FINDING DISTINCT ANSWERS IN WEB SNIPPETS

Alejandro Figueroa and Günter Neumann
Deutsches Forschungszentrum für Künstliche Intelligenz - DFKI
Stuhlsatzenhausweg 3, D - 66123, Saarbrücken, Germany

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

Abstract: This paper presents ListWebQA , a question answering system aimed specifically at discovering answers to list questions in *web snippets*. ListWebQA retrieves snippets likely to contain answers by means of a query rewriting strategy, and extracts answers according to their syntactic and semantic similarities afterwards. These similarities are determined by means of a set of surface syntactic patterns and a Latent Semantic Kernel. Results show that our strategy is effective in strengthening current web question answering techniques.

1 INTRODUCTION

In the last decade, the rapid increase in the number of web documents, in particular HTML pages, has provoked a remarkable and progressive improvement in the power of indexing of vanguard search engines, such as MSN Search. The great success of these search engines in linking users to nearly all the sources that satisfy their information needs, has caused an explosive growth in their number. Analogously, the demand of users for smarter ways of searching and presenting the requested information has also increased. Currently, one growing demand is finding answers to natural language questions. Most of the research in this area has been carried out under the umbrella of Question Answering Systems (QAS), specifically in the context of the Question Answering track of the Text REtrieval Conference (TREC).

TREC encourages QAS to answer several kinds of questions, whose difficulty has been systematically increasing during the last few years. In 2001, TREC incorporated *list questions* such as "*What are 9 novels written by John Updike?*". Simply put, answering this sort of question consists chiefly in discovering a set of different answers across several documents. However, QAS in TREC have obtained a modest success, showing that dealing with this kind of question is particularly difficult (Voorhees, 2001; Voorhees, 2003).

This paper presents ListWebQA , a list question answering system aimed at extracting answers only to list questions exclusively from the brief descriptions of web-sites returned by search engines, called *web snippets*. The motivation behind the use of web snippets as an answer source is three-fold: (a) to avoid the costly retrieval and processing of full web 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 a change in the search strategy of QAS or a request for additional feedback at the user. On the whole, exploiting snippets for list question answering is a key topic in the research realm 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 results, and section 5 draws 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 corpus). QAS usually start by distinguishing the *focus* of the query. The *focus* is the most descriptive noun phrase of the expected answer type (Katz et al., 2003). It thus associates the question with its answer type. Some QAS, hence, take into account pre-defined lists of instances of several *foci*, this way they find out right answers by match-

Figueroa A. and Neumann G. (2008).
FINDING DISTINCT ANSWERS IN WEB SNIPPETS.
In *Proceedings of the Fourth International Conference on Web Information Systems and Technologies*, pages 26-33
DOI: 10.5220/0001518900260033

ing elements of these lists with a set of retrieved passages. For example, (Katz et al., 2004) accounted for a list of 7800 famous people extracted from biography.com. They increased additionally their 150 predefined 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 specifically in adding synonymous noun phrases that could be used to ask about the list. 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 threshold.

(Schone et al., 2005) also cut-off low-ranked answers according to a threshold. These answers were obtained by interpreting a list question as a traditional factoid query and finding its best answers afterwards. 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, therefore, assisted by several deep processing tools, such as co-reference resolution. This way complex noun phrase constructions and relative clauses can be handled (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 and increase the chance of detecting answers, while at the same time, reducing 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 AQUAINT corpus, where previously known answers occurred. They found that the semantic of the lexico-syntactic constructions of these sentences matches the constructions observed by (Hearst, 1992) for recognising hyponomic relations. These constructions, which frequently occur within natural language texts (Hearst, 1992), are triggered by 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.

(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, 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 also not necessarily complete, especially with respect to online encyclopedias.

(Yang and Chua, 2004b) 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 a page, where answers occur, is likely to contain the subject of the relation established by the submitted query. They then extracted answers and projected them on the AQUAINT corpus afterwards. In this method, the corpus acted as a filter of 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). Since a hyponym and its hypernym are expected to share a semantic similarity, the plausibility of a putative hyponomic relationship is given by its degree of semantic similarity in the space provided by Latent Semantic Analysis (LSA). Furthermore, they 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.

