Genetic Algorithms for data-driven Web Question Answering

Alejandro G. Figueroa and Günter Neumann {figueroa|neumann}@dfki.de
DFKI LT-Lab, Stuhlsatzenhausweg 3, D - 66123, Saarbrücken, Germany

Abstract

We present[1] an evolutionary approach for the computation of exact answers to Natural Languages (NL) questions. Answers are extracted directly from the N–best snippets, which have been identified by a standard web search engine using NL questions. The core idea of our evolutionary approach to web question answering is to search for those substrings in the snippets, whose contexts are most similar to contexts of already known answers. This context model together with the words mentioned in the NL question are used to evaluate the fitness of answer candidates, which are actually randomly selected substrings from randomly selected sentences of the snippets. New answer candidates are then created by applying specialised operators for crossover and mutation, which either stretch and shrink the substring of an answer candidate or transpose the span to new sentences. Since we have no predefined notion of patterns, our context alignment methods are very dynamic and strictly data-driven. We assessed our system with seven different data sets of question/answer pairs. The results show that this approach is promising, especially when it deals with specific questions.

Keywords

Genetic algorithms, Question Answering, Web Mining, Natural Language Processing.

1 Introduction

Question Answering Systems (QAS) try to find exact answers to natural language questions submitted by users, by looking for answers in a set of available information sources, which can be spread on a single machine or all over the Internet, in particular, Web QAS focus solely on extracting answers from the whole Web.

In most cases, a natural language question represents some sort of close relation between entities, where the answer is the missing part of this relation: one or more entities, or the kind of relation that entities hold (Lita and Carbonell (2004)). To illustrate this, consider the following relation between two sorts of entities:

Inventor *invented* ***Invention***

In this instructive example, "*inventor*" and "*invention*" are entities, and the relation they hold is "*invented*". The missing part of the relation can be one or more entities:

Who invented ***Invention****?*
What invented ***Inventor****?*

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

In the same way, the missing element may aim at the relation that entities hold:

*How are **Inventor** and **Invention** related?*
*What is the relation between **Inventor** and **Invention**?*

The Web Question Answering Problem can then be viewed as the problem of searching for a set of strings that represent correct answers to a query Q, in other words, a set of strings that are likely to be the missing member of the relation established by the prompted question.

To begin with a formal description, consider S to be a set consisting exclusively of the σ different sentences extracted from a set φ of N snippets, S_s is the s-th sentence in σ, $1 \leq s \leq \sigma$. Let us also consider $B_{sk_1k_2}$ as an *n-gram* (subsequence of n terms from a given sequence) of $\beta = k_2 - k_1 + 1$ words in S_s which starts at position k_1 and ends at position k_2, $len(S_s) \geq k_2 \geq k_1 \geq 1$. If $k_1 = k_2$, $B_{sk_1k_2}$ is an *uni-gram*. The **Web Question Answering Problem** consists then in finding the *n-gram* that successfully fulfils:

$$max\ \ K(B_{sk_1k_2}, Q) \qquad (1)$$

Where K is a function which states how likely the *n-gram* represents a right answer to Q. In other words, how likely $B_{sk_1k_2}$ plays the role of the missing part of the relation established by the query. It seems that there is no standard function K that effectively solves this problem for any kind of question Q and any given set of *n-grams*. To illustrate this model, consider the following set of sentences:

$$S = \{S_1 = \text{“\textit{Igor Sikorsky invented the helicopter}”},$$
$$S_2 = \text{“\textit{The tea bag was invented by Thomas Sullivan}”}\}$$

Some *n-grams* extracted from S are: B_{112}="*Igor Sikorsky*", B_{223}="*tea bag*" and B_{278}="*Thomas Sullivan*". Usually, $B_{sk_1k_2}$ should not contain a set of Q^* banned terms, which are in the submitted query or it should not be in the stop list [2] ϱ of the language:

$$B_{sk\prime k\prime\prime} \notin Q^*, \quad \forall\, k\prime,\ k\prime\prime,\ k_1 \leq k\prime \leq k\prime\prime \leq k_2$$
$$B_{sk\prime k\prime} \notin \varrho, \quad \forall\, k\prime,\ k_1 \leq k\prime \leq k_2$$

This conventional assumption is due to the fact that elements of a *stop-list* or from the query are unlikely to be the missing part of the relation established by the prompted question. In the illustrative example, B_{255}="*invented*" and B_{144}="*the*" can not be considered acceptable answers, because "*invented*" belongs to the set of banned terms Q^*, and "*the*" belongs to the stop-list ϱ.

In general, Web QAS have two major components (Echihabi and Marcu (2003)):

1. A search engine which retrieves a set of promising documents from the collection along with a brief description of relevant passages called *snippets*.

2. An answer extraction module which gets answers from relevant documents and/or snippets.

The former necessarily involves the efficient indexing of documents and the design of a fast algorithm that computes snippets. The latter has to do with identifying

[2]A stop list consists entirely of extremely common words which are useless to index documents such as *articles*, *adverbials* or *adpositions*.

correctly the answer to the request of the user on the previously selected set of documents. For the efficiency sake, extracting answers straightforwardly from snippets is clearly desirable, in that way, QAS avoid downloading and processing a wealth of documents. Certainly, this is not an easy task. On the one hand, snippets provide: (a) localised contextual paragraphs that are highly related to the query, (b) these localised contextual paragraphs express ideas and concepts by means of different paraphrases, which consist primarily of morphological, semantical, orthographical and syntactical variations of these ideas and concepts (Savary and Jacquemin (2000)), which makes easier the identification of promising answer candidates. On the other hand, search engines insert intentional breaks in snippets, in order to show relations amongst words relevant to the query, which are separated by a large span of text. This makes snippets ungrammatical, and therefore, the answer extraction task and the identification of contextual cues is more difficult and dependent upon the algorithm that computes snippets. To illustrate this, consider the following question and set of retrieved snippets as an example: "*When was Albert Einstein born?*"

