Adaptive Algorithm for Interactive Question-based Search

Jacek Rzeniewicz[1], Julian Szymański[1], and Włodzisław Duch[2]

[1] Department of Computer Systems Architecture,
Gdańsk University of Technology, Poland,
julian.szymanski@eti.pg.gda.pl, jrzeniewicz@gmail.com
[2] Department of Informatics, Nicolaus Copernicus University, Toruń, Poland
Google: W. Duch

Abstract. Popular web search engines tend to improve the relevance of their result pages, but the search is still keyword–oriented and far from "understanding" the queries' meaning. In the article we propose an interactive question-based search algorithm that might come up helpful for identifying users' intents. We describe the algorithm implemented in a form of a questions game. The stress is put mainly on the most critical aspect of this algorithm – the selection of questions. The solution is compared to broadly used decision trees learning algorithms in terms of knowledge inconsistencies tolerance, efficiency and complexity.

Keywords: information retrieval, semantic search, decision trees

1 Introduction

Recent years we have observed semantic technology slowly making its way into the web search [1]. Result pages of the most popular searchers like Google, Yahoo or Bing, provide us not only with references to the websites containing desired keywords, but also with additional information on the entities those sites represent. Hints with similar queries are displayed to the users as they type, helping them to express themselves more precisely and receive more relevant results. Finally, synonyms are used to broaden the search.

Moreover, a number of native semantic searchers has been launched last years eg. Duckduckgo, Hakia. However, despite offering categorization of results by context (eg. Yippy!), that aims to lead to better relevance, they generally haven't attracted wide audience yet. Finally, the question answering field has been developed significantly [2]. A number of both general (eg. Yahoo! Answers, Ask.com) and vertical (eg. StackExchange) Q&A engines are available on the Web, offering billions of answers.

On the other hand, broadly used search engines still cannot be considered as semantic in terms of "understanding" users' queries. The same query formulated different ways is usually served with different results [3]. Search is still mostly keyword-oriented. Users need to know proper words related to the subject in order to set the searcher on the right track and obtain relevant results. This is

easy when user is searching the Web for a famous artist or a nearby restaurant, but gets tricky if he or she doesn't know any keywords (index phrases) that are distinctive for the desired entity. Let us consider this example: you saw a suricate in a ZOO and after returning home you want to learn more about it on the Web. The problem is, you can't recall its name and any keywords you can think of relate only to its appearance. You will spend a few minutes adjusting those keywords and clicking result links forth and back, or scrolling image results until you come across some information about suricates. When lacking proper keywords, users are required to devote a significant amount of time and effort looking for the information. That work should be performed by the search engine.

When imagining a perfect semantic searcher, it is handy to consider the notion of a Human Search Engine (HSE) [3]. This is a hypothetical searcher acting like a human being in terms of understanding the query. Back to the suricate example – how could the HSE help the user to find desired information? We imagine it being able to direct the search by prompting the user to provide more specific details. This way, the stress would be put less on the keywords selection and more on what the user interacting with the search engine has in mind – meaning of the query and particularly relevant answer in the form of provided search results.

Some of today's searchers limitations might be overcome by a web searcher displaying the ability to take over the initiative and ask questions in the form of simple dialogues. Such an approach requires an effective algorithm: one that asks the minimum number of questions to recognize the appropriate meaning of the user's query.

In this article we address just that: a question-based search algorithm, especially its critical aspect – the selection of questions. The structure of this paper is as follows: in section 2 we introduce a question-based search model in a form of a word game. In section 3 the question-based search algorithm is described and compared to typical decision trees algorithms. We also gave an outline of our future plans in section 4.

2 Questions game – a computational linguistic challenge

A simple model of dialogue–based search can be approximated in the form of a game of questions. This is a word game where one player thinks of some concept and the other one asks a series of yes/no questions regarding that object. After at most twenty questions the latter guesses what was the concept his partner was thinking of.