3 MINING WEB SNIPPETS FOR ANSWERS

ListWebQA receives a natural language query Q as input and performs the following steps. Firstly, ListWebQA analyses Q in order to determine its noun phrases and *focus* as well as verbs (section 3.1). Secondly, it retrieves web snippets that are likely to contain answers by means of four purpose-built queries (section 3.2). Thirdly, ListWebQA discriminates answers candidates in these web snippets on the ground of a set of syntactic patterns (section 3.3). Lastly, it chooses answers by means of a set of surface patterns, Google n-grams[1], coordinations of answers, and a **Latent Semantic Kernel** (LSK) (section 3.4).

3.1 Query Analysis

ListWebQA starts similarly to (Yang and Chua, 2004b), by removing head words (i. e. "*What are*") from Q. From now on, Q refers to this query without head words. Next, it uses part-of-speech (POS) tags[2] 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 are permanently discarded.
- **Foci** are words or sequences of words tagged as NNS, apart from stop-words. In particular, "*novels*" in "*novels written by John Updike*". 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 positive 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 thing is, reliable and efficient POS taggers for public use currently exist, contrary to chunkers, which still need improvement.

Additionally, we briefly tried the subject-object representation of sentences, like (Yang and Chua, 2004b), provided by MontyLingua[3]. However, some difficulties were encountered, while computing the representation of some queries.

3.2 Retrieving Web Snippets

On the one hand, (Yang and Chua, 2004a) observed that web pages, where answers to list questions occur, contain a noun phrase of Q in the title. On the other hand, state-of-the-art search engines supply a feature "*intitle*" that assists users to fetch web pages, in which their title matches a given input string. ListWebQA makes allowances for this feature to bias the search in favour of pages that are very likely to contain answers, more precisely, web pages predominantly entitled with query NNPSs and/or NNPs. 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 latter brings about a low recall. We call this concatenation *a title clause*.

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 NNSs, they are concatenated with the disjunction "*or*". Since ListWebQA looks for web pages containing both, the desired title and body, they are linked with the conjunction "*and*". The following search query corresponds to $Q =$"*novels written by John Updike*":

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

This query unveils another key aspect of our web search strategy: query verbs are also added to the *body clause*. A snippet retrieved by this query is:

- IMS: **John Updike**, HarperAudio
  Author and poet **John Updike** reads excerpts from his short story "The Persistence of Desire". ... **Updike**'s

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

[2] http://nlp.stanford.edu/software/tagger.shtml

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

other published works include the **novels** "Rabbit Run", "Couples", and "The Witches of ...

Certainly, TREC list question sets have questions that do not contain any NNPS or NNP, especially the query "*Name 6 comets*" provides only the clause *inbody:("COMETS")*. In fact, ListWebQA prefers not adding NNSs to the *title clause*, because they lead the search to unrelated topics. We see this 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*. However, this ambiguity is lessened if the NN/NNS occurs along with an adjective or if it represents a merged sequence of NNS/NNs (section 3.1). In this case, ListWebQA generates a *title clause* instead of a *body clause*, which also accounts for the corresponding lemma. To neatly illustrate, the query "*What are 6 names of navigational satellites?*" supplies the clause *intitle:("NAVIGATIONAL SATELLITES" OR "NAVIGATIONAL SATELLITE")*.

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 "NOVELS INCLUDING") AND inbody:("WRITTEN")

- intitle:("JOHN UPDIKE") AND inbody:("NOVELS SUCH AS" OR "NOVELS INCLUDE") AND inbody:("WRITTEN")

Two search 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, they do not consider lemmas, and the 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 addition, ListWebQA generates an extra search query which aims 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 search query looks as follows:

- inbody:("NAVIGATIONAL SATELLITES") AND (site:en.wikipedia.org OR site:www.answers.com)

In particular, a retrieved snippet by this query is:

- 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

This snippet highlights how our 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 by the answer. ListWebQA retrieves the relevant sentences without downloading and processing this document. Lastly, it is also worth noting that each submission retrieves the first 20 snippets.

Pre-processing

Once all snippets are retrieved, ListWebQA splits them into sentences by means of truncations and JavaRap[4]. Every time ListWebQA detects a truncated sentence that fulfils two conditions, it is submitted to the search engine (in quotes), and the newly fetched sentence replaces the old one. These two conditions are: (a) it contains a coordination of elements, and (b) this coordination is indicated by some hyponomic keywords. Accordingly, sentences are also identified in these fetched extensions.