1. The nobel prize of physics Albert Einstein was born in 1879 in Ulm, Germany.
2. Born: 14 March 1879 in Ulm, Württemberg, Germany.
3. Physics nobel prize Albert Einstein was born at Ulm, in Württemberg, Germany, on March 14, 1879.
4. Died 18 Apr 1955 (born 14 Mar 1879) German-American physicist.
5. Briefwechsel Einstein / Born 1916 - 1955, Albert Einstein, Hedwig Born, ... Kunden, die Bücher von Albert Einstein gekauft haben, haben auch Bücher dieser
6. When was Einstein born? 1911 1879 1954. 2. Where was Einstein born? Ulm, Germany Jerusalem, Israel New York, USA ... Albert Einstein was married:

Looking closer at the retrieved snippets, we observe that snippets one to four provide four different pieces of text that represent different paraphrases of the same underlying idea. The correct answer can be found in each snippet, but it is written in different forms ("*14 March 1879*","*1879*","*March 14, 1879*" and "*14 Mar 1879*"). The fourth snippet also provides an orthographical variation of the right answer ("*14 March 1879*" $\Leftrightarrow$ "*14 Mar 1879*"), where "*March*" is shortened to "*Mar*". In addition, the first and third snippets are morphological variations of the same concept ("*the nobel prize of physics Albert Einstein*" $\Leftrightarrow$ "*physics nobel prize Albert Einstein*"). Furthermore, snippets three and four are semantic variations ("*physics nobel prize Albert Einstein*" $\Leftrightarrow$ "*German-American physicist*"). The last two snippets show breaks inserted deliberately by the search engine (marked as "..."). Additionally, they also reveal two other main drawbacks to snippets: they are written in different languages and they can provide wrong answers ("*When was Einstein born? 1911 1879 1954.*").

Broadly speaking, the difficulty of discovering the answer to a question has to do with successfully uncovering the missing part of the relation established by the query. The complexity of this uncovering task is due not only to the formulation of the question, the underlying linguistic phenomena on the corpus also contributes substantially to the difficulty of the task. In natural language documents, we do not only find uncountable variations that express the same entities, many words can also signal at the same relation (i.e. *"invented"*,*"discovered"*,*"was created"*). In deed, the linguistic phenomena presented in natural language corpus is much more complex than considering only possible variations of entities and relations. It inevitably involves reference resolution, complex syntactical and semantical relations, inference, world knowledge, etc.

For the practical purpose of attempting to uncover the right answer, traditional QAS start by analysing the query, in order to select an adequate strategy for satisfactorily answering the question (Moldovan et al. (2004); Chalendar et al. (2003); Chen et al. (2004); Dumais et al. (2001); Neumann and Sacaleanu (2006)). This initial phase is called *Query Analysis* and it usually aims for determining the *Expected Answer Type* (EAT). At this primary step, the answer is assigned to one of a set of distinct and separate categories (i.e. *location*, *name*, *number*), and this categorisation constrains and guides the whole answering process. The number of categories vary from approach to approach. Some strategies use a wide range of narrow categories (Echihabi et al. (2004)), in contrast to other approaches, where the number is restricted to a few, but broad and general categories (Moldovan et al. (2004)). As well as the EAT, *Query Analysis* provides the semantic content and syntactical relations with the answer.

Many answer extraction modules try to disclose these relations by taking advantage of the redundancy provided by different information sources. This redundancy significantly increases the probability of finding a re-writing of the query, called *paraphrase*, in which the answer can easily be identified. Similarly, a huge set of paraphrases considerably decreases the need for deep linguistic processing like anaphora resolution, uncovering complex syntactical or semantical relations, synonym resolution (Dumais et al. (2002)). In some cases, it reduces the extraction to the use of regular expressions (Echihabi et al. (2004)). Normally, QAS extract paraphrases at the sentence level (Echihabi et al. (2004)). The rules for identifying paraphrases can be written manually or learnt automatically (Dumais et al. (2002); Echihabi et al. (2004)), and they can consist of pre-parsed trees (Echihabi et al. (2004)), or simple string based manipulations (Dumais et al. (2002)). Paraphrases are learnt by retrieving sentences that contain previously known question-answer pairs. For example in (Echihabi et al. (2004)), anchor terms (like "Lennon 1980") are directly sent to the Web, in order to retrieve sentences that contain query and answer terms. Patterns are taken from this set of sentences, and their likelihood is computed in proportion to their redundancy on the Web afterwards. In both cases, the new set of retrieved sentences is matched with known paraphrases in order to extract new answers. This context matching method is the major advantage of this strategy, but it is also its main drawback. If the context of the new answer does not properly match the previously annotated context, the correct answer will not be identified, even though it can be readily distinguished by means of some linguistic pattern.

Hence, linguistic processing is still the central core of the best current QAS (Moldovan et al. (2004)). (Rinaldi et al. (2003)) present a domain-specific QAS which aims for finding answers in a set of technical documents by means of paraphrases. In this strategy, paraphrases are not only string based manipulations or word reordering matching, they are also considered as different syntactical variations and mapped to the same logical representation. From this representation, called *Minimal Logical Form* (Mollá et al. (2000)), they extracted answers by means of a logical proof. As a result, they observed that domain-specific QAS must deal effectively with unknown specific lexicon, abbreviations and acronyms, and for this reason, linguistic processing is still a vital issue, though it requires a large amount of processing time and its implementation takes a sustained and hard effort.