The question-based search algorithm described in the following sections was implemented in the form of a questions game that is available on the Internet at `http://kask.eti.pg.gda.pl/winston`. There users can play the questions game with an avatar called Winston. The avatar took the role of asker and guesser, site visitors need to pick up a concept and answer avatar's questions. The core of the knowledge base was imported from the WordNet dictionary [4].

To test the approach we limit the domain to the information related to the animals branch of WordNet.

Apart from providing the means for the search algorithm evaluation, the questions game has another application. Upon completing each game, some knowledge can be added to the system. This is not only limited to the user answers during the game. After clearing up what the subject of the game was, the player is requested to answer yet another question: *what can you tell me about the [subject]?* User can reply to that by typing in a fact in a form of *object–relation–feature*. For example, to make the fact that suricates are burrowing animals come across, one can type *can* and *burrow* into the text fields. This way both the amount of assertions and the system's vocabulary expand.

3 The question-based search algorithm

In the first section the need for a question-based search algorithm has been justified. The task of assigning a proper meaning to user's input generally relates to the classification problem. However, before any algorithm is discussed, a few assumptions regarding the problem's conditions ought to be stated:

1. Matching target object may not be present in the system.
2. System's knowledge about given concept may be incomplete or partially incorrect.
3. Knowledge base may contain false assertions or represent a different point of view than user's.

3.1 Decision trees

One of the machine learning approaches that can be adopted to the search problems are decision trees algorithms. There are many of their variants: ID3 [5], C4.5 [6] or latest successor See5/C5.0 [7]. Those algorithms provide efficient mathematical model, so it requires a little number of steps and computing power to finalize the search: using binary decisions in nodes it is possible to index over one million classes performing only 20 tests. However, there are strong disadvantages to the decision trees approach when confronted with the stated assumptions. The process of searching develops towards a single hypothesis. Usually it is unlikely that there will be a target concept that matches given input very well. Thus we are interested in receiving a set of most matching answers rather than just one. Decision trees don't facilitate that. Neighbor of the top matching leaf is not necessarily the second best match.

Another crunch with decision trees is that the search paths are only optimized globally. According to the assumptions made in the beginning of this section, it can happen that the system's and user's knowledge differs. Occurrence of an answer that does not conform to the system's knowledge makes the decision tree take a wrong path, which must end up fatal for the overall search.

The next issue is that for a construction of a typical decision tree we usually have many examples and only a few classes. In the case of web search the problem

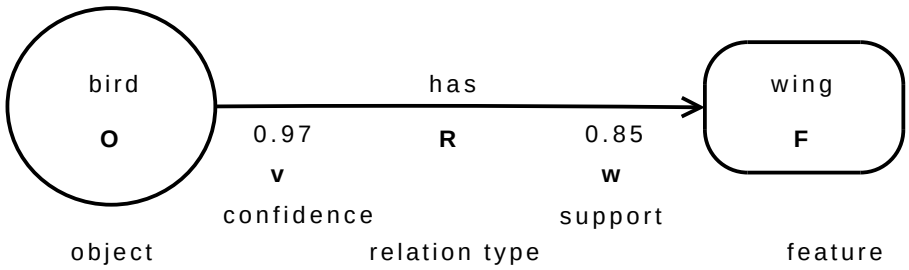

Fig. 1. Example assertion in the *vw*ORF notation.

is opposite: there are many potential classes (web pages relevant to user query) and each of them is described with a single example.

3.2 Knowledge representation for the question-based search algorithm

The approach presented here is mainly based on fuzzy semantic network as an underlying knowledge model [8]. Specifically, the network consists of concepts interconnected with relations by connections called triples of knowledge, expressed in the *vw*ORF notation:

O – the object described
R – the type of relation, denoting how the object is related to the feature
F – the feature expressing given property of the object
v – confidence: the real number in the range $\langle 0, 1\rangle$ describing, how this atom of knowledge is reliable. 0 denotes no certainty at all, 1 means the information is sure.
w – support: the real number in range $\langle -1, 1\rangle$ describing, to what extent the feature is related to the object. Allows to express adjectives such as "always" (1), "frequently" (˜0.5), "seldom" (˜-0.5) or "never" (-1).

