, provoked by the sharp increase in the number of documents published on the Internet, in particular, HTML pages. The great success of search engines in linking users to nearly all the sources that satisfy their information needs has caused an explosive growth in their number, and analogously, in their demands for smarter ways of searching and presenting the requested information. Nowadays, one of these increasing demands is finding answers to natural language questions. Most of the research into this area has been carried out under the umbrella of Question Answering Systems (QAS), especially in the context of the Question Answering track of the Text REtrieval Conference (TREC).

In TREC, QAS are encouraged to answer several kinds of questions, whose difficulty has been systematically increasing during the years. In 2001, TREC incorporated *list questions*, such as "*What are 9 novels written by John Updike?*" and "*Name 8 Chuck Berry songs*", into the question answering track. Simply stated, answering this sort of question consists in discovering a set of different answers in only one or across several documents. QAS must therefore, efficiently process a wealth of documents, and identify as well as remove redundant responses in order to satisfactorily answer the question.

Modest results obtained by QAS in TREC show that dealing with this kind of question is particularly difficult (Voorhees 2001, 2002, 2003, 2004), making the research in this area very challenging. Usually, QAS tackle list questions by making use of precompiled, often manually checked, lists (i. e. famous persons and countries) and online encyclopedias, like Wikipedia and Encarta, but with moderate success. Research has been hence conducted towards exploiting full web documents, especially their lists and tables.

This paper presents our research in progress ("*Greenhouse work*") into list question answering on the web. Specifically, it presents `ListWebQA`, our list question answering system that is aimed at extracting answers to list questions directly from the brief descriptions of web-sites returned by search engines, called *web snippets*. `ListWebQA` is an extension of our current web question answering system[1], which is aimed essentially at mining web snippets for discovering answers to natural language questions, including factoid and definition questions (Figueroa and Atkinson 2006, Figueroa and Neumann 2006, 2007).

The motivation behind the use of web snippets as a source of answers is three-fold: (a) to avoid, whenever possible, the costly retrieval and processing of full documents, (b) to the user, web snippets are the first view of the response, thus highlighting answers would make them more informative, and (c) answers taken from snippets can be useful for determining the most promising documents, that is, where most of answers are likely to be. An additional strong motivation is, the absence of answers across retrieved web snippets can force QAS a change in its search strategy or a request for additional feedback from the user. On the whole, exploiting snippets for list question answering is a key research topic of QAS.

The roadmap of this paper is as follows: section 2 deals at greater length with the related work. Section 3 describes `ListWebQA` in detail, section 4 shows current results, and section 5 draws preliminary conclusions.

2 Related Work

In the context of TREC, many methods have been explored by QAS in order to discover answers to list questions across the target collection of documents (the AQUAINT[2] corpus). QAS usually start by distinguishing the "*focus*" of the query, the most descriptive noun phrase of the expected answer type (Katz et al. 2003). The *focus* associates the question with its answer type, and hence answering depends largely upon its correct identification. To illustrate, the *focus* of the query "*Name 6 comets*" is the plural noun "*comets*", and QAS will then only pay attention to names of comets during the search. For the purpose of finding right answers, some QAS take into

The work presented here was partially supported by a research grant from the German Federal Ministry of Education, Science, Research and Technology (BMBF) to the DFKI project `HyLaP` (FKZ: 01 IW F02) and the EC-funded project QALL-ME.

[1] `ListWebQA` is part of our sustained efforts to implement a public TREC-oriented QAS on web snippets. Our system is available at http://experimental-quetal.dfki.de/.

[2] http://www.ldc.upenn.edu/Catalog/byType.jsp

account pre-defined lists of instances of several *foci*. For example, (Katz et al. 2004) accounted for a list of 7800 famous people extracted from biography.com. They additionally increased their 150 pre-defined and manually compiled lists used in TREC 2003, to 3300 in TREC 2004 (Katz et al. 2003). These lists were semi-automatically extracted from WorldBook Encyclopedia articles by searching for hyponomyns. In TREC 2005, (Katz et al. 2005) generated these lists off-line by means of subtitles and link structures provided by Wikipedia. This strategy involved processing a whole document and its related documents. The manual annotation consisted in adding synonymous noun phrases that could be used to ask about the list. Finding answers, consequently, consists in matching elements of these pre-defined lists with a set of retrieved passages. As a result, they found that online resources, such as Wikipedia, slightly improved the recall for the TREC 2003 and 2004 list questions sets, but not for TREC 2005, despite the wide coverage provided by Wikipedia. (Katz et al. 2005) eventually selected the best answer candidates according to a given threshold.

Another common method used by QAS is interpreting a list question as a traditional factoid query and finding its best answers afterwards. In this strategy, low-ranked answers are also cut-off according to a given threshold (Schone et al. 2005). Indeed, widespread techniques for discovering answers to factoid questions based upon redundancy and frequency counting, tend not to work satisfactorily on list questions, because systems must return all different answers, and thus the less frequent answers also count. Some systems are, for this reason, assisted by several deep processing tools such as co-reference resolution. This way, they handle complex noun phrase constructions and relative clauses (Katz et al. 2005). All things considered, QAS are keen on exploiting the massive redundancy of the web, in order to mitigate the lack of redundancy of the AQUAINT corpus, thus increasing the chance of detecting answers, while at the same time, lessening the need for deep processing.

In the context of TREC 2005, (Wu et al. 2005) obtained patterns for detecting answers to list questions by checking the structure of sentences in the ACQUAINT corpus, where previously known answers occurred. They found that the semantics of the lexico-syntactic constructions of these sentences matches the constructions observed by (Hearst 1992) for recognising hyponomic relations. (Hearst 1992) additionally observed that these patterns frequently occur within natural language texts and are triggered by some keywords like "*including*", "*include*", "*such as*" and "*like*". Later, (Sombatsrisomboon et al. 2003) took advantage of the copular pattern "*X is a/an Y*" for acquiring hypernyms and hyponyms for a given lexical term from web snippets, and suggested the use of Hearst's patterns for acquiring additional pairs hypernym–hyponym. However, the main drawback of these patterns is that the contextual lexical dependency can occur between a large span of text.

(Shinzato and Torisawa 2004a) acquired hyponomic relations from full web documents based on the next three assumptions: (a) hyponyms and their hypernym are semantically similar, (b) the hypernym occurs in many documents along with some of its hyponyms, and (c) expressions in a listing are likely to have a common hypernym. Under these assumptions, (Shinzato and Torisawa 2004b) acquired hyponyms for a given hypernym from lists in web documents. The underlying assumption of their strategy is that a list of elements in a web page is likely to contain hyponyms of the hypermyn signalled on the heading of the list. (Shinzato and Torisawa 2004b) ranked hypernym candidates by computing some statistics based on co-occurrence across a set of downloaded documents. They showed that finding the precise correspondence between lists elements and the right hypernym is a difficult task. In addition, many hyponyms or answers to list questions cannot be found in lists or tables, which are not necessarily complete, specially in online encyclopedias. QAS are, therefore forced to search along the whole text or across several documents in order to discover all answers. To illustrate, two good examples in Wikipedia, at the time of writing, are the TREC questions "*Who were 6 actors who have played Tevye in Fiddler on the Roof?*" and "*What are 12 types of clams?*".

(Yang and Chua 2004c) also exploited lists and tables as sources of answers to list questions. They fetched more than 1000 promising web pages by means of a query rewriting strategy that increased the probability of retrieving documents containing answers. This rewriting was based upon the identification of part-of-speech (POS), Name Entities(NEs) and a subject-object representation of the prompted question. Documents are thereafter downloaded and clustered. They also noticed that there is usually a list or table in the web page containing several potential answers. Further, they observed that the title of pages, where answers are, is likely to contain the subject of the relation established by the submitted query. They extracted then answers and projected them on the AQUAINT corpus afterwards. In this method, the corpus acts like a filter for misleading and spurious answers. As a result, they improved the F_1 score of the best TREC 2003 system.