Due to the huge amount of paraphrases on the Web and its growing multilinguality, statistical methods are also used for extracting answers. In (Echihabi and Marcu (2003)), a statistical strategy is presented which scores a given sentence and a substring of the sentence, that is expected to be the answer, according to the query. The scoring strategy takes advantage of a distance metric between the sentence and the query based on the noisy channel. As a result of testing this strategy, any empirical relation between the type of the question and the performance of the system could be identified. Moreover, this kind of strategy obtains many inexact answers. This is a major problem for statistical-based approaches, because they frequently get inexact answers. The obtained answers usually consist of substrings of the answer, the answer surrounded by some context words, or strings highly closed to answers. Although, data-driven answer extractors get many inexact answers, they are normally preferred to other kinds of extractors, because they are easy to re-train, are intend to be independent of the language and demand less computational resources. In contrast to answer extractors based largely on linguistic processing, which extract exact answers, they demand more computational resources and are dependent upon the language. All things considered, QAS are greatly encouraged to take advantage of both approaches, while they are trying to cope with new questions prompted by users. Hence, one of the open research questions is: When is it appropriate to use deep processing, statistical based approaches and strategies based on distributional patterns (like frequency counts or n-grams)?.

The answer to this question has to do with the trade-off between the implementation of rule-based and easy re-trainable data-driven systems. Therefore, the burning issue of successfully combining different kinds of strategies, in order to re-rank answers, has taken off. In QA jargon, this re-ranking step is widely know as *answer validation*. In (Echihabi et al. (2004)), a strategy for combining the output of different kinds of answer extractors is introduced. This re-ranker is based on a *Maximum Entropy Linear Classifier*, which was trained on a set of 48 different types of features such as ranking in the answer extraction modules, redundancy and negative feedback. Results show that a good strategy for combing answer extractors can considerably improve the overall performance of QAS (see also Moldovan et al. (2004)). The major drawbacks to this sort of method are: (a) every time the system attempts to answer a particular question, the reranker and many answer extractors must be fully executed, and (b) it necessarily needs the special design of a re-ranking strategy.

We show how the intrinsic nature of *Genetic Algorithms* (GA) can contribute to satisfactory answer this open question and how GA can certainly help to considerably decrease the dependency upon patterns seen on training data, analogously to significantly increase the probability of matching new sentences with paraphrases taken from previously annotated question-answer pairs. We begin with `GAQA`, a core genetic algorithm that directly learns syntactical distributional patterns from contextual pairs {sentence, answer} extracted from previously answered questions. For this sole purpose, `GAQA` uses contextual tuples that are wholly consistent with the EAT of the new prompted question, and these syntactical patterns are properly aligned with new sentences in order to discover answers to this new query afterwards. `GAQA` states how likely an *n-gram* represents an answer by means of a fitness function K which aligns these distributional patterns in a word by word fashion. This pattern alignment is similar in nature to (Echihabi et al. (2004)), but `GAQA` does not take into account any pre-defined ranked pattern and considers pairs corresponding to wrong answers, that is, it attempts to be robust and avoids any manual annotation, which is always a extremely demanding task. `GAQA` is for this reason strongly data-driven and it only makes allowances for a language dependent stop-list and for the Web as a target corpus. GAQA is therefore highly independent upon the language and easily retrainable. Secondly, because of the absolute dependence on annotated contextual patterns that GAQA still suffers, we propose the next two strategies based largely on GA for significantly enhancing GAQA:

GASCA The full implementation of a complete set of rules for all possible paraphrases of each particular kind of query is without a doubt undesirable, due to the high variability of texts written in natural language. On the other hand, because of this great variability, learning all possible paraphrases inevitably involves processing and annotating a large-scale corpus. Without a shadow of doubt, this is also an undesirable scenario while we are considering QAS that deal efficiently with several kinds of questions. Consequently, taking advantage of available patterns in a more flexible way is encouraging, that is, taking into account partial matches between patterns on training tuples and new sentences. In GAQA, this matching is grounded on some detectable syntactical distributional patterns of behaviour across words and/or blocks of words respecting the EAT, which are recognisable by their relative position (Belkin and Goldsmith (2002); Schuetze (1997)). These recognisable syntactical patterns help then to distinguish strings that behave like the EAT and therefore promising answers. Indeed, we have to account for two inescapable facts, while GAQA is performing this alignment: (a) blocks of words within new sentences can be fully aligned with some patterns on the training data or (b) this alignment can be slightly distorted by some meaningless words within new sentences leading GAQA to completely miss appropriate answers. Empirically testing many possible distorted alignments is therefore a basic and essential aspect of significantly lessening the total dependence upon the training set. However, as long as we account for more seriously distorted alignments, their number increases exponentially. Here, *Genetic Algorithms* come into play, if we interpret blocks of words as blocks of adjacent ones and zeros, which signal whether a block of words should be properly aligned or not, and recall the *Schemata Theorem* which claims that these compact blocks are most likely to be propagated into next generations with a probability proportional to the fitness of individuals that carry them (Holland (2005)), that is, proportional to the contribution of the block to the alignment with the EAT, then *Genetic Algorithms* are a suitable search heuristic for readily finding out an alignment of blocks of words that best fits training patterns, because these compact blocks will pass on over generations causing a quick convergence. Naturally, this syntactical dependence between words and the EAT can be clearly seen as the interaction between different genes in a chromosome. In GA jargon, this interaction is known as *epistasis* (Holland (2005); Coello (2004)) and the intrinsic behaviour of these syntactical distributional patterns helps GA to construct building blocks that leads them to quickly move towards an optimal or highly close to the optimal alignment. The degree of *epistasis* is predominantly given by the relation amongst words within aligned blocks and blocks of words that are directly related to the EAT, *deceptive* problems are for this reason avoided (Coello (2004); Goldberg (1989)).

PreGA The extraction process considered in GAQA and GASCA are based mainly on syntactical patterns directly learnt from annotated data. The major drawback to these strategies is that if there is not enough syntactical evidence to properly align contextual patterns, the answer will not be unambiguously identified, even though it can be readily distinguished by means of some linguistic pattern. Consequently, many strings will give the misleading and spurious impression of behaving as if they were the EAT, but with a low likelihood. In some cases, right answers can be easily distinguished by some close semantic relation to the query instead of some syntactical distributional patterns.