Figure 1 depicts an example assertion expressed in the *vw*ORF notation. Here the fact that birds have wings was associated with parameters support $w = 0.85$ and confidence $v = 0.97$. This means that a bird most usually has a wing and that this assertion is almost sure to be true. The *vw*ORF notation is capable of expressing elementary sentences. More complex assertions, e.g. that birds usually have two wings, cannot be expressed this way, but this could be enabled by allowing a whole triple on the "object" side of another triple.

As mentioned before, the questions game that is running the system is not only used to evaluate the efficiency of the search algorithm, but also to provide new assertions to the knowledge base. Therefore it is necessary to update triple's w and v parameters whenever a new assertion related to this triple appears. The support value is a simple weighted mean of individual assertions:

$$w = \frac{\sum_{i=1}^{N} |\alpha_i| \cdot \alpha_{i_w}}{\sum_{i=1}^{N} |\alpha_i|} \tag{1}$$

where:

N – total number of assertions

$|\alpha_i|$ – weight of the i-th assertion

α_{i_w} – support of the i-th assertion

and the confidence value is a function of the weighted standard deviation of individual assertions:

$$\sigma^2 = \frac{\sum_{i=1}^{N} |\alpha_i| \cdot (w - \alpha_i)^2}{\sum_{i=1}^{N} |\alpha_i|} \tag{2}$$

$$v = \frac{1}{2} \cdot (1 + cos(\pi\sigma)) \tag{3}$$

where σ is a non–negative value denoting the weighted standard deviation.

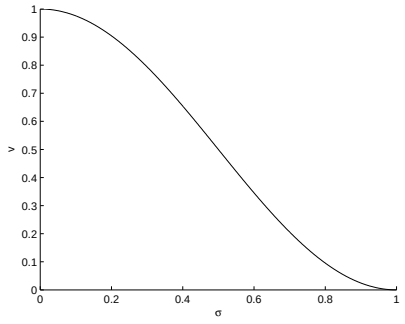

Fig. 2. Relationship between assertions standard deviation and the triple's confidence.

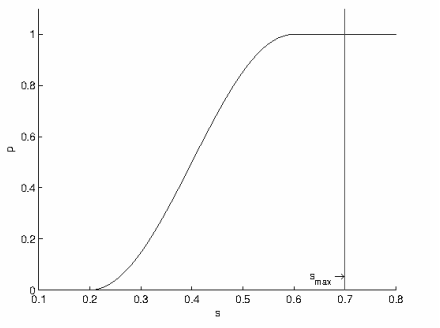

Fig. 3. CDV's probability of inclusion into S depending on similarity to ANSV vector.

In the Figure 2 the relationship between the standard deviation and confidence is presented. When all the individual assertions indicate equal feature support, the standard deviation is 0 and the confidence value equals 1. The confidence decreases as standard deviation goes up for incoherent assertions; in a boundary case of opposite assertions (such of support values 1 and -1), standard deviation approaches 1 and confidence tends to 0.

The semantic network is an economic (in terms of cognitive economy [9] that allows to reduce redundancy) and powerful model [10] that allows flexible complex model utterances. Such knowledge representation is easily browsable for humans using interactive graphs [11] and convenient to process, i.e. to perform inferences based on relation types interpretation that are related to the edges. On the other hand, only little amount of the overall information stored in the semantic network is stated explicitly and therefore it is hard to do any live computations on it. For this reason, for the sake of making the search computations easier, the knowledge is translated into the form of a semantic space.