3.3 Answer Candidate Recognition

One of the major problems of answering list questions is the fact that the type of the *focus* varies widely from one question to another. 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 NERs do not recognise all types of *foci*, and furthermore, their performance 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)

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

where "*S*", "*P*", "*N*" stand for a stop-word, a punctuation sign, and a number, respectively. "*C*" stands for a word, which starts with a capital letter, "*L*" for a lower-cased word, and eventually, "*#*" marks a sentence limit. The first pattern aims at names of persons, places, books, songs, and novels, such as "*The Witches of Eastwick*." The second pattern aims at a single isolated word which starts with a capital letter (i. e. country names).

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*". Then, ListWebQA maps every entity to a place holder "*entityX*", where "*X*" is assigned according to each individual entity.

The next step is replacing all query verbs with a place holder. Here, ListWebQA also considers morphological variations of verbs. For example, the words "*write*", "*writing*", and "*written*" are mapped to the same place holder "*qverb0*", where the zero indexes the respecting verb within Q. ListWebQA then does a similar processing with *foci* in Q. In this case, plural and singular forms are mapped to the same place holder. For instance, "*novel*" and "*novels*" are mapped to "*qfocus0*", where "*0*" is accordingly the corresponding index. Consequently, ListWebQA follows the same strategy for noun phrases within the query. In addition, ListWebQA maps substrings within query noun phrases to the same place holder "*qentity*". The next snippet sketches this abstraction:

- **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**."

From this snippet abstraction, ListWebQA distinguishes a set $\mathcal{A}$ of answer candidates according to the patterns in table 1. It is worth remarking that π_3 and π_7 are only used for matching snippet titles, while π_1 is aimed at the patterns proposed by (Hearst, 1992), and π_4 is aimed at the copular pattern.

3.4 Selecting Answers

First of all, ListWebQA determines a set $\mathcal{P} \subseteq \mathcal{A}$ consisting of all answers matching at least two different patterns in Π. Second, it constructs a set $\mathcal{C} \subseteq \mathcal{A}$ by examining whether any answer candidate occurs in two different coordinations triggered by patterns π_1 and π_8. Third, ListWebQA discriminates a set $\mathcal{E} \subseteq \mathcal{A}$ of answers on the ground of their syntactic bonding with the query by inspecting their frequency given by Google 5-grams as follows:

a. Trims query entities by leaving the last two words. For example: "*Frank Lloyd Wright*" remains as "*Lloyd Wright*".
b. Appends punctuation signs to these trimmed query entities, in such a way that match patterns shown in Π:
   - Lloyd Wright ('s|:|'|")
c. Searches for 5-grams matching this pattern.
d. Partially aligns the beginning of each answer candidate with the context yielded by every (matched) Google 5-grams.

Fourth, ListWebQA determines a set $\mathcal{F} \subseteq \mathcal{A}$ of answers by aligning answers in $\mathcal{A}$ with the context conveyed by Google 5-grams that match the next pattern:

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

Fifth, ListWebQA scores each coordination signalled by patterns π_1 and π_8 according to its set γ of conveyed answers candidates and the next equation:

$$H(\gamma) = 2(\mid \gamma \cap \mathcal{E} \mid + \mid \gamma \cap \mathcal{B} \mid) + \mid \gamma \cap \mathcal{F} \mid + 3(\mid \gamma \cap \mathcal{P} \mid + \mid \gamma \cap \mathcal{C} \mid) +$$

ListWebQA initialises $\mathcal{B}$ as $\emptyset$, and adds answers to $\mathcal{B}$ by bootstrapping coordinations. At each iteration, this bootstrapping selects the highest scored coordination, and finishes when no coordination fulfils $H(\gamma) \geq \mid \gamma \mid$. Every previously selected coordination is unconsidered in the next loops. Consequently, this bootstrapping assists ListWebQA to infer some low frequent answers surrounded by reliable answers.

Sixth, ListWebQA ranks all answers in $\mathcal{A}$ excluding those only matching π_1 and π_8, by measuring the semantic similarity to Q of every context where these answers occur. (Cederberg and Windows, 2003) tested the degree of semantic relationship between two terms by means of LSA. Conversely, ListWebQA determines the semantic similarity of every snippet abstraction to the corresponding abstraction of Q (see section 3.3), that is the similarity between two **sets of terms**, making use of the LSK proposed by (Shawe-Taylor and Cristianini, 2004). ListWebQA weights accordingly the respective frequency matrix with *tf-idf* and normalises the kernel. The rank of an answer candidate is hence given by the sum of all the different contexts, where it occurs, that match π_2 to π_7. Eventually, ListWebQA builds a set $\mathcal{K}$ from the highest 40% ranked answers, whose rank values are also

Table 1: Set of Syntactic Patterns Π for recognising Answer Candidates at the sentence level.