However, discriminating beforehand the appropriate method for finding out the correct answer to a particular question is also a complex task and successfully combining several outputs of different strategies involves the especial design of a purpose-built reranking strategy. As long as generations go by, the population of `GAQA` share the same value on the genes and these values on the genes, which are likely to survive after several generations, can be interpreted as promising contextual patterns. Given the fact that GA subsequently discover these promising patterns, while they are searching for an answer, it can be concluded that they intrinsically offer a mechanism for strictly and effectively controlling the application of linguistic processing during this search, thereby ensuring a framework in which a proper balance between data-driven and linguistic processing can simultaneously guide the search. This is an important factor, because of the processing time that linguistic processing or the complete execution of several answer extractors usually requires. As well as that, this balance provides a way of improving the exactness of answers as a result of combining extraction strategies of different natures. Accordingly, in `PreGA`, this linguistic processing is based solely on computing a semantic representation of promising sentences, which are detected by means of the value of a special gene that implicitly ranks sentences.

For the purpose of the assessment of our methods, experiments were carried out considering the three most common sorts of entity and a set of six different relations. Also, our experiments took into account the CLEF[3] corpus, which considers rigorously questions on several kinds of entity and relation. Results suggest that the presented methods can cope with specific questions, specially with those questions whose answers are inserted into contexts, for which a large amount of morpho-syntactical variations does not exist . Results show also that our methods are robust to noisy training data and substantially reduce the need of manual annotations as well as lessen the dependence upon external lexical resources such as lists of locations or names. Moreover, `PreGA` is capable of getting more exact answers.

This work is organised as follows: the next section introduces the related work, section 3 deals at greater length with the design of the fitness functions for our strategies, this means thoroughly discussing the syntactical word by word alignment and its combinatorial explosion, section 4 presents our core `GAQA` and the two corresponding enhancements, section 5 describes our experiments and results in details, section 6 focuses attention on diverse future work directions, and section 7, draws some conclusions.

2 Related work

Some few approaches have applied *Genetic Algorithms* to *Natural Language Processing* (NLP) tasks: Machine Translation (Otto and Riff (2004)), and the inference of grammars from finite language samples (Keller and Lutz (1997); Aycinena et al. (2003)). As far as we are concerned, two other applications of GA to tasks closely related to QAS exist in the literature (Tiedemann (2005a,b); Day et al. (2006)).

The performance of Web QAS is wholly dependent upon the rank and amount of relevant documents fetched by the search engine. Simply stated, if the right answer is not in the retrieved passages, Web QAS will not be able to answer the question, or what is worse, they can return a wrong answer. Queries are for this reason expanded with, normally weighed, features in order to command or bias search engines in favour of documents with some promising characteristics. Unfortunately, many expansion

[3]http://www.clef-campaign.org/

features exist, especially linguistic features, and what is more, the selection of the right features relies strongly upon the type of query and the target document collection. In order to maximise retrieval performance, GA are used for learning the optimal set of linguistic features for each type of question by carrying out a systematic search and optimising the parameters for a previously annotated set of pairs {question, answer} (Tiedemann (2005a,b)). This sort of optimisation improves the retrieval performance about 15%.

Incidentally, some spans of text within the query are particularly more useful for its classification, and thus for determining an answering strategy. For example, in the following question *"What is the biggest city in the United States?"*, the most important cue is *"city"*, and indicates that the answer starts with a capital letter. This sort of cues are called *query informers* and are distinguished by segmenting and labelling query terms. The accuracy of the strategies, like Conditional Random Fields (CRF) models, for differentiating query informers depends substantially upon the selection of training features. GA have been accordingly used for choosing the subset of training features that optimises the accuracy of models such as CRF (Day et al. (2006)).

3 Syntactical word by word alignment

This section deals at length with the alignment between sentences and syntactical patterns. Firstly, we focus special attention on the acquisition process of the syntactical distributional patterns of the EAT. Secondly, we go over the word by word alignment, especially its complexity. Consequently, the goal functions of GAQA and GASCA are introduced. In order to clearly present our work, PreGA is fully described in a later section.

3.1 Acquiring syntactical distributional patterns of behaviour across words respecting the EAT

Let us assume that all pairs {answer, sentence} are stored up in a QA-STORE, which is indexed by the EAT. These pairs are obtained from previous questions answered by an external system and GAQA accesses the QA-STORE by means of the EAT corresponding to the new question, which is initially computed by an external QUERY-ANALYZER. Once tuples are retrieved from the QA-STORE, they are filtered and the answer is immediately replaced with w_0 in every retrieved pair afterwards. w_0 is an *n-gram* not in the dictionary W^t of all words in the selected tuples, $\mid W^t \mid$ is accordingly the number of different terms in W^t. This preliminary filtering consists simply in permanently removing every tuple in which its sentence contain only terms belonging to the *stop-list* and/or all its terms does not occur or occurs only once within the snippets (retrieved for this new question).

To begin with the detailed description of this acquisition process, consider S^t to be a set consisting solely of the σ^t different sentences taken from the QA-STORE. $S^t_{s^t}$ is the s^t-th sentence in σ^t, $1 \leq s^t \leq \sigma^t$. For the simplicity sake, it is reasonably assumed that the answer occurs only once in its respective sentence, and X is a binary variable which:

$$X_{s^t ik} = \begin{cases} 1 & \text{if the word } w_i \text{ is in the sentence } S^t_{s^t} \text{ at position } k \\ 0 & \text{otherwise.} \end{cases}$$

Where $1 \leq i \leq\mid W^t \mid$ and $len(S^t_{s^t}) \geq k \geq 1$. Consider also $\tau(S^t_{s^t})$ as the position of w_0 in the sentence $S^t_{s^t}$. Table 1 shows the value of $\tau(S^t_{s^t})$ for working QA-STORE tuples.