(Cederberg and Windows 2003) distinguished putative pairs hyponomy-hypernym on the British National Corpus, by means of the patterns suggested by (Hearst 1992). They filtered out some spurious relations found by these patterns, by inspecting their degree of relatedness in the semantic space provided by Latent Semantic Analysis (LSA) (Deerwester 1990). They built this semantic space by taking advantage of the representation proposed by (Schütze 1997), and as a result, they showed that it substantially improved the precision of their method. Specifically, (Cederberg and Windows 2003) used the 1000 more frequent content words to build this semantic space and considered the cosine as a measure of similarity. Since a hyponym and its hypernym are expected to share a semantic similarity, the plausibility of a putative hyponomic relationship is given their degree of semantic similarity in this space. Furthermore, (Cederberg and Windows 2003) extended their work by inferring hyponomic relations by means of nouns co-occurring in noun coordinations. As a result, they proved that LSA is an effective filter when combined with patterns and statistical information.

Incidentally, web snippets haven shown to be useful for assisting the extraction of answers to factoid and definition questions (Figueroa and Neumann 2006, 2007). In particular, (Figueroa and Neumann 2007) took descriptive phrases straightforwardly from web snippets by submitting ten query rewritings to a commercial search engine. These rewritings were based largely upon a set of surface patterns, including the copular pattern, that often convey definitions. In this way, they improved the recall of definition utterances in web snippets, and consequently, the probability of aligning these surface patterns with the retrieved web snippets increased.

`ListWebQA`

`ListWebQA` recognises answers to list questions on the grounds that they share a **similar semantic** and **syntactic context**. This is in sharp contrast to current systems that interpret a list question as factoid query or as the matching of pre-defined lists with a set of retrieved paragraphs. In this way, `ListWebQA` attempts to **get rid of** pre-defined lists.

`ListWebQA` distinguishes answers candidates that behave syntactically similar by means of a set of surface patterns at the sentence level, and measures their semantic closeness by means of LSA. `ListWebQA` accounts for the patterns proposed by (Hearst 1992) and (Sombatsrisomboon et al. 2003), and **four extra patterns** that were found to be useful for distinguishing additional answer candidates in web snippets. Further, `ListWebQA` makes use of Google n-grams[3] and coordinations of answers candidates for identifying the most promising answers.

The most essential and interesting facet of `ListWebQA` is that it aims at discovering answers on **web snippets**, instead of full HTML pages, by means of **four purpose-built queries**. These queries are based upon the observation that pages containing answers are very likely to match a noun phrase of the query with their title.

3 Mining Web Snippets for Lists of Answers

The flow of `ListWebQA` is a follows. `ListWebQA` receives a natural language query, Q, as input and performs the following steps:

1. `ListWebQA` analyses Q in order to determine its noun phrases and the *focus* as well as verbs.
2. `ListWebQA` retrieves web snippets that are likely to contain answers by mean of four purpose-built queries.
3. `ListWebQA` discriminates answers candidates in web snippets on the grounds of a set of syntactic patterns.
4. `ListWebQA` ranks answers candidates by means of LSA and their frequency on the web.

Accordingly, each step is described in detail in the following sections.

3.1 Query Analysis

`ListWebQA` starts similarly to (Yang and Chua 2004c), by removing head words from Q. This is a necessary step, because head words have an influence on the posterior processing of Q (Yang and Chua 2004c), and they serve only an essential role for the determination of the type of question. For example, queries like "*What are 9 novels written by John Updike?*" and "*Name 8 Chuck Berry songs*" after head words are removed, remain as "*novels written by John Updike*" and "*Chuck Berry songs*", respectively. From now on, this query without head words is referred to as the prompted question Q.

Next, `ListWebQA` uses part-of-speech (POS) tags[4] for extracting the following information from Q:

- **Verbs** are terms tagged as VBP, VBZ, VBD, VBN, and VB, as well as VBG. For instance, "*written*" in "*novels written by John Updike*". Stop-words[5] (i. e. *do* and *have*) are permanently discarded.
- **Foci** are words or sequences of words tagged as NNS, apart from stop-words. In particular, "*novels*" and "*songs*" in "*novels written by John Updike*" and "*Chuck Berry songs*" respectively. The *focus* signals the expected answer type (EAT), narrowing the search space. In some cases, the *focus* has a complex internal structure, because nouns can occur along with an adjective that plays an essential role in its meaning. A good example is "*navigational satellites*". In this sort of case, the adjective is attached to its corresponding plural noun (NNS).
- **Noun Phrases** are determined by following the next two steps:
  - A sequence of consecutive NNs and NNPs are grouped into one NN and NNP respectively.
  - Any pair of consecutive tags NN - NNS, NNP - NNPS and NNP - NN are grouped into one NNS, NNPS and NNP, respectively. This procedure is applied recursively until no further merge is possible.

  Accordingly, sequences of words labelled as NNPS and NNP are interpreted as noun phrases. This procedure offers some advantages over chunking to the posterior processing, because some noun phrases are not merged, remaining as simpler constituents, helping to fetch some of its common variations. For example, "*Ben and Jerry*" remains as "*Ben*" and "*Jerry*", which helps to match "*Ben & Jerry*". Another vital aspect is, reliable and efficient POS taggers for public use currently exist, contrary to chunkers, which still need improvement.

Additionally, we briefly tried the use of the subject-object representation of sentences, like (Yang and Chua 2004c), provided by MontyLingua[6], but some difficulties were encountered while it was computing the representation of some queries. Furthermore, the Stanford NER[7] was also attempted, but no tangible improvement was noticed.

3.2 Retrieving Web Snippets

(Yang and Chua 2004a,b) observed that web pages that are likely to contain answers to list questions contain a noun phrase of Q in the title, and therefore, they took titles into account for identifying reliable sources of answers. This empirical observation becomes especially relevant when we consider the feature "*intitle*" provided by search engines like Google or MSN Search. This feature assists users in finding web pages where the title matches a given string. Putting both things together, `ListWebQA` makes allowances for this feature to focus the search on pages that are very likely to contain answers. More precisely, `ListWebQA` searches for web pages entitled with NNPSs and NNPs discovered during query analysis. Accordingly, if several noun phrases occur within Q, they are concatenated with the disjunction "*or*". The reason to prefer the disjunction to the conjunction "*and*" is that the conjunction brings about a lower recall. This concatenation is called *a title clause.* Some illustrative *title clauses* are *(intitle:"JOHN UPDIKE")* and *(intitle:"CHUCK BERRY")*, obtained from the queries "*novels written by John Updike*" and "*Chuck Berry songs*", respectively.

[3]http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html

[4]The Stanford POS Tagger available at http://nlp.stanford.edu/software/tagger.shtml was used.

[5]The 319 highly frequent close class forms contained in http://www.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils/stop_words was used.

[6]http://web.media.mit.edu/∼hugo/montylingua/

[7]http://nlp.stanford.edu/software/CRF-NER.shtml

Search engines also provide a special feature for matching words in the body of the documents ("*inbody*" in MSN Search and "*intext*" in Google). ListWebQA takes advantage of this feature to bias the search engine in favour of documents containing the *focus* of *Q*, especially within the snippet text. In the case of queries with several *foci*, they are concatenated with the disjunction "*or*". Since ListWebQA looks for web pages containing both the desired title and body, both clauses are linked with the conjunction "*and*". The following two queries correspond to the illustrative examples "*novels written by John Updike*" and "*Chuck Berry songs*":

- (intitle:"JOHN UPDIKE") AND (inbody:"NOVELS" OR inbody:"WRITTEN")
- (intitle:"CHUCK BERRY") AND (inbody:"SONGS")

The first generated query unveils another key aspect of our web search strategy: query verbs are also added to the *body clause*. Specifically, some samples of retrieved snippets by these two queries are:

- **Chuck Berry** - You Never Can Tell — videos.superheldenclub.de — USER ...
  Find out the **songs** release date (Wikipedia , Google) 3. Grab the YouTube-URL, push the button, fill ... Now Popular. **Chuck Berry** - You Never Can Tell; Artists ...
- IMS: **John Updike**, HarperAudio
  Author and poet **John Updike** reads excerpts from his short story "The Persistence of Desire". ... **Updike**'s other published works include the **novels** "Rabbit Run", "Couples", and "The Witches of ...)

