Mining Wikipedia Resources for Discovering Answers to List Questions in Web Snippets

Alejandro Figueroa

German Centre for Artificial Intelligence - DFKI
Stuhlsatzenhausweg 3, D - 66123, Saarbrücken, Germany
figueroa@dfki.de

***Abstract*— This paper presents LiSnQA, a list question answering system that extracts answers to list queries from the short descriptions of web-sites returned by search engines, called *web snippets*. LiSnQA mines Wikipedia resources in order to obtain valuable information that assists in the extraction of these answers. The interesting facet of LiSnQA is, that in contrast to current systems, it does not account for lists in Wikipedia, but for its redirections, categories, sandboxes, and first definition sentences. Results show that these resources strengthen the answering process. In this work, we additionally deal at length with some findings concerning the distribution of answers to list questions across the web.**

I. Introduction

The number of people who access the Internet has quickly grown in the last five years. A chief reason behind this marked increase is the huge amount of documents about wide-ranging topics on the web. In particular, Google's current index size can be estimated at 30 billion pages whereas Yahoo's at 37 billion. The availability of this growing and extensive open-domain collection, heartens users to carry out searches of all sorts, including answers to natural language questions. This salient fact has naturally led vanguard search engines into the consideration of applications, like Yahoo! Answers[1], that are aimed at providing users with answers to their queries. In essence, in Yahoo! Answers, people ask and search for questions in order to find answers and get insights from real people. By and large, in this sort of methodology, whenever a user enters a new question, he/she must wait until another user satisfactorily answers the query.

In recent years, most of the research into automatic textual Question Answering has been conducted in the context of the Text REtrieval Conference (TREC). In TREC, Question Answering Systems (QAS) are challenged to extract answers to a rich diversity of queries, including list questions, from the AQUAINT[2] corpus. By nature, list queries, such as "*Name books written by C. S. Lewis*", are aimed at finding a set of correct answers that share some sort of relationship with each other, and with the question. For the purpose of successfully discovering these answers, some TREC QAS make use of public free online resources, especially lists supplied by online encyclopaedias, like **Wikipedia**. To QAS, Wikipedia is an invaluable source of answers, because it provides wide coverage in several domains and its semi-structured texts enhance their performance.

This paper presents LiSnQA, a list question answering system that mines Wikipedia resources, in order to obtain valuable information for extracting answers to list queries from *web snippets*. The interesting facet of LiSnQA is that, contrary to current list QAS, it does **not** account for **lists** in **Wikipedia** pages, but for its **categories**, and **sandboxes**, as well as its **first definition** sentences.

The motivation behind finding answers to list questions within *web snippets* is two-fold: (a) to the user, web snippets are the first view of the response, thus highlighting answers would make them more informative, and (b) answers taken from snippets can be useful for determining the most promising documents; more precisely, where most of answers are likely to be. The former is the primary driving force of this work, because it would empower search engines and applications like Yahoo! Answers to find answers on the web, at the time that the user searches or submits his/her question. An extra strong incentive is that the absence of answers across retrieved web snippets can force a change in the search strategy or a request for additional feedback at the user.

The roadmap of this paper is as follows: section II deals at greater length with the related work. Section III describes our strategy for mining Wikipedia in detail, and section IV highlights how LiSnQA uses the mined data for answering list questions. Accordingly, section V thoroughly discusses our experiments and results, section VI draws conclusions and yields some brief insights about the future work.

II. Related Work

In TREC, QAS have explored several strategies in order to find answers to list questions across the AQUAINT corpus. Normally, QAS begin by identifying the *focus* of the query, that is the most descriptive noun phrase of the expected answer type [1]. Hence, the *focus* makes a connection between the question and its answer type. Therefore, some TREC QAS have accounted for pre-defined lists of instances of several *foci*, this way they extract correct answers by matching elements of these lists with a set of fetched passages from the corpus. In particular, [2] made allowances for a list of 7800 famous

[1] http://answers.yahoo.com/
[2] http://www.ldc.upenn.edu/Catalog/byType.jsp

people taken from biography.com. In addition, they increased their 150 pre-defined and manually compiled lists used in TREC 2003 to 3300 in TREC 2004 [1]. These lists were semi-automatically extracted from WorldBook Encyclopedia articles by searching for hyponomyns. In TREC 2005, [3] generated these lists off-line by means of **subtitles** and **link structures** supplied by **Wikipedia**. This method **processed** a **full Wikipedia page** and its **related documents**. The manual annotation consisted specifically of adding synonymous noun phrases that could be used to ask about the list. One of their findings was 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.

In TREC 2005, [4] obtained patterns for recognising answers to list questions by checking the structure of sentences in the AQUAINT corpus where previously known answers occurred. They discovered that the semantics of the lexico-syntactic constructions of these sentences corresponds to the constructions observed by [5] for detecting hyponomic relations. These constructions, which frequently occur within natural language texts [6], are triggered by keywords like "*including*", "*include*", "*such as*" and "*like*". Later, [7] 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.

Reference [8] 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, [9] 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. [9] 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. Thus taking into consideration the whole document is positively encouraging. Moreover, 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?*".

References [10] and [11] 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 on the web page containing several potential answers. They further 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 some misleading and spurious answers.

The work of [12] was strongly guided by the empirical observation of [11]. They made use of the feature "*intitle*" supplied by commercial search engines for boosting the recall of answers to list questions within web snippets. More precisely, their system searched for web pages entitled with NNPSs and NNPs discovered during query analysis. As a result, they found that this strategy markedly betters the recall and precision of the search, but there are still many answers contained in documents that are not titled with query terms. They showed, nevertheless, that it is feasible to extract some answers to list questions from web snippets.

Reference [13] distinguished putative pairs hyponomy-hypernym on the British National Corpus by means of the patterns suggested by [5]. 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.