The semantic space is an N-dimensional space where one dimension is associated with a unique pair (*relation*, *feature*). Those pairs are referred to as questions. Each concept in the semantic space is represented by a Concept Description Vector (CDV) [12]. Every element e of each CDV holds two real numbers e_w and e_v denoting respectively support of the connection and the confidence of knowledge. Those values are copied from corresponding triple if exists or inherited from another concept, provided that there is a positive support triple with relation IS_A pointing to that concept.

3.3 The algorithm

The search algorithm operates on the semantic space constructed from the semantic network. Over the consecutive iterations, a subspace S of candidate vectors is maintained. Any CDV can enter or leave S in between two iterations. The answers that user gives build up a vector ANSV. Each time a new answer appears, similarity s between ANSV and every CDV is computed:

$$\cos(\phi) = \frac{\sum_{q:ANSV[q]\neq 0} CDV[q] \cdot ANSV[q]}{\sqrt{\sum_{q:ANSV[q]\neq 0} CDV[q]} \cdot \sqrt{\sum_{q:ANSV[q]\neq 0} ANSV[q]}} \tag{4}$$

$$s = \max(\cos(\phi), 0) \tag{5}$$

where $CDV[q]$ denotes the confidence and support product of CDV's element for dimension q. Applying the max function in the end of calculations puts similarity into the range $\langle 0, 1\rangle$, which makes computations more convenient.

The main flow of the question-based search algorithm is as follows:

```
1. Include each CDV into S.
2. Ask the most informative question according to S.
3. Obtain an answer to the question and update ANSV.
4. Compute similarity between every CDV and ANSV.
5. If maximum number of questions has been asked:
   return vector with the highest similarity to ANSV.
6. Rearrange S so that it include most similar vectors.
7. Go back to step 2.
```

Note that upon adding a new answer to the ANSV, all vectors' similarities are updated, even those excluded from S. This way no concept can ever be permanently eliminated, even if it doesn't go well with answers in the beginning.

It is also more accurate to always take all possible dimensions into consideration when evaluating a CDV. In practice it often happens that system asks questions that are unrelated to what the user thinks of. But somehow even those seemingly unrelated and unspecific features can help a lot to identify the right object [13,14]. Global evaluation of a vector guarantees the most accurate results.

3.4 Selection of a question

In each iteration system picks a question that is the most informative across the candidates subspace S. Any CDV is included into S with probability p:

$$p = \begin{cases} 0, & \text{if } s \leq s_l \\ \frac{1}{2} - \frac{1}{2}\cos\left(\pi \frac{s - s_l}{s_u - s_l}\right) & \text{if } s_l < s < s_u \\ 1, & \text{if } s \geq s_u \end{cases} \tag{6}$$

where:

s_u – safety threshold parameter equal to $s_{max} - 0.1$
s_l – rejection threshold parameter equal to $s_{max} - 0.5$
s_{max} – highest similarity between CDV and ANSV in this iteration

Figure 3 presents a sample plot of inclusion probability.

Once the subspace S is defined, information gain IG for each question q can be computed:

$$IG = -\frac{\omega_\oslash}{\omega} \cdot \left(\frac{\omega_\ominus}{\omega_\oslash} \log_2 \frac{\omega_\ominus}{\omega_\oslash} + \frac{\omega_\oplus}{\omega_\oslash} \log_2 \frac{\omega_\oplus}{\omega_\oslash} \right) \tag{7}$$

where:

ω – total weight of all CDV in S
$\omega_\oslash$ – total weight of all CDV in S such that $CDV\,[q] \neq 0$
$\omega_\oplus$ – total weight of all CDV in S such that $CDV\,[q] > 0$
$\omega_\ominus$ – total weight of all CDV in S such that $CDV\,[q] < 0$

and a similarity between CDV and ANSV is considered CDV's weight.

The question-based search algorithm described above to some extent can be considered opposite to the decision trees approach. As pointed out before, decision trees return only a single concept, because they are optimized globally and may very often reject correct branches due to local in-compliances between user's input and knowledge base state. Presented algorithm is more careful: in each step we process all possible concepts and evaluate them according to all possessed information. Any rejections are valid only within a single iteration, which eliminates the risk of decisions. At any time the algorithm can deliver a collection of best matching results, precisely defining how much they relate to the user specifications.