Π	Pattern
π_1	**qfocus** (*such as*|*like*|*include*|*including*) (**entity**,)+ (and|or) **entity** ***qentity0****'s other published works include the* ***qfocus0*** *"****entity3****", "****entity4****", and "****entity5****".* ⇒ ***Updike****'s other published works include the novels* "Rabbit Run"*,* "Couples" *and* "The Witches of Eastwick".
π_2	\w* (**qentity**|**qfocus**) \w* ("**entity**"|'**entity**') \w*. \w* ("**entity**"|'**entity**') \w* (**qentity**|**qfocus**) \w*. ***qentity0*** *wrote the* ***qfocus0*** *"****entity6****".* ⇒ ***John Updike*** *wrote the novel "Brazil".*
π_3	:**qentity**:(\w+:){0,1}**entity** :**entity**:(\w+:){0,1}**qentity** *Amazon.com:****entity10****:Books:****qentity0*** ⇒ *Amazon.com:****Terrorist****:Books:****John Updike***
π_4	**entity** is \w+ **qfocus** \w* (**entity**,)+ and **entity** are \w+ **qfocus** \w* ***entity1*** *is ...* ***qentity0****'s* ***qfocus0*** *brand.* ⇒ *Chubby Hubby is ... Ben and Jerry's ice cream brand.*
π_5	**qentity**'s **entity** **qentity**'s (**entity**,)+ (and|or) **entity** ***qentity0****'s* ***entity9*** *or* ***entity11****.* ⇒ *Frank Lloyd Wright's Duncan House or The Balter House.*
π_6	(**qentity**|pronoun|**qfocus**) \w{0,3} **qverb** \w{0,3} **entity** **entity** \w{0,3} **qverb** \w{0,3} prep \w{0,3} **qentity** ***qentity0 qverb0*** *his native* ***entity16****.* ⇒ *Pope John Paul II visited his native Poland.*
π_7	**entity qfocus** ***entity15 qfocus****.* ⇒ *The Cincinnati Subway System.*
π_8	**qentity0** \w* **qfocus** (:|,) (**entity**,)+ (and|or) **entity** *Six* ***qentity0*** *...* ***qfocus0****:* ***entity3****,* ***entity1****,* ***entity7****,* ***entity13****,* ***entity1****, and* ***entity9****.* ⇒ *Six Nobel Prizes ... categories: Literature, Physics, Chemistry, Peace, Economics, and Physiology & Medicine.*

greater than an experimental threshold (0.74). If $|\mathcal{K}| < 10$, $\mathcal{K}$ is extended to the ten top ranked answers.

ListWebQA builds a set $\mathcal{E}' \subseteq \mathcal{E}$ of answers that are closely (semantically) related to Q, by ensuring a similarity greater than the experimental threshold (0.74). Last, if $\mathcal{B} = \emptyset$, it outputs $\mathcal{E}' \cup \mathcal{K}$, otherwise $\mathcal{B} \cup \mathcal{E}'$.

4 EVALUATION

ListWebQA [5] was assessed by means of the list question sets supplied by TREC from 2001 to 2004. Accordingly, errors in query analysis are discussed in section 4.1, and section 4.2 highlights the increase in recall obtained by our snippet retrieval strategy. In addition, section 4.3 remarks the accuracy of patterns in table 1, and eventually, section 4.4 compares our results with other systems.

4.1 Query Rewriting

Stanford POS Tagger outputted significant mistaggings for one question in the TREC 2002 and 2003 data sets, while answering two questions in the TREC 2004 list question set. The main problem was caused by words like "*agouti*" and "*AARP*", which were interpreted as RB. Since ListWebQA does not consider RBs while it is rewriting Q, these mistaggings brought about misleading search results.

4.2 Answer Recall

ListWebQA increases the recall of answers by retrieving a maximum of 80 snippets (see section 3.2). Accordingly , a baseline (BASELINE) was implemented that also fetches a maximum of 80 snippets by submitting Q to the search engine. The achievements for the four TREC datasets, are shown in table 2.