Reference [13] tested the degree of semantic relationship between two terms by means of LSA. Conversely, [14] determined the semantic similarity of every snippet to the corresponding query by making use of the *Latent Semantic Kernel* (LSK) proposed by [15]. Hence, answers candidates belonging to snippets semantically closer to the question are more likely to be answers. Another two interesting facets of this answer extraction strategy is that they take advantage of coordinations and bootstrapping in order to infer low frequent answers, hence showing that combining semantic inferences with syntactic patterns is encouraging. This method finished with encouraging results, especially considering that it does not make use of any online encyclopedias.

This paper introduces `LiSnQA`, a list question answering system built on top of the system in [14]. We extend this work by mining Wikipedia resources for extracting answers to list questions from web snippets. Since previous works have shown that lists slightly improve the performance, we investigate how additional resources can contribute to answer list questions. In this work, we also discuss two interesting findings that emerge when QAS search for answers to list questions on the web, and how they can strengthen the answering process.

TABLE I

HIGHEST FREQUENT N-GRAMS, IN WIKIPEDIA, THAT SIGNAL ALTERNATIVE NAMES.

a.k.a.	also called	colloquially known as	generally known as	or more commonly	referred to simply as
aka	also known as	commonly abbreviated	generally written as	or more precisely	sometimes called
nicknamed	also spelled	commonly called	informely known as	or simply	sometimes known as
, called	also written	commonly known as	initially known as	otherwise known as	sometimes spelled
, known as	also written as	commonly referred to as	officially called	previously called	sometimes spelt
, or the	an acronym for	commonly written as	officially known as	previously known as	sometimes written as
abbreviation for	best known as	formely known as	officially written as	previously written as	still known as
abbreviation of	better known as	generally called	often abbreviated	referred to as	widely known as

III. MINING WIKIPEDIA RESOURCES

Wikipedia[3] consists of different sorts of pages including redirection, disambiguation, definition, list, and categories. In particular, redirection pages contain no definition content, but they link an input string with the respective definition page. We interpret these input strings as rewritings of the main concept. To neatly illustrate, the redirection page of "*Clive S Lewis*" connects this name rewriting to the definition page of "*C. S. Lewis*". Essentially, we use these mappings for building an **off-line** database of name rewritings:

<C. S. Lewis, Clive Staples Lewis>
<C. S. Lewis, C.S. Lewis>
<C. S. Lewis, Clive S Lewis>

Additionally, this database is enriched with the alternative name rewritings conveyed in first definition sentences. Consider the following example corresponding to "*C. S. Lewis*":

*"**Clive Staples 'Jack' Lewis**" (29 November 1898 - 22 November 1963), commonly referred to as "**C. S. Lewis**", was an Irish author and scholar.*

Sentences containing alternative names are discriminated **off-line** on the grounds of pre-defined lexico-syntactic clues. These clues have been used in particular by definition QAS for discovering synonyms, and we extend this set of clues by inspecting high frequent n-grams, occurring in these sentences, that indicate alternative names (see table I). In our working example, we obtain the next mapping:

<C. S. Lewis, Clive Staples 'Jack' Lewis>

Here, the underlying assumption is that the first line conveys information of the corresponding main concept. Definition pages also provide an additional source of rewritings: translations. For example, the following are rewritings extracted from the translations of C.S. Lewis's book "*Mere Christianity*": "*Christentum schlechthin*" into German, and "*Mero Cristianismo*" into Spanish:

<Mere Christianity, Christentum schlechthin>
<Mere Christianity, Mero Cristianismo>

To illustrate, these translations assist in inferring that the following two names refer to the same entity in the next snippet:

*Amazon.ca: **Mero Cristianismo**: CS Lewis,Veronica Fernandez Muro: Books ... **Mere Christianity** is a book that uncovers common...*

Simply put, we built an **off-line** database of rewritings by mining redirection pages, and the first sentence in definition pages, as well as translations. Then, finding out the rewritings of a particular concept consists of looking for the right entry in this database.

Additionally, we constructed a second database consisting of all concepts and their respective first definition sentence. In the case of sentences shorter than 150 characters, we consider the first two sentences. All expressions that convey rewritings were removed, along with pieces of text in parentheses. An entry in this database looks as follows:

Clive Staples 'Jack' Lewis = CONCEPT was an Irish author and scholar.

The entries in this database are enriched **off-line** by its predicate-arguments structure[4]:

be(CONCEPT, Irish author and scholar)

In order to illustrate how these two databases link different concepts, consider the following entry:

The Dark Tower = CONCEPT is a novel written by C.S. Lewis.
be(CONCEPT, novel)
write(novel, by C.S. Lewis)

The equivalence between the different rewritings of "*C. S. Lewis*" is obtained from the first database, and consequently, it can be inferred that both concepts are related to each other, with the type of relationship given by the predicates. Additionally, we built a third **off-line** database consisting exclusively of sandboxes, commonly referred to as infoboxes, supplied by Wikipedia. An entry in this database is as follows:

A Grief Observed = {type:book; author:C. S. Lewis; country:United Kingdom; isbn:0553274864; followed_by:Letters to Malcolm; pages:160; preceded_by:An Experiment in Criticism; publisher:Faber and Faber; release_date:1961;}

Lastly, we constructed a fourth **off-line** database consisting of the categories provided by Wikipedia. In the case of *C.S. Lewis*, there are six categories including "*Books by C. S.*

[3]In the scope of this work, we use the snapshot supplied by Wikipedia in January 2008.

[4]For this purpose, we use Montylingua available at http://web.media.mit.edu/∼hugo/montylingua/

Lewis", and "*Novels by C. S. Lewis*". A sample entry of this categories database is as follows:

Poetry by C.S. Lewis={Spirits in Bondage; Dymer;}

In the next section, we describe the use of these databases for answering list questions.