The second snippet shows three vital aspects of the recognition of answers to list questions within snippets: (a) a list of answers can be signalled by a coordination of elements, (b) this list can be indicated by some lexico-syntactic patterns, and (c) due to the size of the snippets, this coordination is normally truncated. Therefore, every time ListWebQA detects a sentence that fulfils these three conditions, it submits the truncated sentence to the search engine (in quotes) and replaces the old with the newly fetched one. In the example, the new sentence is as follows:

- **Updike**'s other published works include the **novels** "Rabbit Run", "Couples", and "The Witches of Eastwick". This recording was made in 1969. .au format (4 Mb), .gsm format (0.8 Mb), .ra format (0.5 Mb

Certainly, the TREC list question sets have questions that do not contain any NNPS or NNP, especially "*Name 6 comets*" and "*What are 6 names of navigational satellites?*". This sort of question only provides a *body clause*. To neatly illustrate, the queries are as follows:

- (inbody:"COMETS")
- (inbody:"NAVIGATIONAL SATELLITES")

In fact, ListWebQA prefers not to add NNSs to the "*title clause*", because they lead the search to unrelated topics. This is seen as a consequence of the semantic/syntactic flexibility of some NN/NNS, especially to form compounds. For example, pages concerning the sport team "*Houston Comets*" are retrieved while searching for "*intitle:comets*", which is a compound likely to occur in the title of a web page.

From this first purpose-built query, ListWebQA derives the second and third queries. Following the observation that sometimes answers are likely to be signalled by some hyponomic words like "*such as*", "*include*", "*including*" and "*include*". ListWebQA appends these words to the *focus* as follows:

- (intitle:"JOHN UPDIKE") AND (inbody:"NOVELS LIKE" OR inbody:"NOVELS INCLUDING") AND (inbody:"WRITTEN")
- (intitle:"JOHN UPDIKE") AND (inbody:"NOVELS SUCH AS" OR inbody:"NOVELS INCLUDE") AND (inbody:"WRITTEN")
- (intitle:"CHUCK BERRY") AND (inbody:"SONGS LIKE" OR inbody:"SONGS INCLUDING")
- (intitle:"CHUCK BERRY") AND (inbody:"SONGS SUCH AS" OR inbody:"SONGS INCLUDE")
- (inbody:"NAVIGATIONAL SATELLITES LIKE" OR inbody:"NAVIGATIONAL SATELLITES INCLUDING")
- (inbody:"NAVIGATIONAL SATELLITES SUCH AS" OR inbody:"NAVIGATIONAL SATELLITES INCLUDE")

Two queries are generated from these keywords because of the query limit imposed by search engines (150 characters). It is also worth pointing out, that unlike the first query, verbs are concatenated in another *body clause*. In brief, these two purpose-built queries bias search engines in favour of snippets that are very likely to contain coordinations with answers. In particular, these queries above provide the next two snippets:

- www.heritage.org
  With necessary missile guidance modifications, midcourse correction could be provided for hydralaunch MXs through data transmission from military **navigational satellites such as** GPS and Navstar.
- Amazon.com: **Chuck Berry** Is on Top: Music: **Chuck Berry**
  Chuck Berry's genius is in full bloom on practically every song here: **Songs like** "Maybelline", "Roll Over Beethoven", "Around and Around", "Carol", and "Little Queenie" are, like any self-respecting ...

In addition, ListWebQA generates an extra query aimed specifically at exploiting the content of on-line encyclopedias. To achieve this, ListWebQA takes advantage of the feature "*site*" provided by search engines to crawl in Wikipedia and Answers.com. In our working examples, this fourth query looks like as follows:

- (inbody:"NAVIGATIONAL SATELLITES") AND (site:en.wikipedia.org OR site:www.answers.com)
- (intitle:"JOHN UPDIKE") AND (inbody:"NOVELS" OR inbody:"WRITTEN") AND (site:en.wikipedia.org OR site:www.answers.com)

In particular, two retrieved snippets by these two queries are:

- John Updike - Wikipedia, the free encyclopedia
  ... is well known for his careful craftsmanship and prolific writing, having published 22 novels ... The book's title is "YOUR SHOES TOO BIG TO KICKBOX GOD" which is 20 page book written by John Updike as a ...
- GPS: Information from Answers.com
  GPS Global Positioning System (GPS) is a navigation system consisting of a constellation of 24 navigational satellites orbiting Earth, launched and

The second snippet stresses how this query strategy exploits the indexing power of search engines. Many answers occur in many documents belonging to on-line encyclopedias, which are not straightforwardly reachable by matching query with topic-document keywords. This sort of document usually contains a paragraph or a couple of sentences relevant to the query, and hence, in order to find this piece of text, it is necessary to download, process the entire topic-related document, and what is more, some of its related documents. In the example, the answer "*GPS*" is contained in the body of a document related to "*navigational satellites*" titled with the answer. ListWebQA retrieves the relevant sentences without downloading and processing this document. Furthermore, it does not need to follow any document structure or linkage to discover the answer. Lastly, it is also worth highlighting that each submission retrieves the first ten snippets.

A final remark regarding the query construction is, words like "*people*", "*names*", "*U.S.*" are not considered in the title, because it was found that they usually bias the search engine to unrelated topics, probably due to the fact that they are frequently in titles of web pages, and therefore they occur in several contexts.

Pre-processing

Once all snippets are retrieved, ListWebQA interprets intentional breaks as sentence endings. The identified pieces of text are processed with JavaRap[8] afterwards, in order to identify sentences within snippets. If a sentence contains an unfinished list of items triggered by a hyponomic keyword, ListWebQA attempts to retrieve the missing part of the list by submitting the known part of the sentence to the search engine. If a more complete snippet is found, it is accordingly extended. Sentences are also identified in these fetched extensions.

The next step is replacing all instances of all query verbs with a place holder. Here, ListWebQA considers also morphological variations of verbs, in particular, the words "*write*", "*writing*", "*written*" are mapped to the same place holder "*qverb0*", where the zero indexes the respecting verb within Q. ListWebQA then does similar processing with *foci* in Q. In this case, plural and singular forms are mapped to the same place holder; that is "*novel*" and "*novels*" are mapped to "*qfocus0*", where "*0*" is accordingly the corresponding index to the *focus* in the query. Consequently, ListWebQA follows the same strategy for noun phrases within the query, but ListWebQA accounts for some of their variations. In this step, ListWebQA searches for substrings contained in the noun phrases of the query and if the ratio of their frequency is lower than 1.75, both are mapped to the same place holder "*qentity*". In our two working snippets concerning "*John Updike*", "*Updike*" and "*John Updike*" are accordingly mapped to "*qentity0*" as follows:

- **qentity0** - Wikipedia, the free encyclopedia
  ... is well known for his careful craftsmanship and prolific **qverb**, having published 22 **qfocus0** ... The book's title is "YOUR SHOES TOO BIG TO KICKBOX GOD" which is 20 page book **qverb** by **qentity0** as a ...
- IMS: **qentity0**, HarperAudio
  Author and poet **qentity0** reads excerpts from his short story "The Persistence of Desire". ... **qentity0**'s other published works include the **qfocus0** "Rabbit Run", "Couples", and "The Witches of Eastwick." This recording was made in 1969. .au format (4 Mb), .gsm format (0.8 Mb), .ra format (0.5 Mb

The first snippet emphasises an additional significant aspect, if ListWebQA discovers a noun like "*writing*", which is a variation of the verb "*write*", it is also mapped to "*qverb0*". This helps ListWebQA to detect some close paraphrases. The next step, entity recognition, is discussed in the next section.