On the other hand, such approach lacks significant features that searching in decision trees has. Every step of the algorithm requires considerable amount of computations. Specifically, for a semantic space of N vectors and M questions iteration cost consist of:

- CDV's similarity update: $O\,(N)$ (assuming ANSV iteration complexity $O\,(1)$)
- candidates subspace S re-arrangement: $O\,(N)$
- IG's update: $O\,(N \cdot M)$

which results in the overall iteration complexity of $O\,(N \cdot M)$, a considerable requirement compared to decision trees' $O\,(1)$.

Second major drawback lies in the efficiency of the algorithm. Maintaining a broad set of candidate objects is safe and guarantees each concept a fair chance to be found in the assumed unstable conditions. Consequently, it can happen that obviously incorrect concepts are considered candidates.

Finally, it is noticeable that the algorithm presented here rejects to process according to common sense hierarchy. Assume there is a knowledge base containing 10 dog concepts and 100 related to cats. Human would most likely ask whether the concept is a cat or a dog. The algorithm promotes high-cardinality branches over those less numerous, and such a behaviour is unjustified.

Described question-based search algorithm overcomes significant drawbacks that searching in decision trees has. On the other hand, it introduces a cost of computation complexity and efficiency fall. In the next part of this section we address those issues.

3.5 Question asking strategies

Certain patterns of the system state that may result in algorithm's incorrect behaviour were distinguished in previous section. For each of these patterns, there is a strategy determining how to select a question:

1. A gap between the best– and the second best–matching concept: probability of inclusion of a CDV to the candidates subspace S depends on the difference between its similarity to the ANSV and maximum similarity in current step. Therefore it is unfortunate when incorrect concept remains top-matching for a number of iterations, since it worsens other concepts' odds. Appropriate strategy that should be applied here is to ask user about a feature that relates to that object's specific feature.
2. Numerous group of best-matching concepts: it happens that there is a whole group of very similar objects at the top of the candidates list. The algorithm is then unable to select a question separating this group since it maintains a broad candidates subspace S. In such case, S should be reduced only to the size of those top–matching set.
3. Mutually exclusive questions asked one after another: system happens to ask questions inconsistent with the latest answer. In example, once user gives positive answer to "is it a feline?" question, candidates subspace can still contain vectors describing dogs. Therefore a question "is it a canine?" is likely to be asked next. To avoid such contradictions, the following strategy was implemented: if user answers "yes" to a question q containing relation "IS_A", then the candidates subspace S should only be composed of such CDV's that $CDV\,[q] > 0$. This way, in the scope of one step, the process resembles searching in a decision tree built on the hypernym relationship. The quality of avoiding incorrect rejections is abandoned, but chances for fast identification of specific features increase instead.

The search algorithm has been rearranged to allow applying the most appropriate strategy in each step. Previous logic of question selection was also encapsulated into a strategy. Algorithm's step 2 has been defined as follows:

```
a. Instantiate each strategy.
b. Rank each strategy according to the state of the system.
c. Draw one strategy with probability proportional to its rank.
d. Ask a question returned by the selected strategy.
```

The rank of a strategy depends on the distribution of similarities between CDV's and ANSV, the index of current iteration and the previously picked strategies. Actual question selection is delegated to the picked strategy.

4 Discussion and future plans

We have shown that typical approach used in decision trees are not suitable for applications where uncertainties and inconsistencies happen both on the knowledge base level and on the user's side. Our proposed algorithm proved the ability of being resistant to such inconsistencies. On the other hand, it introduces a demand for additional computations and is less efficient than typical decision tress.

Several behaviour strategies have been implemented into the algorithm, and presented adaptive approach improves algorithm's efficiency significantly. It also demonstrates the flexibility of the algorithm – we showed that it is possible to apply a strategy that temporarily turns system's operation into decision trees alike. It seems that by tweaking this adaptive approach specifics we can achieve a high search efficiency in the end.