IV. Answering List Questions

LiSnQA makes use of the previously presented databases for getting reliable answers, extending the query, and improving the recognition of names and answers within web snippets.

A. Discovering Reliable Answers

LiSnQA fetches predicates corresponding to sentences that match entities within the query. Here, if LiSnQA cannot find a definition that matches a query entity, it attempts to find sentences that match query entity rewritings taken from the rewritings database. If the question contains no entity, then LiSnQA tries to retrieve sentences containing query plural nouns, excluding the *focus*. If there is no plural noun in the query, LiSnQA fetches sentences, where the *focus* occurs. Here, it is worth remarking that words in multi-word *foci* are matched separately, and *foci*, such as "*Name auto-immune diseases*", are matched with and without hyphen. From now on, we refer to this set as $C_W(Q)$, the set of concepts found in Wikipedia that are related to the prompted question Q.

In order to identify which concept shows strong evidence of being an answer, the corresponding fetched predicates are processed as follows. If the question consists solely of a *focus* (e.g. "*Name auto-immune diseases*"), LiSnQA verifies if any concept in $C_W(Q)$ fulfills the next predicate-arguments structure:

be(CONCEPT, arg_1, arg_2, ..., arg_N)

Where the lemma of *focus* must be aligned with arg_1 or arg_2. This alignment consists of verifying whether or not this lemma is at the end of one of these two arguments, or followed by the conjunction "*or*", or the disjunction "*and*". A deeper alignment is due to *multiword foci*, like "*auto-immune disease*". In this case, LiSnQA additionally checks if the multi-word *focus* lemma is split across the same argument:

be(CONCEPT,chronic auto-immune skin disease)

If this alignment fails, the lemma of the *focus* is aligned with arg_1 and/or arg_2 of the second predicate. However, the subject of this predicate must be a pronoun. A good example is the autoimmune disease "*birdshot retinochoroidopathy*":

be(CONCEPT, form, of uveitis)
be(It, autoimmune disease)

Any concept in $C_W(Q)$ satisfying these constrains is interpreted as a *reliable answer*. However, some queries, such as "*cities with a subway system*", contain additional plural nouns. In these cases, LiSnQA also checks whether or not the lemma of a plural noun matches an argument of this predicate, but in a position posterior to the lemma of the *focus*. Queries containing entities, like "*songs of The Clash*", are verified if they fulfill one of the next three sets of predicates:

P_1:be(CONCEPT, focus lemma, ..., qentity, ..., arg_N)

P_1:be(CONCEPT, arg_1, arg_2, ..., arg_N)
P_2:vv(*, ..., focus lemma , ..., qentity, ..., arg_N)

P_1:be(CONCEPT, arg_1, arg_2, ..., arg_N)
P_2:vv(*, ..., focus lemma , ..., arg_N)
P_3:vv(*, ..., qentity, ..., arg_N)

The star, the tokens "*vv*" and "*qentity*" stand for anything as subject, anything as verb, and a query entity or a query entity rewriting, respectively. For instance, the following entry in our database satisfies the first set, and hence, "*Groovy Times*" is seen as a *reliable answer*:

Groovy Times = CONCEPT is a song by The Clash, featured on their "The Cost of Living", and released as a promotional single on 1979 in Australia by Epic Records.
be(CONCEPT, song, by The Clash)
feature(, on their "The Cost,of Living")
release(,)

However, some ambiguous concepts, including "*Tommy Gun*", convey the lemma of the *focus* within the title as a disambiguation feature. In this sort of question, LiSnQA additionally takes into consideration the alignment of the *focus* lemma with this disambiguation feature. In particular[5],

Tommy Gun (song) = CONCEPT was London punk rock band The Clash's 7th single, and the 1st single taken from their 2nd album "Give 'Em Enough Rope".
be(CONCEPT, London punk rock band The Clash)
take(, 1st, from their 2nd album Give 'Em Enough Rope)

This alignment consists in checking whether or not the *focus* lemma is at the end of the content in parenthesis, and also, it helps to lessen the impact of wrong outputs computed by Montylingua, and for this reason, to distinguish more *reliable answers*. In contrast, in the case of questions with verbs, such as "*novels written by John Updike?*", the query verb is enforced by matching the verb of the second or third predicate. The matched verb cannot, however, belong to a predicate posterior to the one that matches a query entity or a plural noun lemma. The next example illustrates this:

Terrorist (novel) = CONCEPT is the 22nd novel written by lauded author John Updike.
be(CONCEPT, 22nd novel)
write(novel,)
laud(, author John Updike)

In this strategy of inference, LiSnQA does not make allowances for questions composed simultaneously of both plural nouns and query entities, because we observed that the recognition of answers to this sort of question requires more than the first definition line. Simply put, the described inference procedure assists LiSnQA in determining a set $A_W(Q) \subseteq C_W(Q)$ of *reliable answers*. LiSnQA then extends $A_W(Q)$ by exploring the database of categories in two ways.

[5]It is worth noting that we show predicates as outputted by Montylingua. This means they are not manually corrected.

Firstly, if the query consists solely of a multi-word *focus*, then LiSnQA adds elements of the category that its label exactly matches the *focus*. If the question additionally has entities, LiSnQA accordingly adds elements of categories that match the *focus*, and an entity within the query or one of its rewritings. For example, the question "*Name Chuck Berry songs*" retrieves the following category:

CHUCK BERRY SONGS = {Around and Around; Back in the U.S.A.; Come On; I'm Talking about You;
Johnny B. Goode; Maybellene; Miracles; Memphis, Tennessee;
My Ding-a-Ling; Rip It Up; Rock and Roll Music; Roll Over Beethoven; Run Rudolph Run;
School Days;Sweet Little Sixteen; Too Much Monkey Business;
You Can't Catch Me;You Never Can Tell}

Since the label of the category matches the query *focus* and entity, all its members are added to $A_W(Q)$, and $C_W(Q)$. Here it is worth highlighting that not all answers in $A_W(Q)$ are included in the matched category. In the particular case of our working example, the *reliable answers* "*No Particular Place To Go*", and "*Promised Land*" are not members of the category "**CHUCK BERRY SONGS**". Secondly, LiSnQA adds concepts belonging to categories that contain a member in $A_W(Q)$, and their labels match a query entity or a query entity rewriting. It is worth duly noting here that all these new added answers do not necessarily have a definition page in Wikipedia. One last remark regarding categories is that Wikipedia does not provide categories for all possible list questions.