[8]http://www.comp.nus.edu.sg/∼ qiul/NLPTools/JavaRAP.html.

3.3 Recognising Entities in Web Snippets

One of the major problems of list questions is that the type of the *focus* varies widely from question to question. For instance, the query "*Name 10 countries that produce peanuts*" has countries (locations) as *foci*, but the question "*What are 9 novels written by John Updike?*" names of books. This variation plays a crucial role in determining answers, because state-of-the-art NER do not recognise all types of *foci.* Specifically, Stanford's NER identifies person names, organisations and locations, which are useful, but provide a low coverage for the wide scope of types occurring in list questions. Additionally, the performance of NERs is directly affected by truncations on web snippets. For these reasons, ListWebQA mainly distinguishes entities by means of two regular expressions grounded on sequences of capital letters surrounded by stop-words and punctuation:

1. (#|S|L|P)((N|)(C+)(S{0,3})(C+)(|N))(L|S|P|#)
2. (S|L|P)C(L|S|P)

where "*S*", "*P*", "*N*" stand for a stop-word, a punctuation sign and a number respectively. "*C*" stands for a capitalised word, "*L*" for a lowercased word, and eventually, "*#*" marks a sentence limit. The first pattern is aimed at names of persons, novels, books, places and songs such as "*You Never Can Tell*". The second pattern is aimed at a single isolated word which starts with a capital letter (i. e. cities or country names). The entities recognised for our working snippets are:

- You Never Can Tell, USER, Find, Wikipedia , Google, Grab the YouTube-URL, Now Popular, You Never Can Tell, Artists.
- IMS, HarperAudio, Author, The Persistence of Desire, Rabbit Run, Couples, The Witches of Eastwick.

Since the generalisation process given by these regular expressions causes too much noise. ListWebQA filters out some misleading and spurious entities by removing entities whose frequencies are greater than a frequency threshold determined by Google n-grams counts. In order to avoid discarding some possible answers, we manually checked high-frequent Google n-grams referring to country names like "*United States*" and "*Germany*", and organisations or person names such as "*George Bush*" and "*Jim Clark*". In our illustrative snippets, this step assists ListWebQA in reducing the list of candidates to:

- You Never Can Tell, Grab the YouTube-URL, Now Popular, Artists.
- IMS, HarperAudio, The Persistence of Desire, Rabbit Run, Couples, The Witches of Eastwick.

Then, ListWebQA maps every entity to a place holder "*entity*". In the working example, the snippets[9] remain as follows:

- **qentity0 - entity0** — videos.superheldenclub.de — USER ...
  Find out the **qfocus0** date (Wikipedia , Google) 3. **entity1**, push the button, fill ... **entity2**. **qentity0 - entity0**; **entity3** ...
- **entity0**: **qentity0**, **entity1**
  Author and poet **qentity0** reads excerpts from his short story "**entity2**". ... **qentity0**'s other published works include the **qfocus0** "**entity3**", "**entity4**", and "**entity5**." This recording was made in 1969. .au format (4 Mb), .gsm format (0.8 Mb), .ra format (0.5 Mb

This snippet representation eases the next step; the application of patterns for distinguishing promising answer candidates.

[9]The indexes correspond to the order in the filtered list of entities.

3.4 Answer Candidates

`ListWebQA` identifies answers candidates by means of the following lexico-syntactical patterns:

- **Hyponomic keyword pattern** (`Hyp-P`) is aimed at discriminating answers that co-occur along with the hyponomic keywords found by (Hearst 1992): "*such as*", "*like*" and "*include*" as well as "*including*". This pattern sees every element "*entity*" in the coordination, yielded by these keywords, as an answer candidate. `ListWebQA` attempts to ensure the proper semantic context by checking that a "*qfocus*" in the same sentence exists. In our illustrative example, the sentence "***qentity0**'s other published works include the **qfocus0** "**entity3**", "**entity4**", and "**entity5**.*" provides the answers candidates: "*Rabbit Run*","*Couples*", "*The Witches of Eastwick*" as well.

- **Copular pattern** (`Cop-P`) follows the work of (Sombatsrisomboon et al. 2003) and is aimed at distinguishing answers expressed definitionally by means of copular patterns:

  1. **entity** is \w+ **qfocus** \w*
  2. (**entity**,)+ and **entity** are \w+ **qfocus** \w*

  In particular, this pattern assists `ListWebQA` to detect the answer "*Chubby Hubby*" in "*Chubby Hubby is an original flavour of the ice cream pints created and manufactured by the famous Ben and Jerry's ice cream brand.*".

In addition, the following patterns was also observed to convey answers to list questions in web snippets:

- **POS pattern** (`Pos-P`) identifies answers expressed as possessives according to the following pattern:

  1. **qentity**'s **entity**
  2. **qentity**'s (**entity**,)+ (and|or) **entity**.

  For example: "*John Updike's Buchanan Dying*" and "*Frank Lloyd Wright's Duncan House or The Balter House*".

- **Quotes pattern** (`Quo-P`) recognises answer candidates conveyed in quotations. For instance, the sentence "*Author and poet **John Updike** reads excerpts from his short story 'The Persistence of Desire'*" yields the answer "*The Persistence of Desire*".

- **Qverb pattern** (`Qv-P`) discovers answer candidates yielded by some close paraphrases of the query. `ListWebQA` accounts for paraphrases caused by query-words permutations and local word insertions:

  1. (**qentity**|pronoun|**qfocus**) \w{0,3} **qverb** \w{0,3} **entity**
  2. **entity** \w{0,3} **qverb** \w{0,3} prep \w{0,3} **qentity**

  In the last case, "*prep*" indicates the insertion of a preposition. This pattern can find the answer "*Poland*" in the sentence "*Pope John Paul II visited his native Poland*" to the question "*Name 32 countries Pope John Paul II has visited.*", whereas the first pattern discovers "*Hollyhock House*" in "*Hollyhock House designed by Frank Lloyd Wright*". One last remark is, entities, pronouns, prepositions, query verbs and words can only be separated for three words at most.

- **Punctuation pattern** (`Pun-P`) discriminates answer candidates in the title of snippets on the ground of colons. Any bracketed content is removed:

  1. :**qentity**:(\w+:){0,1}**entity**
  2. :**entity**:(\w+:){0,1}**qentity**

  These patterns discover answers in context such as "*Terrorist*" in "*Amazon.com:Terrorist:Books:John Updike*".

In our working examples, this step filters out spurious entities like "*Grab the YouTube-URL*", "*Now Popular*", "*Artists*" and "*IMS*" as well as "*Harper-Audio*". Unfortunately, the song "*You Never Can Tell*" does not match any pattern, and for this reason, `ListWebQA` misses this answer. The two underlying assumptions behind the use of this restricted set of patterns are: (a) correct answers are more likely than spurious answers to occur in several syntactic contexts, and (b) they do not provide a full coverage, but wide enough to ensure a good performance. Since patterns do not provide an unerring accuracy, `ListWebQA` ranks identified answers candidates afterwards.