Table 2: TREC Results (Answer Recall).

	2001	2002	2003	2004
BASELINE (Recall)	0.43	0.49	0.4	0.65
ListWebQA (Recall)	0.93	0.90	0.56	1.15
BASELINE (NoS)	77.72	77.33	80	78.87
ListWebQA (NoS)	59.83	53.21	51.86	46.41
BASELINE (NAF)	2	4	8	12
ListWebQA (NAF)	6	2	8	11

In table 2, NoS signals the average number of retrieved snippets per query, and NAF the number of questions in which there was no answer in these fetched snippets. This involved a necessary manual inspection of the retrieved snippets, because they do not necessarily contain the same answers supplied by TREC gold standards. Overall, ListWebQA retrieved

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

significantly less snippets and markedly increased the recall of distinct answers. This recall was computed as the average ratio of the number of answers retrieved by the system to the number of answers provided by TREC. The reason to use this ratio is twofold: (a) TREC provides at least one answer to every question, this way undefined ratios are avoided, and (b) additional answers are rewarded according to the size of the reference set, that is one extra answer is rewarded higher if the reference set contains less answers for the respective question. ListWebQA fetched **a larger number of answers** than the number provided by TREC gold standards in 41 out of the 142 questions. In particular, in 11, 9, 5, and 16 questions corresponding to TREC 2001, 2002, 2003 and 2004, respectively. It is also worth highlighting, TREC gold standard considers all answers found in the AQUAINT corpus by the assessors and also includes new answers found by the different systems. The major difference exists in the 32nd question of TREC 2004 "*Wiggles' songs*". Here, ListWebQA retrieved 62 distinct answers, whereas TREC gold standards only supplied four.

A second point to consider is that, the three sets of answers radically differ. For example, three of Edgar Allan Poe's works retrieved by BASELINE are "*Annabel Lee*", "*Landor's Cottage*" and "*The Haunted Palace*". In this case, neither the TREC gold standard or the output of ListWebQA contained all these works. Therefore, it was computed the ratio of common answers to the number of all distinct answers in both retrieved snippets. Overall, an average of 0.21 was obtained. To sum this up, ListWebQA retrieved a **smaller set of snippets** with **more distinct answers**, and we hypothesise that both strategies could be combined to achieve a higher recall.

4.3 Answer Candidate Recognition

Table 3: Patterns Accuracy.

π_1	π_2	π_3	π_4	π_5	π_6	π_7	π_8
0.35	0.36	0.15	0.34	0.22	0.26	0.14	0.19

Table 3 indicates the accuracy of each pattern in Π. One reason for this low accuracy 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 π_1 and its title contains the noun phrase "*United States*", but it is regarding a topic unrelated to "*vehicle emission inspection programs*". Consequently, this kind of semantic mismatch supplies incorrect answers. This illustrative mismatch, provided four wrong answers (according to TREC gold standards). All in all, ListWebQA recognised an average of 60% of the retrieved distinct answers.

4.4 Answer Selection

QAS in the list question subtask of TREC have been assessed with different measures. In 2001 and 2002, the measure of performance was accuracy (Acc.), which was computed as the number of distinct instances returned by the system divided by the target number of instances (Voorhees, 2001). Since accuracy does not account for the length of the response, it was changed to the F_1 score in 2003 (Voorhees, 2003). Accordingly, Table 4 highlights the average accuracy and F_1 score obtained by ListWebQA .

Table 4: TREC Final Results.

	2001	2002	2003	2004
ListWebQA (F_1)	.35/.46	.34/.37	.22/.28	.30/.40
ListWebQA (Acc.)	.5/.65	.58/.63	.43/0.55	.47/.58
Top one (Acc.)	0.76	0.65	-	-
Top two (Acc.)	0.45	0.15	-	-
Top three (Acc.)	0.34	0.11	-	-
Top one (F_1)	-	-	0.396	0.622
Top two (F_1)	-	-	0.319	0.486
Top three (F_1)	-	-	0.134	0.258

Two scores are shown for each measure and 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, there is uncertainty as to whether or not at least one answer can be found on the web for every question. Since accuracy does not account for the length of the response, it was calculated considering the set $\mathcal{A}$ of answer candidates. Conversely, the F_1 score was determined from the set after answer selection. Independently of taking into account all questions or not, ListWebQA ranks between the **top one and two** systems in the first two question sets, while between the **second and the third** in the last two data sets. These results are 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**. These two reasons remark our highly promising results especially considering

other approaches (Yang and Chua, 2004a; Yang and Chua, 2004b), which **download and process** more than 1000 full web documents, or submit more than 20 queries to different search engines, finishing with an F_1 score of .464 $\sim$ .469 on TREC 2003. Our strategy can strengthen their strategy, specially their classification and clustering of full documents.