Then, LiSnQA searches for entries in the sandbox database corresponding to elements in $A_W(Q)$. In the set of found entries, LiSnQA looks for a common (higher frequent) property that matches an entity (or a rewriting) within the query. In our working category, the higher matching attribute is "**artist**", and has the value "**Chuck Berry**". In addition, LiSnQA obtains the two higher frequent types of sandbox in these entries, in the illustrative category: "**song**" and "**single**". Consequently, LiSnQA adds to $A_W(Q)$ entries in $C_W(Q) - A_W(Q)$ that their type match one of these two last values, and share the same value for the attribute "**artist**". Lastly, if one type of sandbox differs from the *focus* of the query, then it is seen as an *alternative focus*, in our working example, the *alternative focus* is "**single**".

B. Query Expansion

LiSnQA makes use of $A_W(Q)$ for extending the purpose-built search queries presented in [14]. This strategy considered the observation of [16], and hence, it searches for web pages predominantly entitled with query entities, NNPSs, NNPs, and documents containing in their body the query *focus*, nouns, and verbs as well. For instance, the next search query corresponds to the question "*Name Chuck Berry songs*":

intitle:("Chuck Berry") ∧ inbody:("songs")

From this first purpose-built query, a second and a third query are derived by appending hyponomic words to the *focus* as follows:

intitle:("Chuck Berry") ∧ inbody:("songs like" ∨ "songs including")
intitle:("Chuck Berry") ∧ inbody:("songs such as" ∨ "songs include")

In addition, a fourth search query is generated, which is aimed specifically at exploiting the content of on-line encyclopedias. In [14], each of these four submissions fetched a maximum of 20 snippets. In the present study, these four purpose-built queries are expanded with $A_W(Q)$. Since we have our databases taken from Wikipedia, we discard the fourth query, and we split the first submission into four similar submissions that retrieve ten snippets. Answers in $A_W(Q)$ are then allocated in these four queries according to their frequency in Google 5-grams[6]. To illustrate this, the following are two 5-grams with their corresponding frequencies, and partly or fully aligned Chuck Berry's songs:

213 Chuck/1 Berry/2 's/3 "/4 **Maybellene**/5
138 Chuck/1 Berry/2 's/3 "/4 **Sweet**/5 Little Sixteen

In bold is highlighted the aligned words. In this query expansion method, four aspects are vital: (a) the frequency estimate of each answer is the higher frequency of its partial or full alignments, when it is appended to the *focus* and to the query entities as shown in the previous illustration, (b) the hierarchy of answers is given by these frequencies, this means answers with high frequencies are allocated in the same search query, so do answers with low frequencies, (c) *reliable answers* with no frequency estimate are allocated alphabetically, and (d) the number of answers added to a query is limited by the length of queries accepted by search engines. In our illustrative example:

intitle:("Chuck Berry") ∧ inbody:("singles" ∨ "songs")
inbody:("You Never Can Tell" ∨ "Sweet Little Sixteen" ∨ "Roll Over Beethoven" ∨ "Maybellene")

intitle:("Chuck Berry") ∧ inbody:("singles" ∨ "songs")
inbody:("No Particular Place To Go" ∨ "Rock and Roll Music" ∨ "You Can't Catch Me" ∨ "Come On")

intitle:("Chuck Berry") ∧ inbody:("singles" ∨ "songs")
inbody:("Back in the U.S.A." ∨ "Too Much Monkey Business" ∨ "Run Rudolph Run" ∨ "Around and Around")

intitle:("Chuck Berry") ∧ inbody:("singles" ∨ "songs")
inbody:("School Days" ∨ "Promised Land" ∨ "Surfin' USA" ∨ "Memphis, Tennessee")

These queries also unveil that an *alternative focus* is concatenated with the query *focus*. In the case of the second and third original queries, if an alternative focus is discovered, these two queries are copied, and the query *focus* is replaced with the *alternative focus*. This generates four queries that are aimed at fetching ten snippets each. If no alternative focus is found, then the two copied queries are extended with *reliable answers*, similar to the previous four queries. If no answers

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

are discovered, and no *alternative focus* is found, then the four original queries are sent.

C. Name Rewritings Recognition

One of the main problems of answering list questions is that the type of the *focus* varies widely from one question to another. For instance, the query "*countries that produce peanuts*" has countries (locations) as the *focus*, but the question "*Name Chuck Berry songs*" names of songs. 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 in web snippets.

LiSnQA also uses concepts in $C_W(Q)$ for identifying names within web snippets. First, concepts, that are correctly aligned with web snippets, are mapped to a placeholder. Accordingly, their rewritings are also mapped to the respective placeholder. Second, like [14], LiSnQA uses regular expressions for discriminating names on the ground of punctuation and sequences of terms that start with a capital letter. The obtained names, and their corresponding rewritings, are also replaced with a placeholder. Here, the rewriting database assists LiSnQA in improving the name recognition by identifying different name instances. For example, "*Maybellene*" and "*Maybelline"* are mapped to the same placeholder.