In two sentences, the answer occurs at the beginning of the sentence, in the other two sentences, the answer is at the end:

Who invented the Radio?		
w_0	$S^t_{s^t}$	$\tau(S^t_{s^t})$
Nikola Tesla	Nikola Tesla invented the radio in...	1
	The radio was invented by Nikola Tesla	6
Guillermo Marconni	Guillermo Marconni invented the radio in...	1
	The radio was invented by Guillermo Marconni	6

Table 1: $\tau(S^t_{s^t})$ for the illustrative QA-STORE tuples.

Next, it is calculated the frequency $freq_l$ in which a word w_i is to the left of w_0 with ϵ words between them:

$$freq_l(w_i,\epsilon) = \sum_{s^t=1}^{\sigma^t} X_{s^t i(\tau(S^t_{s^t})-\epsilon-1)} \tag{2}$$

$freq_l$ models the strength or the degree of the relation or correlation between the EAT and a word w_i to the left in a sentence $S^t_{s^t}$. In a similar way, it is computed the frequency of w_i to the right of w_0 with ϵ words between them:

$$freq_r(w_i,\epsilon) = \sum_{s^t=1}^{\sigma^t} X_{si(\tau(S^t_{s^t})+\epsilon+1)} \tag{3}$$

Lastly, frequencies are normalised in order to eliminate the bias in favour of highly frequent words. Eventually, the corresponding probabilities P_r and P_l are defined as:

$$P_l(w_i,\epsilon) = \frac{freq_l(w_i,\epsilon)}{freq(w_i)} \quad P_r(w_i,\epsilon) = \frac{freq_r(w_i,\epsilon)}{freq(w_i)} \tag{4}$$

In the illustrative example, the values for P_l and P_r are respectively:

P_l	ϵ				
w_i	0	1	2	3	4
invented	0	0.5	0	0	0
the	0	0	0	0	0.5
radio	0	0	0	0.5	0
in	0	0	0	0	0
was	0	0	1	0	0
by	1	0	0	0	0

P_r	ϵ				
w_i	0	1	2	3	4
invented	0.5	0	0	0	0
the	0	0.5	0	0	0
radio	0	0	0.5	0	0
in	0	0	0	1	0
was	0	0	0	0	0
by	0	0	0	0	0

Table 2: P_l and P_r for the instructive QA-STORE tuples.

Where the frequency $freq(w_i)$ is four for *"invented"*, *"the"* and *"radio"*, whereas the value for *"in"*, *"was"* and *"by"* is two. The obtained model is properly aligned by GAQA with new sentences in order to discriminate answers to new prompted questions.

3.2 Word by Word Alignment

In this section, we describe the alignment methods performed by GAQA and GASCA in detail. To start with a comprehensive description of our alignment strategies, let us

$$K(B^*,Q) = \sum_{\forall S_s \in S : B^* \in S_s}$$
$$\left(\sum_{k=1}^{\tau(S_s^*)-1} \alpha(w_{sk},Q) P_l(w_{sk}, \tau(S_s^*) - k - 1) + \sum_{k=\tau(S_s^*)+1}^{len(S_s^*)} \alpha(w_{sk},Q) P_r(w_{sk}, k - \tau(S_s^*) - 1) \right) \tag{5}$$

recall the fact that S is a set consisting of the σ different sentences taken from a set φ of N snippets. S_s is then the s-th sentence in S, $1 \leq s \leq \sigma$, and S_s^* is its corresponding sentence where B^* is already replaced with the special string w_0. In addition, $len(S_s^*)$ is the number of words in S_s^*, W is the dictionary of all words in the snippets, and $\mid W \mid$ is the number of different terms in W. With the express purpose of neatly illustrating our alignments, consider that the next new question Q is sent to the search engine: *"Who invented the helicopter?"*. For simplicity, let us account for only one retrieved sentence: S_1=*"The helicopter was invented by Igor Sikorsky"*. Let us also assume that the *bi-gram* $B^* = B_{167}$=*"Igor Sikorsky"* is been tested. Consequently, S_1^*=*"The helicopter was invented by w_0"* and $len(S_1^*) = 6$.

3.2.1 Simple word by word Alignment

When B^* is evaluated according to Q, we obtain two sentences from the QA-STORE that provide with alignment (see table 1):

The	radio	was	invented	by	Nikola Tesla
The	radio	was	invented	by	Guillermo Marconni
The	helicopter	was	invented	by	Igor Sikorsky

Table 3: Sample of alignment.

The likelihood or fitness K of a *n-gram* B^* as a correct answer is given by the formula 5. In this equation, w_{sk} is the word at position k within the sentence S_s^* and $\alpha(w_{sk}, Q)$ is a special weight for every w_{sk} which is also in the query Q. K assigns a higher fitness to *n-grams* with a similar syntactical behaviour to answers in the QA-STORE. This degree of similarity is directly proportional to the contribution of contextual words to the alignment between B^* and answers in the QA-STORE. As was fully discussed, this alignment is grounded on the relative position in which these contextual words occur and their contribution is specially weighted in favour of query terms, this way the goal function is smoothed in order to choose strings near aligned terms within the submitted question, which are plausibly closer to the right answer. The fitness function K also accounts for every occurrence of B^* in S, and therefore it makes allowance for the contribution of all its contextual alignments. Consequently, K naturally prefers relatively frequent strings, which behave like the EAT, and as a result, it avoids some incorrect answers, which are somewhat unlikely to frequently occur within localised contextual snippets. In our working example, the probability of *"the"* and *"invented"* is one, the probability of *"was"* and *"by"* is 0.5 (see table 2), then $K(B^*, Q) = 2 * 0.5 + 1 * 1 + 2 * 0.5 + 1 * 1 = 4$.

In this concrete example, $\alpha(w_{sk}, Q)$ is equal to two for query terms, otherwise one.