3.5 Ranking Answer candidates

Let W be the set of augmented answer candidates, that is the set of all answers candidates A augmented with the set of *foci* F and query entities E. `ListWebQA` builds an augmented answer candidates-snippets matrix M, where each cell M_{ij} is one if the element $W_i \in W$ is in the snippet S_j, otherwise zero. The next figure sketches this augmentation:

$$M = \left(\begin{array}{cccc} & S_1 & \dots & S_N \\ A_1 & 0 & \dots & 1 \\ A_2 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ F_1 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ F_\phi & 1 & \dots & 1 \\ E_1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ E_\epsilon & 0 & \dots & 1 \end{array}\right) \quad M^* = \left(\begin{array}{rcc} & S_1 & S_2 \\ entity2 & 1 & 0 \\ entity3 & 0 & 1 \\ entity4 & 0 & 1 \\ entity5 & 0 & 1 \\ qfocus0 & 1 & 1 \\ qentity0 & 1 & 1 \end{array}\right)$$

Where $F_f \in F$, $E_e \in E$ and $A_a \in A$. N is the number of fetched snippets, and the Greek letters ϕ and ϵ stand for the number of *foci* and query entities, respectively. Accordingly, the matrix MM^T captures the semantic relatedness between answer candidates, query foci and entities through the use of information regarding their co-occurrence across web snippets. It is worth remarking that M makes allowances for all instances of answers candidates, not only ones included in syntactical contexts covered by the patterns shown in section 3.3. It provides therefore a clearer view of their semantic relatedness.

Next, `ListWebQA` rewrites M as UDV^T using the *Singular Value Decomposition* (SVD). In this rewriting, the semantic closeness of two elements $W_i, W_j \in W$ is given by the cosine of the angle between their corresponding vectors in U (Bellegarda 2000). `ListWebQA` constructs a smoothed semantic representation of this space by keeping the $k = 3$ vectors that explain the most amount of variance in the data. (Landauer et al. 1998) hypothesised that this sort of smoothing will cause better inferences than the original data-set (i. e. MM^T), when words are selected from paragraphs containing similar words, like web snippets. In this way, `ListWebQA` attempts to infers true patterns of word co-occurence and relations, even tough words do not directly co-occur across web snippets, this means a value in the matrix MM^T equal to zero. This new semantic relatedness is consequently given by $R = \hat{U}\hat{D}^2\hat{U}^T$, where $\hat{D}$, $\hat{U}$ and $\hat{V}$ correspond to the columns of these k vectors in D, U and V, respectively.

`ListWebQA` prefers the dot product $\hat{U}\hat{D}^2\hat{U}^T$ to the traditional cosine as a measure of the semantic relatedness. The major reasons are: (a) it was observed experimentally that, because of the size of web snippets (texts shorter than 200 words), the cosine draws an unclear distinction of the semantic neighbourhood, bringing about spurious inferences (Wiemer-Hastings and Zipitria 2001), and (b) the length of vectors was found to draw a clearer distinction of the semantic neighbourhood, as this biases R in favour of contextual terms, which LSA knows better (Deerwester 1990).

`ListWebQA` ranks two elements $W_i, W_j \in W$ according to their semantic closeness $R(W_i, W_j) = \hat{u_i}\hat{D}^2\hat{u_j}'$ $(\hat{u_i}, \hat{u_j} \in \hat{U})$. In this semantic space, the semantic relation is defined for pairs of elements. `ListWebQA` must therefore check the relatedness of every $A_a \in A$ to every $F_f \in F$ and $E_f \in E$. Then, `ListWebQA` selects for each $A_a \in A$ its maximum value $R_{max}(A_a, Q) = \max_{W_* \in F \cup E} R(A_a, W_*)$. For instance, this method ranks "*John Updike's novels*" candidates as follows:

Table 1: $R_{max}(A_a, Q)$ for "*John Updike's novels*".

Rank	Answer Candidate	Status
32	Rabbit Angstrom	-
31	Eastwick	-
30	Rabbit Redux	+
29	Reviewing 101	-
28	National Book Award	-
27	Rabbit Run	+
26	The Poorhouse Fair	+
25	Lilies	*
24	Winner	-
23	Don Swaim	-
22	See also Donald Greiner	-
21	Self-Consciousness	+
20	Winner of the Pulitzer Prize	-
19	Seek My Face	+
18	Poorhouse Fair	*
17	In the Beauty	*
16	Brazil	+
15	1966 Run Time	-
14	Rabbit Is Rich	+
13	Language Literature Movie Type	-
12	30 minutes	-
11	In the Beauty of the Lilies	+
10	The Centaur	+
9	The Witches of Eastwick	+
8	Terrorist	+
7	YOUR SHOES TOO BIG TO KICKBOX GOD	+
6	Couples	+
5	Rabbit At Rest	+
4	Biography Release Year	-
3	CRITICAL MASS	-
2	Roger	-
1	Picked Up Pieces	+

In table 1, the best ranked answer candidate is the book "*Picked Up Pieces*", and STATUS signals whether the answer candidate is an exact answer ("+"), inexact answer ("*") or wrong answer ("-"). Looking closer upon table 1, it can be seen that some relevant answers (best-selling novels) such as "*Rabbit Run*" and "*Rabbit Redux*" are low-ranked. This is interpreted as a result of a lack of co-occurence information across fetched snippets to draw proper inferences, even though these answers have a high frequency on the web. In this case, Google bi-grams provides a frequency count of 37419 and 3966 for these two novels, respectively. However, a straightforward frequency count will not draw reasonable inferences, because some answers candidates, like "*Lilies*", often occur in several contexts, and hence, they have a high frequency count. Therefore, `ListWebQA` necessarily needs to count frequencies in contexts closer to Q. For this purpose, `ListWebQA` makes use of the next strategy to identify contextual Google n-grams:

1. Trims query entities by leaving the last two words. For example: "*Frank Lloyd Wright*" remains as "*Lloyd Wright*", whereas "*John Updike*" is not trimmed at all.

2. Appends punctuation signs to these trimmed query entities, in such a way that match patterns shown in section 3.4:

   - Lloyd Wright (’s|:|’|”)
   - John Updike (’s|:|’|”)

   `ListWebQA` then searches for Google 4-grams and 5-grams that match these patterns. In case of uni-grams answer candidates, Google 3-grams are also considered in the search. To illustrate, table 2 shows some Google 5-grams with respect to the query entity "*John Updike*".

3. Partially aligns the beginning of each answer candidate with the context yielded by every (obtained) Google n-grams. `ListWebQA` consequently assigns a new rank value to every aligned answer candidate A_a according to:

$$R_{max}(A_a, Q) + 2 * R^+_{max}$$

   where R^+_{max} is the value of the rank, supplied by $R_{max}(A_a, Q)$, of highest answer candidate.

This reranking locates more reliable answers in the top of the rank, sorted by their prior positions. For the working "*John Updike's novels*", some reranked novels are "*The Witches of Eastwick*", "*Seek my face*" and "*Rabbit Redux*" (matches Google 4-grams). In case of questions with no query entity, trimmed *foci* are used for the search, along with the hyponomic keywords of `Hyp-P`:

- **qfocus** (like|include|including|such)

Table 2: An excerpt from 5-grams of "*John Updike*".