D. Selecting Answers

In [14], answers were selected by inspecting name entities that share syntactic and semantic similarities. This method ranked answers candidates by measuring the semantic similarity to the prompted query of every context where these answers candidates occur. These similarities were computed by making use of the LSK proposed by [15]. LSK is computed from the terms-document matrix, which considers the submitted query as a *pseudo-document*. LiSnQA augments this matrix with the first line definitions respecting concepts in $C_W(Q)$, especially in $A_W(Q)$. This way LiSnQA tries to draw more accurate semantic inferences.

Additionally, [14] constructed a set of $R(Q)$ of reliable answers by examining whether or not any answer candidate occurs in two different coordinations within web snippets (coordinations signalled by the *focus* and hyponomic words). For instance, the next three snippets add the answers "*The Screwtape Letters*" and "*Mere Christianity*" to $R(Q)$:

C S Lewis An Assessment.
The Chronicles of Narnia, written by C. S. Lewis, between 1950 and 1956, is a series of seven fantasy...
Popular books, ***such as The Screwtape Letters****, have given many people ample material for thought and reflection...*

VQR " "
In recent years he has written articles for VQR about Abraham Lincoln, C. S. Lewis, Garrison Keilloir, Frank Sinatra, Ward....
The religion shelves will be chock full, of course-books ***like Mere Christianity****,* ***The Screwtape Letters****,* ***The Problem of Pain****, and* ***Miracles*** *continue to sell millions of copies each year...*

C. S. Lewis in Christian History Institute's Pages.
Lewis was one of the most influential Christian thinkers of the 20th century, the author of several apologetics books, ***including The Screwtape Letters*** *and* ***Mere Christianity****.*

In [14], $R(Q)$ was thereafter extended by discriminating extra reliable answers on the ground of their syntactic bonding with the query by inspecting their frequency given by Google 5-grams, similar to how LiSnQA obtains the frequencies to allocate elements in $A_W(Q)$ within the search queries. They then took advantage of $R(Q)$ for bootstrapping coordinations, and consequently, for inferring some low frequent answers surrounded by *reliable answers*. In the three previous working snippets, this boostrapping infers that "*The Problem of Pain*" and "*Miracles*" are also answers. This boostrapping strategy, however, misses answers in coordinations that do not share an answer with another coordination. The following is a good illustrative snippet:

C. S. Lewis Summer Institute, Williams College.
A Guide to the Fiction of C. S. Lewis (2006, 1987)], as well as numerous popular books ***including Splendor in the Ordinary****,* ***Christ the Tiger****,* ***Chance or the Dance****, and most recently,* ***Dove Descending****.*

LiSnQA aims to lessen this drawback, and thus it improves this inference process, by adding answers in $A_W(Q)$, that are in the fetched snippets, to $R(Q)$ and the final output. This way LiSnQA can distinguish extra reliable coordinations, and hence, infer additional answers.

V. Experiments

LiSnQA[7] was assessed by means of the widespread list question sets supplied by TREC from 2001 to 2004. Like [14], we use these standard question sets as reference questions-answers pairs, but contrary to TREC systems, we applied them to the web instead of the AQUAINT corpus.

The reason to left unconsidered questions corresponding to posterior tracks is that it is not possible to unambiguously determine the queries, because these questions are largely dependent upon their context. To illustrate this, consider the TREC 2005 list question "*Which countries expressed regret about the loss?*". Here, "*the lost*" refers to the previously set context: "*Russian submarine Kursk sinks*". The manual resolution of this kind of reference brings about an evaluation that cannot be straightforwardly compared with other systems.

Answer Recall

LiSnQA fetches a maximum of 80 web snippets (see section IV-B). Accordingly, a baseline (BASELINE-I) was implemented that also fetches a maximum of 80 snippets by submitting the original query Q to the search engine. In addition, in order to assess the improvement of the techniques

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

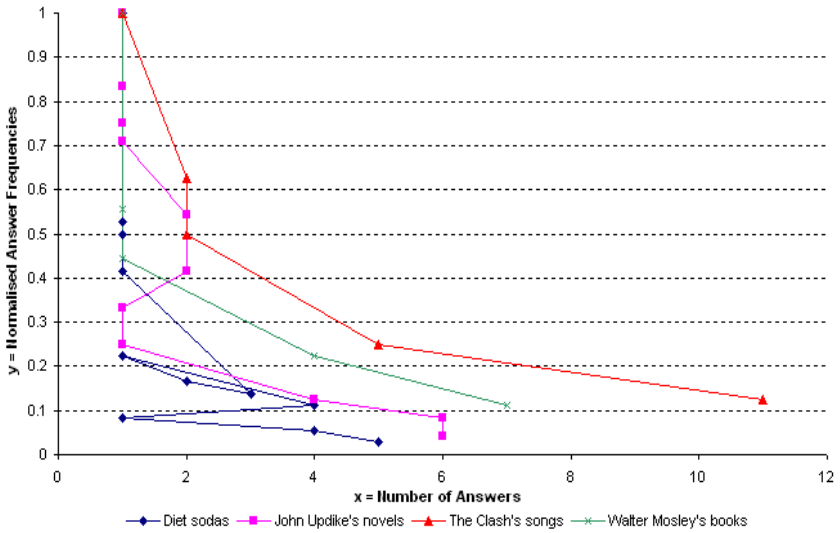

Fig. 1. Normalised answer frequencies vs. Number of answers.

proposed in the present work, we considered the strategy in [14] as a second baseline (BASELINE-II). The achievements for the four TREC datasets, are shown in table II.

TABLE II
TREC RESULTS (ANSWER RECALL).