It is worth also to observe that this fitness is calculated every time `GAQA` directly tests an *n-gram*, thus the number of times that K must be computed in order to find out the best answer candidate is given by the number of possible *n-grams* in the retrieved snippets. If we retrieve an average of $\bar{\sigma}$ sentences per snippet, and we assume $\bar{\Upsilon}$ as the average number of words on a sentence, the **Number of Possible Answers** (NPA) is given by:

$$NPA = \frac{N * \bar{\sigma} * \bar{\Upsilon} * (\bar{\Upsilon} - 1)}{2} \tag{6}$$

The factor $\frac{\bar{\Upsilon}(\bar{\Upsilon}-1)}{2}$ represents the number of possible *n-grams* of different length. For simple values like $N = 30$, $\bar{\sigma}$=2, $\bar{\Upsilon}$=11, there are 3300 possible answers, while $N =$ 60 are 6600, $N = 120$ are 13200, and $N = 240$ are 26400. NPA provides however only a near approximation, because it does not account for highly frequent *n-grams*, especially *uni-grams* like *stop words* and query terms. At this point, it is good to highlight: (a) the recommended number of snippets to extract answers ranges between 50 and 200 (Dumais et al. (2002)), (b) all *n-grams* are potential answers for a strategy which does not make allowances for any lexicon or external knowledge that helps it to distinguish signs from words or different spellings of the same word, (c) wrong answers can be found in this set of answer candidates as well as different variations of right answers such as synonyms, (d) correct answer strings will not be necessary in a context closely related to the query or their contexts could not be correctly aligned, (e) spurious and misleading answers can have a greater likelihood than right answers, for instance some EAT combinations which share a similar syntactical behaviour equally (i. e. DATES and LOCATIONS), and (f) intentional inserted breaks in snippets make any contextual matching more difficult.

3.2.2 Full word by word Alignment

The alignment strategy presented in equation 5 still suffers from an absolute dependence upon learnt syntactical patterns. The following retrieved sentence S_1 serves to amply illustrate this great dependence: "*The helicopter was really invented in 1939 by Igor Sikorsky in Kyiv*". Thus, the *word-by-word* alignment respecting the answer with tuples in the `QA-STORE` looks like as follows (see table 1):

	The The	radio radio	was was	invented invented	by by	Nikola Tesla Guillermo Marconni		
...	really	invented	in	1939	by	Igor Sikorsky	in	Kyiv

Table 4: Sample of alignment.

The only match is due to the preposition "*by*". Consequently, the syntactic fitness of the answer candidate B^*="*Igor Sikorsky*" according to equation 5 is $K(\text{“}Igor\,Sikorsky\text{”}, Q) = 1 * 1 = 1$. Looking at table 4, it is clear that words like "*really*" or prepositional phrases such as "*in 1939*" do not contribute significantly to enhance the fitness of B^* and seriously distort the alignment. In general, two phenomena can have noticeable impact on the alignment:

1. **Proliferation** Some words are more likely than others to occur with its modifiers. Two clear examples are *noun-adjective* and *verb-adverb* combinations. As a natural consequence, it is not certain if they are likely to occur along with their modifiers

in the training data or not. In the instructive example, *"really"* does not occur along with *"invented"* in training tuples.

2. **Distortion** Some constituents such as *Propositional Phrases* (PP) are more likely than others to occur in different orders within sentences or by being arbitrarily inserted. Some good examples are: *"in 1939 by Igor Sikorsky"*, *"by Igor Sikorsky in 1939"*, *"a man called"*, *"a man named"*, *etc*.

It is certainly worth to remark pointedly: (a) taking into account *Proliferation* and *Distortion* partly alleviate the effects of intentional breaks on the alignment, and (b) removing words not necessary leads to increase the syntactic fitness, because the alignment depends crucially upon distributional patterns presented on training tuples. Thus, many combinations must be directly tested in order to determinate exactly the best syntactic fitness for an answer candidate. As a result of mitigating the effects of *proliferation* and *distortion* in our instructive example, one possible optimal alignment is shown in table 3 and the syntactic fitness of B^* is exactly calculated as follows (equation 5):

$$K(\text{“}Igor\ Sikorsky\text{”}, Q) = 2 * 0.5 + 1 * 1 + 2 * 0.5 + 1 * 1 = 4$$

To start with a formal description of this alignment strategy, let us think on the following sentence S_1^* =*"The Helicopter was really invented in 1939 by w_0 in Kyiv"* and $len(S_1^*) = 11$. Let us also assume that $P_l(w_i, \epsilon)$ and $P_r(w_i, \epsilon)$ are functions as defined in the previous section. Consider now an aligned sentence $S_s^{'}$ in which words from S_s^* are temporarily removed in such a way that the remaining words maximise the likelihood of B^* to the EAT. In particular, an aligned sentence $S_1^{'}$ is *"The Helicopter was * invented * * by w_0 in Kyiv"*[4]. For the purpose of keeping track of words that remain on $S_s^{'}$, consider the next binary variable:

$$Y_k(S_s^{'}) = \begin{cases} 1 & \text{if the word at position } k \text{ is in the aligned sentence } S_s^{'} \\ 0 & \text{otherwise.} \end{cases}$$

Where $1 \leq k \leq len(S_s^{'})$. In our instructive sentence $S_1^{'}$, $Y_4 = Y_6 = Y_7 = Y_9 = 0$ and $Y_1 = Y_2 = Y_3 = Y_5 = Y_8 = Y_{10} = Y_{11} = 1$. The **number of remaining words** (NRW) between the word at position k and w_0 within $S_s^{'}$ is defined as follows:

$$NRW(k, S_s^{'}) = \begin{cases} \sum_{j=k+1}^{\tau(S_s^{'})-1} Y_j(S_s^{'}) & \text{if } k < \tau(S_s^{'}). \\ 0 & \text{if } k = \tau(S_s^{'}). \\ \sum_{j=\tau(S_s^{'})+1}^{k-1} Y_j(S_s^{'}) & \text{if } k > \tau(S_s^{'}). \end{cases}$$

In our working example, $NRW(11, S_1^{'}) = NRW(5, S_1^{'}) = 1$. Since the goal is to discover an alignment that maximises the syntactic fitness of the answer candidate with respect to its context, the new function A takes into account the set of values for Y_k as follows:

$$A(S_s^{'}, Q) = \sum_{k=1}^{\tau(S_s^{'})-1} \alpha(w_{sk}, Q) Y_k(S_s^{'}) P_l(w_i, NRW(k, S_s^{'}) + \delta_l) +$$

$$\sum_{k=\tau(S_s^{'})+1}^{len(S_s^{'})} \alpha(w_{sk}, Q) Y_k(S_s^{'}) P_r(w_i, NRW(k, S_s^{'}) + \delta_r) \qquad (7)$$

[4]The removed words are signalled by means of a star.