Our implementation employs lexical database provided by WordNet. The initial results obtained by the algorithm strongly depend on the knowledge quality stored in this repository. It should be stressed that the system interaction with the humans allows to introduce new, and correct already possessed knowledge. We employ this approach to crowdsourcing lexical knowledge [15] and extending the Wordnet by large amount of interactions with people playing the word game.

During the following months we will attempt to overcome major remaining problems. The algorithm promotes branches of higher cardinality over smaller ones – this characteristic may be removed by filtering out the concepts that are too specific. This way, also the amount of computations will decrease. The scheme of assigning weights to answered questions can also improve the relevance of candidate concepts. Finally, implementation of more accurate behaviour strategies will increase the algorithm's efficiency as well. We believe that high flexibility of the approach we work on allows finding the right balance between limiting incorrect rejections and search efficiency.

Successful application of the presented approach has served us as a test bed of the question-based search algorithm. Implementation in limited domain of animals, employing knowledge from the WordNet dictionary, shows the ability of specifying the search according to interaction with the user. Now we scale up the solution and apply the algorithm for improvement the search in Wikipedia. For now we have created a prototype system for refining the query based search results using the category system in Polish and Simple English Wikipedia. The first results, deployed under `http://bettersearch.eti.pg.gda.pl` are promising, and they indicate the proposed method can be applied for improvement of

information retrieval precession.

ACKNOWLEDGMENTS: The work has been supported by the Polish Ministry of Science and Higher Education under research grant N N516 432338.

References

1. R. Studer and YU Yong. Editorial-special issue semantic search. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(4), 2012.
2. P. Forner, A. Peñas, E. Agirre, I. Alegria, C. Forăscu, N. Moreau, P. Osenova, P. Prokopidis, P. Rocha, B. Sacaleanu, et al. Overview of the clef 2008 multilingual question answering track. *Evaluating Systems for Multilingual and Multimodal Information Access*, pages 262–295, 2009.
3. Alessio Signorini and Tomasz Imielinski. If you ask nicely, i will answer: Semantic search and today's search engines. In *Proceedings of the 2009 IEEE International Conference on Semantic Computing*, pages 184–191, Washington, DC, USA, 2009. IEEE Computer Society.
4. G.A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.
5. J.R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.
6. Tom M. Mitchell. *Machine Learning*. McGraw Hill, 1997.
7. Ross Quinlan. Data mining tools see5 and c5.0, March 2012. `http://www.rulequest.com/see5-info.html`.
8. J. Szymański and W. Duch. Information retrieval with semantic memory model. *Cognitive Systems Research*, 14:84–100, 2012.
9. C. Conrad. *Cognitive economy in semantic memory.* American Psychological Association, 1972.
10. J.F. Sowa. *Principles of semantic networks*. Morgan Kaufmann Publishers, 1991.
11. Julian Szymański. Cooperative wordnet editor for lexical semantic acquisition. In *Knowledge Discovery, Knowlege Engineering and Knowledge Management*, volume 128 of *Communications in Computer and Information Science*, pages 187–196. Springer Berlin Heidelberg, 2011.
12. W. Duch, J. Szymański, and T. Sarnatowicz. Concept description vectors and the 20 question game. *Intelligent Information Processing and Web Mining*, pages 41–50, 2005.
13. Peter Eckersley. How unique is your web browser? In Mikhail Atallah and Nicholas Hopper, editors, *Privacy Enhancing Technologies*, volume 6205 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2010.
14. Latanya Sweeney. Simple demographics often identify people uniquely. Technical report, Carnegie Mellon University, School of Computer Science, Data Privacy Laboratory, 2000.
15. J. Szymański and W. Duch. Context search algorithm for lexical knowledge acquisition. *Control & Cybernetics*, 41(1), 2012.