	2001	2002	2003	2004
BASELINE (Recall)	0.43	0.49	0.4	0.65
BASELINE-II(Recall)	0.93	0.90	0.56	1.15
LiSnQA(Recall)	1.14	1.44	0.70	1.45
BASELINE (NoS)	77.72	77.33	80	78.87
BASELINE-II(NoS)	59.83	53.21	51.86	46.41
LiSnQA(NoS)	74.50	59.12	52.59	50.95
BASELINE (NAF)	2	4	8	12
BASELINE-II (NAF)	6	2	8	11
LiSnQA (NAF)	2	2	8	12

In table II, 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, LiSnQA retrieved slightly more snippets than BASELINE-II, and markedly increased the recall of distinct answers for all question sets. 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 two-fold: (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. The achieved improvement is due essentially to the fact that answers to list questions tend to co-occur at the sentence or paragraph level. Hence, retrieving pieces of texts containing one answer automatically increases the probability of fetching more answers. In the light of these results, intensifying efforts in discovering few reliable answers is therefore fairly positive for answering list questions.

Figure 1 depicts another significant aspect of the answers distribution within web snippets. This figure shows the number of answers that have a given frequency. In order to compare distributions corresponding to different questions, this graph shows normalised frequencies. Specifically, few answers tend to be characterised by a high frequency (points closer to $y = 1$), and most of the answers tend to get less than 20% of the frequency of these highest frequent answers. This beneficial aspect can help the answer extraction in two different ways: (a) to know beforehand that it is possible to select few, but promising, answer candidates to perform the query expansion after document retrieval, also known as *pseudo relevance feedback* (PRF), cf. [17], and (b) to shape or check the frequency distribution of the final set of answers. The former is also supported by the previously discussed fact that answers to list questions tend to co-occur in a small span of text, and the latter can assist in filtering out some -probably few- misleading answers.

Unfortunately, answers do not always have the distribution shown in figure 1. In some cases, this distribution is distorted by a small subset of answers that have a high co-occurence, sloping the curve or moving it towards a higher number of answers ($x > 1$) while approaching to $y = 1$ (see figure 2). The example in section IV-D is apt for illustrating this phenomenon. In this example, the answers "*The Screwtape Letters*" and "*Mere Christianity*" tend to have a high frequency and are likely to co-occur, bringing about the distribution in figure 2. Another factor that produces a similar effect is duplicate snippets, namely copies of coordinations with answers. If several duplicate -or not necessarily too exact- copies are fetched from the search engine, then this graph will show a similar distribution.

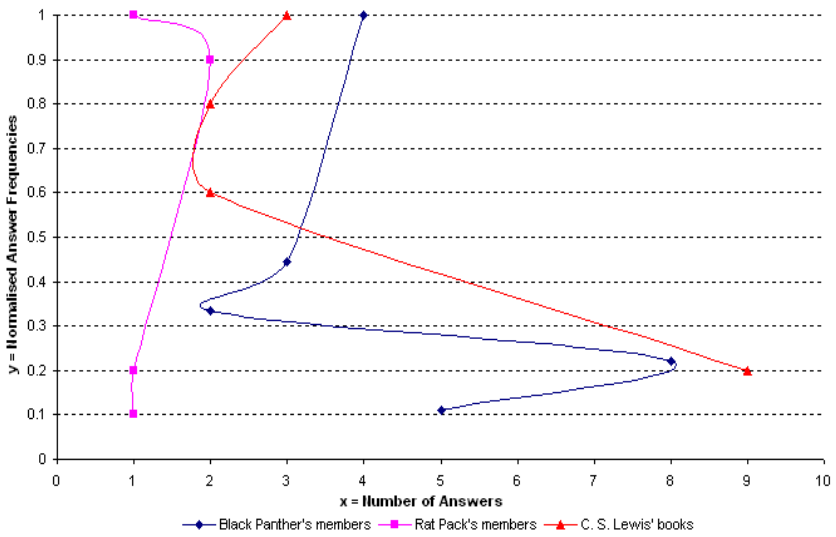

Fig. 2. Normalised answer frequencies vs. Number of answers (high frequent co-occurrence).

Another kind of distribution is due to list questions that have few answers. In this sort of query, all answers are usually conveyed in the same piece of text or sentence. Consider the answers to the query "*type of quarks*":

There are six different types of quark, usually known as flavors: up, down, charm, strange, top, and bottom.

Therefore, in this class of query, answers will tend to share a similar frequency count.

Given the results in table II, we can conclude that the first line database, taken from Wikipedia, supplied a positive way for inferring *reliable answers*, and thus extending the query. On the other hand, it makes the query expansion strategy largely dependant upon the coverage of Wikipedia, which is an undesirable and inescapable consequence. This coverage can be, nevertheless, extended by means of definitions extracted from the web. Here, we envisage that techniques developed in definition question answering on the web can be used for adding new concepts to this database (see for example [18]). Moreover, these techniques can also be applied to obtain alternative definitions of concepts in $C_W(Q) - A_W(Q)$. These alternative definitions can increase the probability of finding more concepts that fulfill the conditions of being a *reliable answer*, especially if $A_W(Q)$=$\emptyset$.

Indeed, the application of these web based techniques would involve the design of an information retrieval engine capable of handling these syntactic relations when indexing and searching as well as ranking. In particular, the resolution of aliases discussed in section III. Furthermore, this purpose-built information retrieval engine can also make allowances for the syntactic constructs used for rewriting the query (see section IV-B), this way it can rank and compute more suitable web snippets for answering list questions.

F_1 *score*

The widespread measure used for assessing list question answering systems is the F_1 score, and it is defined as follows [19], [20]:

$$F_1 = \frac{2 * IP * IR}{(IP + IR)}$$

The instance precision (IP) and instance recall (IR) are calculated as $\frac{D}{N}$ and $\frac{D}{S}$, respectively. Where D stands for the number of distinct correct answers returned by the system, N is the total number of responses returned by the system, and S is the number of correct answers in the target corpus. In the case of the present work, S is the number of distinct answers within the retrieved snippets.