1	2	3	4	5	frequency
John	Updike	”	Long	term	456
John	Updike	”	The	essence	42
John	Updike	’s	”	Rabbit	175
John	Updike	’s	”	Separating	46
John	Updike	’s	”	The	94
John	Updike	’s	Licks	of	57
John	Updike	’s	Rabbit	,	145
John	Updike	’s	Rabbit	Angstrom	70
John	Updike	’s	Rabbit	Tetralogy	65
John	Updike	’s	Rabbit	at	76
John	Updike	’s	Rabbit	novels	78
John	Updike	’s	Roger	’s	48
John	Updike	’s	Seek	My	44
John	Updike	’s	The	Witches	40
John	Updike	’s	review	of	78
John	Updike	:	The	Coup	73
John	Updike	:	The	Early	1858
John	Updike	:	The	Witches	989
John	Updike	:	Towards	the	45
John	Updike	:	Villages	</S>	307

ListWebQA, like (Cederberg and Windows 2003), infers additional reliable answers by means of coordinations of answer candidates. ListWebQA inspects whether or not any list of quoted answer candidates or any list signalled by an hyponomic keyword contains a reliable answer candidate. If any list exists, ListWebQA interprets its remaining answer candidates as inferred answers. To illustrate, the novels "*The Witches of Eastwick*" and "*Rabbit Redux*" would assist ListWebQA to infer "*Couples*"[10] as a reliable answer from the sentence "***qentity0**'s other published works include the **qfocus0** 'Rabbit Run', 'Couples', and 'The Witches of Eastwick'*". ListWebQA thus assigns them new rank values according to:

$$R_{max}(A_a, Q) + R^{+}_{max}$$

In this way, these inferred answers are located between the reliable and the remaining answers, sorted by their prior positions. The final ranking for "*John Updike's novels*" is shown in table 3.

Table 3: Final ranking for "*John Updike's novels*".

Rank	Answer Candidate	Status
32	Reviewing 101	-
31	Winner	-
30	Don Swaim	-
29	See also Donald Greiner	-
28	Self-Consciousness	+
27	Winner of the Pulitzer Prize	-
26	Poorhouse Fair	-
25	1966 Run Time	-
24	Language Literature Movie Type	-
23	30 minutes	-
22	In the Beauty of the Lilies	+
21	YOUR SHOES TOO BIG TO KICKBOX GOD	+
20	Biography Release Year	-
19	CRITICAL MASS	-
18	Picked Up Pieces	+
17	Eastwick	*
16	National Book Award	-
15	Lilies	*
14	In the Beauty	*
13	Brazil	+
12	Terrorist	+
11	Rabbit Angstrom	-
10	Rabbit Redux	+
9	Rabbit Run	+
8	The Poorhouse Fair	+
7	Seek My Face	+
6	Rabbit Is Rich	+
5	The Centaur	+
4	The Witches of Eastwick	+
3	Couples	+
2	Rabbit At Rest	+
1	Roger	*

4 Evaluation

List questions started to catch the attention of TREC in 2001, and thus a question set has been provided yearly. With the purpose of assessing our work in progress, we used the standard list question sets supplied by TREC for the years 2001 and 2002.

We carried out separate evaluations from measuring different facets of ListWebQA: answer recall and precision in recognising answers as well as ranking. These evaluations are described in the next sections.

[10]In fact, the novel "*Couples*" also matches Google 4-grams.

4.1 Answer Recall

ListWebQA increases the recall of answers by retrieving forty snippets from the web[11]. This retrieval is done by means of the four purpose-built queries presented in section 3.2. In order to assess our improvement, we implemented B-I, a baseline that, like ListWebQA, fetches forty snippets by submitting Q to the search engine. Table 4 highlights accordingly our achievements for the 25 questions in the TREC 2001 and 2002.

Table 4: TREC 2001-2002 Results (Recall).

	TREC 2001			**TREC 2002**		
Q_{id}	TREC	B-I	ListWebQA	TREC	B-I	ListWebQA
1	10	9	13	4	2	3
2	17	9	17	24	10	6
3	12	9	20	13	2	3
4	14	8	7	9	6	9
5	11	3	16	7	5	5
6	39	6	6	23	3	15
7	39	5	19	21	0	1
8	7	4	2	17	0	12
9	4	1	1	17	0	0
10	6	3	6	7	3	7
11	4	9	8	30	0	0
12	3	0	0	10	5	8
13	9	6	10	10	4	1
14	8	1	5	14	3	12
15	15	14	26	13	3	24
16	22	7	14	8	1	3
17	21	5	23	12	2	4
18	5	4	5	22	2	7
19	7	3	9	12	5	21
20	5	3	1	3	2	4
21	32	1	0	23	6	12
22	15	6	8	3	0	0
23	17	4	4	9	3	2
24	5	3	5	12	0	0
25	23	0	0	10	6	9

In table 4, the column TREC signals the number of answers provided by TREC gold standards and the columns B-I and ListWebQA indicate the number of answers manually found on web snippets retrieved by the respective system. This manual inspection is necessary, because fetched snippets do not necessarily contain all answers supplied by TREC gold standards. In fact, this is a demanding task, because many names must be carefully checked on the web.

ListWebQA did not retrieve any answers to seven questions (14%), and in six out of these seven cases, B-I also could not fetch any answers. In addition, B-I retrieved more answers to eight (16%) questions than ListWebQA, where the larger difference arises from the second question in TREC 2002: "*cities that have a subway system*". In this case, ListWebQA searched for pages containing "*subway system*" in the title, but many answers occured only along with the word "*subway*" in the title, in particular, "*Tokyo Subway*" and "*Moscow Subway*". ListWebQA could not, for this reason, retrieve these snippets.

ListWebQA outperformed B-I in 32 (64%) questions, and fetched more answers than supplied by TREC in seven (14%) cases. The major difference exists in the fifteenth question of TREC 2002 "*works by Edgar Allan Poe*", the 24 retrieved answers are:

```
A Decent Into The Maelstron
A Tale Of The Ragged Mountains
An Acrostic
Annabel Lee
Ligeia
Mesmeric Revelation
```

[11]For our all experiments, we used MSN Search: http://www.live.com/

```
Morella
The Black Cat
The Cask of Amontillado
The Devil in the Belfry
The Domain of Arnheim
The Fall Of The House of Usher
The Man of the Crowd
The Murders in the Rue Morgue
The Pit and the Pendulum
The Purloined Letter
The Raven & Ulalume
The Tales of Edgar
The Tell-Tale Heart
The Thousand-and-Second Tale of Scheherezade
The Valley of Unrest
Three Sundays in a week
Von Kempelen and his Discovery
William Wilson
```

`ListWebQA` and `B-I` retrieved the same number of answers for ten (20%) of the questions. Nevertheless, it is worth stressing that, both sets of answers differ radically. For example, the three Edgar Allan Poe's works retrieved by `B-I` are "*Annabel Lee*", "*Landor's Cottage*" and "*The Haunted Palace*". In this case, neither the TREC gold standard and the output of `ListWebQA` contain these two works. It was computed, therefore, the ratio of different answers in both snippets sets to the number of answers in the fetched snippets. In this illustrative example, this ratio is $\frac{24+3-1}{24+3} = 0.96$ (see table 4), because only one answer is contained in both sets ("*Annabel Lee*"). Overall, an average of 0.822 and a standard deviation of 0.15 was obtained. To sum this up, `ListWebQA` retrieves a set of snippets with more answers, and we hypothesise that both strategies can be combined to achieve a higher recall of answers.

One last remark on answer recall; both systems could not fetch any answers to the eleventh question of TREC 2002 "*musical compositions by Aaron Copland*". This case was inspected separately, and subsequently queries like "*composed by Aaron Copland*" were found to be more adequate to obtain a higher recall of answers. On the one hand, this sort of query rewriting offers the advantage of retrieving contexts that would match the pattern `Qv-P`. On the other hand, this rewriting involves finding the right preposition. In this particular case, Google n-grams or a collocation dictionary would be helpful.

4.2 Precision in Answer Recognition

`ListWebQA` distinguishes answers by means of the patterns discussed in section 3.4. Table 5 shows the current achievements. In this table, QUESTION COVERAGE indicates the number of questions, for which the respective pattern supplied at least one correct answer. On the one hand, `Hyp-P` gives the wider coverage, supplying answers to 38 questions, on the other hand, it provides many wrong answers (low accuracy). One reason for this low precision is uncovered by the question "*countries other than the United States have a vehicle emission inspection program*" and the following fetched snippet:

- February 16, 2005: China Replacing the United States as World's ...
  CHINA REPLACING THE UNITED STATES AS WORLD'S LEADING CONSUMER Lester R. Brown ... Strategic relationships with resource-rich countries such as Brazil, Kazakhstan, Russia, Indonesia ...