In this equation, δ_l and δ_r are the left and right offset respectively. An offset attempts to tackle the displacement of contextual terms head-on. In natural language texts, some combinations of constituents or words are more likely than others to be swapped or inserted within sentences or training tuples. The most critical displacement is due to terms or constituents next to w_0 on the training set, because they can grossly distort the alignment by displacing a whole block of aligned words while sentences in retrieved snippets are being aligned with tuples in the QA-STORE. This kind of inevitable distortion is therefore doubly important and two ways of dealing with it exist: (a) increasing the number of training tuples by means of a set of manual or automatical rules that swap and/or insert words according to each particular kind of question or linguistic role, and (b) testing different aligned sentences $S^{'}_{s}$. The first option necessarily involves the design of a set of rules and a wealth of data processing while our methods are building the acquisition model, and the former, testing an enormous number of ungrammatical sentences. In the following aligned sentence, offsets are marked with a "+": *"The Helicopter was * invented * * by + + +* w_0 *+ + in 1909"*, the values of the respective offsets are $\delta_l = 3$ and $\delta_r = 2$. In the example, "+ + +" could match a string like *"a man named"* on the QA-STORE. Certainly, when $\delta_l = \delta_r = 0$ and $Y_k(S^{'}_{s}) = 1\ \forall k$, A returns the same value as K. It is good to observe that A does not change the answer candidate B^*, because it aims specifically for maximising the similarities between the context of B^* and the model acquired from tuples extracted from the QA-STORE by properly testing temporal values of $Y_k(S^{'}_{s})$. In order to be consistent with K, A also particularly favours *n-grams* near query terms by means of the same weight $\alpha(w_{sk}, Q)$. At this point, we can explicitly define a new function K^*, which make allowances for the best contextual alignment of all occurrences of an answer candidate B^*:

$$K^*(B^*, Q) = \sum_{\forall S_s \in S : B^* \in S_s} max_{\forall S^{'}_{s} \rightarrow S^{*}_{s}}\, A(S^{'}_{s}, Q) \qquad (8)$$

It is clear that K^* does not take into account all possible alignments, but the number of contextual alignments exponentially increases as long as: (a) the number of words also increases, and (b) we take into account more complex linguistic phenomena. To begin with our discussion about the complexity of this syntactical alignment, let us consider $len_l(S^*_s) = \tau(S^*_s) - 1$ as the number of words to the left of the answer candidate, and $len_r(S^*_s) = len(S^*_s) - \tau(S^*_s)$ is the number of words to the right. All possible combinations of $len_l(S^*_s)$ words are given by $len_l(S^*_s)!$. Here, we consider each word different from each other. It is a good approximation, because sentences are split into small pieces of text in which each word rarely occurs more than once. Similarly, the number of combinations to the right of w_0 is $len_r(S^*_s)!$. Incidentally, every combination of words to the left can occur simultaneously along with any combination of words to the right. The total number of **Contextual Alignments** (CA) of a sentence S^*_s is then given by:

$$CA(S^*_s) = len_l(S^*_s)! * len_r(S^*_s)!$$

In addition, consider that words can be arbitrarily removed from both contexts. Combinations regarding different context lengths must then be taken into account ($len_l(S^*_s)$,$len_l(S^*_s) - 1$,. . .,0). Therefore, the number of contextual alignments is defined as follows:

$$CA(S^*_s) = \sum_{j=0}^{len_l(S^*_s)} j! * \sum_{j=0}^{len_r(S^*_s)} j! \Longrightarrow CA^*_1 = \sum_{j=0}^{8} j! * \sum_{j=0}^{2} j! = 46234 * 4 = 184936$$

For our working sentence S_1^* ($\tau(S_1^*) = 9$), the number of possible word alignments CA_1^* is 184936. If we aim to be consistent with the syntactical nature of our methods, we must account for word orderings which are deliberately restricted to combinations that preserve their relative position. The number of contextual combinations is then:

$$CA(S_s^*) = \sum_{j=0}^{len_l(S_s^*)} \binom{len_l(S_s^*)}{j} * \sum_{j=0}^{len_r(S_s^*)} \binom{len_r(S_s^*)}{j} \Longrightarrow CA_1^* = \sum_{j=0}^{8} \binom{8}{j} * \sum_{j=0}^{2} \binom{2}{j}$$

$$CA_1^* = 256 * 4 = 1024$$

In the example, the number of possible alignments without fully considering the effects of the offsets is 1024. Naturally, this imposed restriction decreases drastically the number of contextual alignments, but all effects of offsets are not considered yet. The current formula takes into account only when values for offsets are zero. At the same time that the value for any of the offsets is greater than zero, the corresponding word next to w_0 must be aligned. Similarly, the number of new possible combinations due to the left and right context respectively are:

$$\Delta_l * \sum_{j=0}^{len_l(S_s^*)-1} \binom{len_l(S_s^*)-1}{j} \qquad \Delta_r * \sum_{j=0}^{len_r(S_s^*)-1} \binom{len_r(S_s^*)-1}{j}$$