`LiSnQA`: *Answer Extraction*

TABLE III
TREC FINAL RESULTS.

	2001	2002	2003	2004
`LiSnQA`(F_1)	0.40	0.37	0.29	0.32
`LiSnQA`(Acc.)	0.55	0.48	0.56	0.56
`BASELINE-II`(F_1)	0.35	0.34	0.22	0.30
`BASELINE-II`(Acc.)	0.5	0.58	0.43	0.47
`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

Table III compares the results obtained by `LiSnQA` with the top TREC systems and `BASELINE-II`. First of all, it is worth remarking that, 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. Second, table III shows that `LiSnQA` ranks between the **top one and**

TABLE IV

THREE EXAMPLES OF GOOD PERFORMANCE.

WHAT MOVIES DID JAMES DEAN APPEAR IN? (TREC 2004)

LiSnQA: F_1=0.706; S=6	BASELINE-II: F_1=0.6667; S=3
Has Anybody Seen My Gal, Treasury Men in Action, Warner Bros Studios Warner-Hollywood Studios, **East of Eden**, **Fixed Bayonets**, **Giant**, Paramount Studios, **Rebel Without a Cause**, **Sailor Beware**, Studio One, **Trouble Along the Way**	**Rebel Without a Cause**, **East of Eden**, Porsche

WHAT MAGAZINES DOES CONDE NAST PUBLISH? (TREC 2004)

LiSnQA: F_1=0.778; S=14	BASELINE-II: F_1=0.727; S=10
House Garden, L'Uomo, **Architectural Digest**, **Vanity Fair**, **Vogue**, **Wired**, **Bon Appetit**, Britain, Business Wire, Conde, **Domino**, Esquire, Forbes, **GQ**, **Glamour**, **Gourmet**, **Lucky**, **Traveler**, Maxim, Portfolio, **Allure**, **The New Yorker**	Forbes, Fortune, **GQ**, **Glamour**, **Gourmet**, **Domino**, **Vanity Fair**, **Vogue**, Topical News, Business Title, **The New Yorker**, **Traveler**

WHO ARE PROFESSIONAL FEMALE BOXERS? (TREC 2003)

LiSnQA: F_1=0.737; S=9	BASELINE-II: F_1=0.167; S=9
Ada Velez, **Christy Martin**, **Emiko Raika**, **Ann Wolfe**, **Laila Ali**, **Lucia Rijker**, **Marcela Acuña**, The Bear, Billie, Laila	Billie, **Lucia Rijker**, The Bear

two TREC systems in the first two question sets, while between the **second and the third** in the last two data sets. Although these top TREC approaches are not directly comparable, because they extracted answers from AQUAINT corpus whereas LiSnQA (and BASELINE-II) did it from the Web, the difference in performance is still very fair. Third, LiSnQA betters the performance of BASELINE-II in all question sets, specifically in the case of TREC 2003. In this particular case, LiSnQA obtained a F_1 score 31.81% higher than BASELINE-II, and 10% lower than the score obtained by the **system ranked second**. These results are encouraging, because LiSnQA makes allowances solely for **web snippets**, not for **full documents**. This remarks our highly promising results, especially considering other approaches [16], [11], 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.

TABLE V

NUMBER OF QUESTIONS ($D = 0$).

	2001	2002	2003	2004
LiSnQA	5 (2)	4 (2)	13(8)	18 (12)
BASELINE-II	7 (6)	7 (2)	19 (8)	23 (11)

Table V shows another interesting aspect: the number of questions ($D = 0$), for which both systems could not distinguish any answers. In this table, the number in parentheses corresponds to the respective values of NAF in table II. Results reveal that LiSnQA discovers at least one answer to more queries than BASELINE-II.

The enhancement achieved by LiSnQA is essentially due to two direct causes: (a) the *reliable answers* taken from Wikipedia, which assist LiSnQA in recognising and inferring answers that are hard to extract directly from the retrieved snippets, and (b) the compression of the final output by means of the identification of name rewritings. Certainly, localised contextual web snippets reduce the probability that misleading rewritings worsen the performance. More interestingly, list QAS can enrich the database of rewritings by searching for additional first definition lines on the web that convey alternative names. These rewritings can be found by looking for the target concepts along with clues that convey alternative names (see table I).

On the whole, LiSnQA fetches snippets containing **more answers** than BASELINE-II, but nonetheless, it finishes with a **higher** F_1 score. This proves that the presented strategies for mining Wikipedia directly contribute to answer list questions.

In order to give a sense of the performance of LiSnQA, table IV provides the output of three questions, where the system obtained positive results. In two out of these three cases, LiSnQA retrieved more answers than BASELINE-II, and in the three queries, it outperformed this baseline. In the third example, one can see that inexact answers, like "*Laila*" or "*The Bear*", still hurt the performance. These inexact answers come from truncations in web snippets and/or errors during name recognition.

For the purpose of fostering the research into list question answering on the web, table VI yields two cases, in which LiSnQA performs poorly. In both examples, LiSnQA fetched more answers, but it identified one answer less than BASELINE-II. Even though, in the second case, LiSnQA outputted a list substantially smaller than BASELINE-II, it performs poorer in terms of the F_1 score. Here, the performance of BASELINE-II is severely affected by the following inexact answers:

- "*Alibi*" $\Leftrightarrow$ "*A is for Alibi*".
- "*Burglar*" $\Leftrightarrow$ "*B is for Burglar*".
- "*Corpse*" $\Leftrightarrow$ "*C is for Corpse*".
- "*IS FOR*" matches all answers.

TABLE VI

TWO EXAMPLES OF BAD PERFORMANCE.