This snippet matches `Hyp-P`, and its title contains the noun phrase "*United States*", but its topic is unrelated to "*vehicle emission inspection programs*". Consequently, matching this pattern brings about four wrong answers (according to TREC gold standards). By the same token, matching pre-defined lists with fetched paragraphs suffers the same drawback.

Table 5: Patterns Coverage/Accuracy (Precision).

	TREC 2001		
Pattern	Question Coverage	Recognised Answer Candidates	Accuracy
`Hyp-P`	18	349	0.54
`Cop-P`	6	17	0.35
`Pos-P`	7	59	0.47
`Quo-P`	3	50	0.56
`Qv-P`	6	34	0.68
`Pun-P`	6	45	0.2
	TREC 2002		
Pattern	Question Coverage	Recognised Answer Candidates	Accuracy
`Hyp-P`	20	426	0.19
`Cop-P`	2	21	0.19
`Pos-P`	6	29	0.41
`Quo-P`	4	37	0.40
`Qv-P`	3	21	0.19
`Pun-P`	4	49	0.33

Additionally, different spellings are likely to significantly affect the recognition of answers. For example, `ListWebQA` retrieved three different spellings for the Chuck Berry's song "*Maybelline*" (also found as "*Maybellene*" and "*Maybeline*"). Further, `ListWebQA` finds inexact or incomplete answers. For instance John Updike's novel "*In the beauty of the Lilies*" is also found as "*In the Beauty*" and "*Lilies*". Furthermore, these incomplete answers can be ranked higher than their respective exact answers (see table 3).

Table 6: TREC 2001-2002 Results (Precision).

	TREC 2001			**TREC 2002**		
Q	Answer Recall	Recognised Answers	Total	Answer Recall	Recognised Answers	Total
1	13	13	16	3	3	8
2	17	10	38	6	0	1
3	20	17	61	3	3	26
4	7	6	32	9	7	21
5	16	13	33	5	3	8
6	6	5	14	15	8	35
7	19	15	45	1	0	10
8	2	2	29	12	6	19
9	1	1	21	0	0	0
10	16	4	22	7	2	11
11	8	7	0	0	0	2
12	0	0	11	8	7	24
13	10	9	34	1	0	13
14	5	3	12	12	5	27
15	26	9	26	24	15	53
16	14	10	32	3	0	1
17	23	21	35	4	4	33
18	5	5	9	7	5	45
19	9	9	21	21	11	23
20	1	0	17	4	4	57
21	0	0	18	12	3	32
22	8	7	22	0	0	27
23	4	1	19	2	1	93
24	5	2	35	0	0	29
25	0	0	27	9	6	56

Table 6 highlights the number of recognised answers per question. Overall, `ListWebQA` identified 65% of the right answers. The lower performance is due to the 19th TREC 2002 question "*Name 21 Godzilla movies*". Here, `ListWebQA` could not recognise the right answer candidates, because of the fact that the two regular expressions in section 3.3 do not cover entities like "*Godzilla vs. the Cosmic Monster*". In five cases, `ListWebQA` could not extract any answers, where at least one existed. In particular, ListWebQA did not distinguish any of the six answers (“*New York*”, “*Boston*”, “*London*”, “*Cincinnati*”, “*Rochester*” and “*Seoul*”) to the second question of TREC 2002 “*cities that have a subway system*”. These answers were undetected because of the fact that patterns in section 3.4 do not cover local contexts, such as “*The Cincinnati Subway System*” and “*Map of Rochester's subway system*”.

Table 7: TREC 2001-2002 Final Results (Accuracy).

	2001	2002
ListWebQA	.63/.75	.36/.53
Top one	0.76	0.65
Top two	0.45	0.15
Top three	0.34	0.11

In TREC 2001 and 2002, the measure of performance was accuracy, which was computed as the number of distinct instances returned by the system, divided by the target number of instances (Voorhees 2002). Since accuracy does not account for the length of the response, it was computed as the number of recognised answers, divided by the number of retrieved answers. Table 7 shows the respective average values. Two scores are shown for each data set: the lower value concerns all questions in the set, and the higher value only questions for which at least one correct answer in the retrieved snippets existed. Contrary to the AQUAINT corpus, it is uncertain whether or not at least one answer can be found on the web to every question. Independently of taking into account all questions or not, ListWebQA ranks between the **top one and two** TREC systems. These results are strongly encouraging, due to the next two reasons: (a) ListWebQA did not use any specific predefined or compiled list of instances of foci, and (b) ListWebQA makes allowances for **web snippets**, not for **full documents**.

All in all, we envisage the use of Google n-grams to increase the precision of patterns, especially by matching the context of Qv-P and Pun-P, or the use of the number of patterns that support an answer as a measure of reliability.

Finally, it is definitely worth remarking that ListWebQA deals with an additional challenge while it is mining the web for answers to list questions. Unlike some QAS in TREC, ListWebQA has any passages retrieved from another corpus or pre-defined lists that acts like a filter for answers, and thus it must determine right answers directly from their contexts. We strongly believe that this is a more realistic scenario, and for this reason, the TREC gold standards were only used as reference list question sets.

4.3 Ranking

For the purpose of assessing our two ranking strategies (with and without the usage of Google n-grams), the next two aspects were taken into account: (a) all ranking strategies account for the same set of answer candidates, and (b) the higher ranked right answers are, the better the ranking strategy is. For these two reasons, in order to assess our ranking strategies, a variation of the standard metric of *Mean Reciprocal Rank* (MRR) was used. MRR rewards each answer according to its position in the ranking, by assigning each the inverse of its position (Lita and Carbonell 2004). Consequently, the MRR of a rank is interpreted as the sum of the ranking of each right answer.

In addition, B-II was implemented, a baseline that ranks answers according to the matrix MM^T and R_{max} (see section 3.5), this way the contribution of the dimensionality reduction provided by LSA can be measured. B-II breaks ties randomly. Table 8 describes our results. R^*_{max} stands for the ranking that accounts for LSA and the information supplied by Google n-grams.

Table 8: TREC 2001-2002 Results (Ranking).

	TREC 2001			**TREC 2002**		
	B-II	R_{max}	R^*_{max}	B-II	R_{max}	R^*_{max}
1	2.46	1.88	1.88	0.68	0.84	0.68
2	1.87	2.33	2.65	0	0	0
3	2.07	1.93	2.42	0.22	0.16	0.75
4	0.3	0.28	0.56	1.24	2.14	2.22
5	1.75	2.25	2.28	1.61	1.63	1.67
6	1.98	1.99	1.14	0.73	0.64	0.88
7	2.37	0.78	1.03	0	0	0
8	0.08	0.13	0.15	1.25	1.16	1.14
9	0.13	0.33	0.5	0	0	0
10	1.63	1.84	0.44	0.19	0.31	0.38
11	0.54	0.33	0.33	0	0	0
12	0	0	0	1.69	0.84	1.71
13	1.51	1.7	0.67	0	0	0
14	1.46	1.44	1.46	0.34	0.95	1.08
15	2.53	1.57	2.67	1.31	1.53	1.8
16	2.73	2.45	1.67	0	0	0
17	2.88	2.75	2.84	1.31	1.29	1.38
18	1.91	1.99	1.33	1.15	0.17	0.38
19	2.04	2.28	2.51	1.84	1.33	2.27
20	0	0	0	0.2	0.05	0.1
21	0	0	0	0.32	0.31	0.41
22	1.2	1.73	1.8	0	0	0
23	0.09	0.07	1	0.01	0.04	0.04
24	0.018	0.23	0.26	0	0	0
25	0	0	0	1.21	1.18	0.8

R^*_{max} obtains the higher MRR value for 22 out of 38 questions. It betters the ranking provided by LSA in 27 cases, and due to the alignment of spurious answers, it worsens it for eight questions. Under these results, we can claim that Google n-grams provide a valuable source for improving the ranking of answers. In addition, given the unclear distinction drawn by the MRR values of B-II and R_{max}, we can also conclude that LSA did not improve the ranking noticeably. We interpret this as a consequence of the need for additional contextual information to infer more reliable semantic relations, especially, when the number of answers in the fetched snippets is large. Here, we envisage the usage of a larger number of snippets and another vector representation of the answer candidates.