In contrast to the observations in TREC 2001 (Voorhees, 2001), duplicate answers have a considerable impact on the performance, because answers are taken from many different sources. One singular case is the several spellings and misspellings of an answer. For instance, `ListWebQA` retrieved three different spellings/misspellings for the Chuck Berry's song "*Maybelline*" (also found as "*Maybellene*" and "*Maybeline*"). Additionally, inexact or incomplete answers also have an impact on the performance. For example, John Updike's novel "*The Poorhouse Fair*" was also found as "*Poorhouse Fair*".

5 CONCLUSIONS AND FUTURE WORK

This paper presented `ListWebQA` , a question answering system which aimed specially at extracting answers to list questions from web snippets. Our results indicate that it is feasible to discover answers in web snippets. We envisage that these answers will help to select the most promising documents, and afterwards, detecting the portions where these answers are.

Additionally, we envision that dependency trees can be used to increase the accuracy of the recognition of answer candidates, and extra search queries can be formulated in order to boost the recall of answers in web snippets. For this last purpose, we deem that Google n-grams and on-line encyclopaedias would be tremendously useful.

ACKNOWLEDGEMENTS

This work 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.

REFERENCES

Cederberg, S. and Windows, D. (2003). Using lsa and noun coordination information to improve the precision and recall of automatic hyponymy extraction. In *Conference on Natural Language Learning (CoNLL-2003)*, pages 111–118, Edmonton, Canada.

Hearst, M. (1992). Automatic acquisition of hyponomys from large text corpora. In *Fourteenth International Conference on computational Linguistics*, pages 539–545, Nantes, France.

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

Katz, B., Lin, J., Loreto, D., Hildebrandt, W., Bilotti, M., Felshin, S., Fernandes, A., Marton, G., and Mora, F. (2003). Integrating web-based and corpus-based techniques for question answering. In *TREC 2003*, pages 426–435, Gaithersburg, Maryland.

Katz, B., Marton, G., Borchardt, G., Brownell, A., Felshin, S., Loreto, D., Louis-Rosenberg, J., Lu, B., Mora, F., Stiller, S., Uzuner, O., and Wilcox, A. (2005). External knowledge sources for question answering. In *TREC 2005*, Gaithersburg, Maryland.

Schone, P., Ciany, G., Cutts, R., Mayfield, J., and Smith, T. (2005). Qactis-based question answering at trec 2005. In *TREC 2005*, Gaithersburg, Maryland.

Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel methods for pattern analysis*, chapter 10, pages 335–339. Cambridge University Press.

Shinzato, K. and Torisawa, K. (2004a). Acquiring hyponymy relations from web documents. In *HLT-NAACL 2004*, pages 73–80, Boston, MA, USA.

Shinzato, K. and Torisawa, K. (2004b). Extracting hyponyms of prespecified hypernyms from itemizations and headings in web documents. In *COLING '04*, pages 938–944, Geneva, Switzerland.

Sombatsrisomboon, R., Matsuo, P., and Ishizuka, M. (2003). Acquisition of hypernyms and hyponyms from the www. In *2nd International Workshop on Active Mining*, Maebashi, Japan.

Voorhees, E. M. (2001). Overview of the trec 2001 question answering track. In *TREC 2001*, pages 42–51, Gaithersburg, Maryland.

Voorhees, E. M. (2003). Overview of the trec 2003 question answering track. In *TREC 2003*, pages 54–68, Gaithersburg, Maryland.

Wu, L., Huang, X., Zhou, Y., Zhang, Z., and Lin, F. (2005). Fduqa on trec2005 qatrack. In *TREC 2005*, Gaithersburg, Maryland.

Yang, H. and Chua, T. (2004a). Effectiveness of web page classification on finding list answers. In *SIGIR '04*, pages 522–523, Sheffield, United Kingdom.

Yang, H. and Chua, T. (2004b). Web-based list question answering. In *Proceedings of COLING '04*, pages 1277–1283, Geneva, Switzerland.