WHAT ARE 6 NAMES OF NAVIGATIONAL SATELLITES? (TREC 2001)

LiSnQA: F_1=0.286; S=10	BASELINE-II: F_1=0.6; S=6
Europe, **Galileo**, **GPS**, Orbit	Europe, **GPS**, **Galileo**, **Navstar**

WHAT ARE THE TITLES OF THE BOOKS IN SUE GRAFTON'S ALPHABET SERIES OF MYSTERIES? (TREC 2003)

LiSnQA: F_1=0.118; S=11	BASELINE-II: F_1=0.2; S=8
1992 On, A Handbook , MPR, **P is for Peril**, Writers of America, Kinsey Millhone	Alibi, Alphabet, **B is for Burglar**, BIRTHDAY, Bio-Farm, Burglar, **C is for Corpse**, Corpse, IS FOR, Kinsey Milhone A, Metroactive, Author

If this problem is solved, the value of N will decrease to 9, D will increase to 3, and consequently, the F_1 score will markedly increase from 0.2 to 0.353. Accordingly, we reasonably presume that dealing efficiently with this problem would lead into a substantial increase in performance. In particular, it will clear the output of some answers that might give the user a negative impression about the system.

VI. CONCLUSIONS AND FUTURE WORK

Our key findings are interesting, in particular the fact that short definition descriptions yield useful information to discriminate concepts that have **strong** evidence of being answers to a list question. This also helps to partially overcome the drawback that not all answers are conveyed on listings. Furthermore, our results also corroborate that it is perfectly feasible to discover answers to some list questions within web snippets.

We envisage that these answers will help to select the most promising documents, and afterwards, detecting the portions (like lists or coordinations) where these answers are. In addition, we envision that our off-line databases, taken from Wikipedia, can be enriched and extended by strategies developed in web definition question answering. For instance, by discovering extra pairs of alternative names across the web, this way the coverage can be considerably widen.

Due to the fact that quantitative evaluations has been a driver of advances in language technologies during the last decades, it is important in the future to account for a test collection of questions and answers extracted from the web. In this way, one can reduce the uncertainty as to whether or not at least one answer can be found on the web for every question, thereby ensuring a fairer, and more reliable as well as stable comparison of systems.

ACKNOWLEDGMENTS

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

[1] B. Katz and J. Lin and D. Loreto and W. Hildebrandt and M. Bilotti and S. Felshin and A. Fernandes and G. Marton and F. Mora, "Integrating web-based and corpus-based techniques for question answering," in *Proceedings of TREC 2003*, 2003.

[2] B. Katz and M. Bilotti and S. Felshin and A. Fernandes and W. Hildebrandt and R. Katzir and J. Lin and D. Loreto and G. Marton and F. Mora and O. Uzuner, "Answering multiple questions on a topic from heterogeneous resources," in *Proceedings of TREC 2004*, 2004.

[3] B. Katz and G. Marton and G. Borchardt and A. Brownall and S. Felshin and D. Loreto and J. Louis-Rosenberg and B. Lu and F. Mora and S. Stiller and O. Uzuner and A. Wilcox, "External Knowledge Sources for Question Answering," in *Proceedings of TREC 2005*, 2005.

[4] L. Wu, X. Huang, Y. Zhou, Z. Zhang, and F. Lin, "FDUQA on TREC2005 QATrack," in *Proceedings of TREC 2005*, 2005.

[5] M. Hearst, "Automatic Acquisition of Hyponomys from Large Text Corpora," in *Proceedings of the Fourteenth International Conference on computational Linguistics*, 1992.

[6] M. Hearst, "Automated Discovery of WordNet Relations," *WordNet: An Electronic Lexical Database and some of its Applications*, 1992.

[7] R. Sombatsrisomboon, P. Matsuo, and M. Ishizuka, "Acquisition of hypernyms and hyponyms from the www," in *Proceedings of 2nd International Workshop on Active Mining*, 2003.

[8] K. Shinzato and K. Torisawa, "Acquiring hyponymy relations from web documents," in *Proceedings of HLT-NAACL 2004*, 2004, pp. 73–80.

[9] K. Shinzato and K. Torisawa, "Extracting hyponyms of prespecified hypernyms from itemizations and headings in web documents," in *Proceedings of 20th International Conference on Computational Linguistics*, 2004.

[10] H. Yang and T. Chua, "FADA: find all distinct answers," in *Proceedings of WWW Alt. '04'*, 2004, pp. 304–305.

[11] H. Yang and T. Chua, "Web-based List Question Answering," in *Proceedings of Cooling '04*, 2004.

[12] A. Figueroa and G. Neumann, "Mining Web Snippets to Answer List Questions," in *AI07: the 2nd International Workshop on Integrating AI and Data Mining*, 2007.

[13] S. Cederberg and D. Windows, "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)*, 2003, pp. 111–118.

[14] A. Figueroa and G. Neumann, "Finding Distinct Answers in Web Snippets," in *WEBIST 2008: 4th International Conference on Web Information Systems and Technologies*, 2008.

[15] J. Shawe-Taylor and N. Cristianini, "Kernel methods for pattern analysis," in *Cambridge University Press*, 2004, pp. 335–339.

[16] H. Yang and T. Chua, "Effectiveness of Web Page classification on Finding List Answers," in *Proceedings of SIGIR '04, Sheffield, United Kingdom.*, 2004.

[17] R. K. Belew, *Finding out About: A Cognitive Perspective on Search Engine Technology and the WWW*. Cambridge University Press, 2000.

[18] A. Figueroa and G. Neumann, "A Multilingual Framework for Searching Definitions on Web Snippets," in *KI*, 2007, pp. 144–159.

[19] E. M. Voorhees, "Overview of the TREC 2003 Question Answering Track," in *Proceedings of TREC 2003*, 2003.

[20] E. M. Voorhees, "Overview of the TREC 2004 Question Answering Track," in *Proceedings of TREC 2004*, 2004.