Furthermore, Google n-grams did not improve the ranking of answers corresponding to queries like “*Name 5 Diet Sodas*”, because no n-gram could be aligned with an answer candidate.

Finally, the improvement caused by Google n-grams unveils one more important fact; they supply a reliable way to detect some (short) answers candidates before the snippet retrieval. They can also play an essential role in the rewriting of the query, that is, in building efficient queries, which can bring about a higher recall and a massive redundancy, causing the detection of more reliable answers and a better final rank.

5 Preliminary Conclusions

In this paper, we presented ListWebQA, a question answering system which aimed specially at extracting answers to list questions from web snippets. Our research in progress has shown that it is possible to fetch answers to list questions in web snippets and indicates that it is feasible to discover some of these answers.

Our ultimate goal is supporting opinion questions, which is a system capable of answering questions such as "*What are the best Chuck Berry's songs?*". For this purpose, our intended system needs to find out the list of answers, and afterwards, summarise their corresponding opinions given by people.

References

Bellegarda, J. R. (2000), Exploiting Latent Semantic Information in Statistical Language Modeling, *in* IEEE, Vol. 88, No. 8, August 2000, pp. 1279–1296.

Cederberg, S., Windows, D. (2003), Using LSA and Noun Coordination Information to Improve the Precision and Recall of Automatic Hyponymy Extraction, *in* 'Proceedings of Conference on Natural Language Learning (CoNLL-2003)', Edmonton, Canada, pages 111-118.

Deerwester, S., Dumais, S. T., Furnas, G. W., T. Landauer, K., Harshman, R. (1990), Indexing By Latent Semantic Analysis, *in* Journal of the American Society For Information Science, 41, pp. 391–407, 1990.

Figueroa, A., Neumann, G. (2006), Language Independent Answer Prediction from the Web, *in* 'Proceedings of the FinTAL 5th International Conference on Natural Language Processing', August 23-25 in Turku, Finland.

Figueroa, A., Atkinson, J. (2006), Using Syntactic Distributional Patterns for data-driven Answer Extraction from the Web, *in* 'the 5th Mexican International Conference on Artificial Intelligence', November 13-17, Apizaco, Mexico, 2006.

Figueroa, A., Neumann, G. (2007), A Multilingual Framework for Searching Definitions on Web Snippets, *in* 'KI 2007: Advances in Artificial Intelligence', LNCS, Volume 4667/2007, p. 144-159.

Hearst, M. (1992), Automatic Acquisition of Hyponomys from Large Text Corpora, *in* 'Proceedings of the Fourteenth International Conference on computational Linguistics', Nantes, France, 1992.

Hearst, M. (1992), Automated Discovery of WordNet Relations, *in* 'WordNet: An Electronic Lexical Database and some of its Applications', Christiane Fellbaum (Ed.), MIT Press.

Katz, B., Marton, G., Borchardt, G., Brownell, A., Felshin, S., Loreto, D., Louis-Rosenberg, J., Lu, B., Mora, F., Stiller, S., Uzuner, O., Wilcox, A. (2005), External Knowledge Sources for Question Answering, *in* 'Proceedings of TREC 2005', Gaithersburg, Maryland, 2005.

Katz, B., Bilotti, M., Felshin, S., Fernandes, A., Hildebrandt, W., Katzir, R., Lin, J., Loreto, D., Marton, G., Mora, F., Uzuner, O. (2004), Answering multiple questions on a topic from heterogeneous resources, *in* 'Proceedings of TREC 2004', Gaithersburg, Maryland, 2004.

Katz, B., Lin, J., Loreto, D., Hildebrandt, W., Bilotti, M., Felshin, S., Fernandes, A., Marton, G., Mora, F. (2003), Integrating Web-based and Corpus-based Techniques for Question Answering, *in* 'Proceedings of TREC 2003', Gaithersburg, Maryland, 2003, pages 426–435.

Kintsch, W. (1998), Predication, *in* 'Cognitive Science', Volume 25, pages 173–202.

Landauer, T. K., Foltz, P. W., Laham, D. (1998), Introduction to Latent Semantic Analysis, *in* Discourse Processes, 25, pp. 259–284.

Lita, L., Carbonell, J. (2004), Unsupervised Question Answering Data Acquisition From Local Corpora, *in* Proceedings of the Thirteenth Conference on Information and Knowledge Management (CIKM 2004), November, 2004.

Shinzato, K., Torisawa, K.(2004), Acquiring hyponymy relations from web documents, *in* 'Proceedings of HLT-NAACL 2004', pages 73–80, 2004.

Shinzato, K., Torisawa, K.(2004), Extracting hyponyms of prespecified hypernyms from itemizations and headings in web documents, *in* 'Proceedings of 20th International Conference on Computational Linguistics', pages 938–944, 2004.

Schone, P., Ciany, G., Cutts, R., Mayfield, J., Smith T. (2005), QACTIS-based Question Answering at TREC 2005, *in* 'Proceedings of TREC 2005', Gaithersburg, Maryland, 2005.

Schütze, H. (1997), Ambiguity Resolution in Language Learning, Computational and Cognitive Models, *in* Computational and Cognitive Models. CSLI Lecture Notes, number 71, 1997.

Sombatsrisomboon, R., Matsuo, P., Ishizuka, M. (2003), Acquisition of Hypernyms and Hyponyms from the WWW, *in* 'Proceedings of 2nd International Workshop on Active Mining', 2003.

Voorhees, E. M. (2001), Overview of the TREC 2001 Question Answering Track, *in* 'Proceedings of TREC 2001', Gaithersburg, Maryland, 2001.

Voorhees, E. M. (2002), Overview of the TREC 2002 Question Answering Track, *in* 'Proceedings of TREC 2002', Gaithersburg, Maryland, 2002.

Voorhees, E. M. (2003), Overview of the TREC 2003 Question Answering Track, *in* 'Proceedings of TREC 2003', Gaithersburg, Maryland, 2003.

Voorhees, E. M. (2004), Overview of the TREC 2004 Question Answering Track, *in* 'Proceedings of TREC 2004', Gaithersburg, Maryland, 2004.

Wiemer-Hastings, P., Zipitria, I. (2001), Rules for Syntax, Vectors for Semantics, *in* 'Proceedings of the 23rd Annual Conference of the Cognitive Science Society', 2001.

Wu, L., Huang, X., Zhou, Y., Zhang, Z., Lin, F. (2005), FDUQA on TREC2005 QATrack, *in* 'Proceedings of TREC 2005', Gaithersburg, Maryland, 2005.

Yang, H., Chua, T. (2004), Effectiveness of Web Page classification on Finding List Answers, *in* 'Proceedings of SIGIR '04', Sheffield, United Kingdom, pp. 522–523.

Yang, H., Chua, T. (2004), FADA: find all distinct answers, *in* 'Proceedings of WWW Alt. '04', New York, NY, USA, pp. 304–305.

Yang, H., Chua, T. (2004), Web-based List Question Answering, *in* 'Proceedings of Cooling '04', Geneva, Switzerland, pp. 1277–1283.