Conversely, this effect increases considerably CA, because offsets add a number of alignments proportional to Δ_l and Δ_r, which are upper bounds for their respective offsets. The number of contextual alignments is eventually defined as follows:

$$CA(S_s^*) = \left(\sum_{j=0}^{len_l(S_s^*)} \binom{L_l}{j} + \Delta_l * \sum_{j=0}^{len_l(S_s^*)-1} \binom{len_l(S_s^*)-1}{j} \right) *$$
$$\left(\sum_{j=0}^{len_r(S_s^*)} \binom{len_r(S_s^*)}{j} + \Delta_r * \sum_{j=0}^{len_r(S_s^*)-1} \binom{len_r(S_s^*)-1}{j} \right)$$

Regarding the working example, if values for $\Delta_l = 5$ and $\Delta_r = 5$ are considered, then the number of possible combinations is 12544:

$$CA_1^* = \left(\sum_{j=0}^{8} \binom{8}{j} + 5 * \sum_{j=0}^{7} \binom{7}{j} \right) * \left(\sum_{j=0}^{2} \binom{2}{j} + 5 * \sum_{j=0}^{1} \binom{1}{j} \right)$$
$$= (256 + 5 * 128) * (4 + 5 * 2) = 896 * 14 = 12544$$

In actual fact, this result accounts for the contribution to the alignment of only one tuple {sentence, answer candidate} consisting of ten contextual words. However, the overall number dramatically increases due to the following two factors: (a) for each sentence, many answer candidates are feasible, and (b) the number of sentences is larger than one. Using the result in the previous section, a rough approximation of the number of contextual alignments for a small set of thirty snippets is given by $CA^*(S) \approx 3300 * 12544 \approx 41395200$. Consequently, efficient search algorithms are necessary to early detect promising answer candidates and test their best alignments.

4 Genetic Algorithms for Extracting Answers

This section focuses on the remaining components of GAQA and GASCA, the parameter selection of our strategies and it deals at length with the full description of PreGA.

4.1 The Genetic Algorithm for syntactical alignment (GAQA)

GAQA is described in algorithm 1. The input for this algorithm is T (the number of iterations), S (the set of sentences extracted from the snippets), I (the size of the population), ϱ is the *stop-list* of the language, p_m (the probability of mutation) and p_c (the crossover probability). P_l and P_r are also inputs for GAQA and regard the syntactic model (see section 3.1).

Algorithm 1: GAQA

```
input: $T$,$S$,$P_l$,$P_r$,$I$,$p_m$,$p_c$,$\varrho$
begin
    t ← 0;BestFound ← ∅;
    AC[t] ← createInitialPopulation($I$,$S$,$\varrho$);
    EvaluatePopulation(AC[t],$P_l$,$P_r$);
    while $(t < T)$ do
        t++;
        CAC ← Crossover(AC[t-1],$p_c$);
        MAC ← Mutate(AC[t-1],$p_m$);
        EvaluatePopulation(AC[t],$P_l$,$P_r$);
        AC[t] ← selectPopulation(CAC,MAC,AC[t-1]);
    end
    return $BestFound$
end
```

In GAQA, a *chromosome* is the tuple $\{s, k_1, k_2\}$, where its *genes* represent the sentence S_s and the boundaries of the *n-gram* B^* in this sentence. In the illustrative *chomosome* in figure 1, the value 2 represents the sentence index, and the last two numbers are the position of the boundary words of *bi-gram* B_{21314}.

2	13	14

Figure 1: GAQA Chromosome.

In this codification, two sorts of detectable distributional pattern can be recognised. First, if the position of the answer in the sentence observes a fixed pattern, it will be recognised by k_1 and k_2. This position can reveal some cues about which sort of relationship between the query and the answer exists. In particular, if questions aim for an answer that is usually the subject of the sentence, the genes k_1 and k_2 will converge to values which signal the beginning of the sentence (if it is not in passive form). If the answer does not observe any fixed pattern, different values of k_1 and k_2 will be spread unevenly across individuals in the population, because K takes into account the alignment of all occurrences of an *n-gram* while it is evaluating individuals, as a result these individuals will share the same fitness value and thus seen as different equally fit "*points*" in the search space that represent the same answer, but not necessarily inserted into the same context. This sort of pattern is however distinguished hardly in some order-free languages in which some constituents can occur in several parts of the sentence or can be omitted such as subjects in Spanish. In these cases, the convergence of these genes will rely strongly upon the amount of redundancy or some linguistic processing that unveils language specific hidden relations. Second, the first gene implicitly ranks sentences according to their contextual evidence. To illustrate this, let us think on the following pair:

{*"The Helicopter was invented in 1939 by Igor Sikorsky in Kyiv"*,*"in 1939 by Igor Sikorsky"*}

when GAQA aligns this pair with tuples on the QA-STORE (see table 3), it discovers enough contextual evidence ($K = 4$) to consider *"in 1939 by Igor Sikorsky"* to be a promising *n-gram*, though it is an unacceptable answer. Here, the interesting fact is that this gene signals if relevant contextual evidence can be found in the corresponding sentence, even though the exact answer can not be easily identified. It is worth noting that *"in 1939 by Igor Sikorsky"* will be probably lower ranked than *"Igor Sikorsky"*, because of their corresponding frequencies on the retrieved snippets.

GAQA creates an initial solution by choosing a random sentence S_s and two random cutting points afterwards: the beginning of the answer candidate $k_1 \in [1, len(S_s)]$, and the end $k_2 \in [k_1, len(S_s)]$. Another pair of cutting points is chosen every time the answer candidate contains a query word or belongs to the *stop-list*. Mutation (Figure 2) consists in randomly changing a value of a gene to an individual by choosing randomly a number r between 0 and 1. If $r < 0.33$, the index of the sentence of the answer candidate is changed. If the answer candidate exceeds the limit of the new sentence, then a sequence of words of the same length at the end of the new sentence is chosen. If $0.33 \leq r \leq 0.66$, the start index k_1 of the answer candidate is changed. If another random number r_j is smaller than 0.5 and k_1 is greater than one, the next word to the left is added to the answer candidate. If $r_j > 0.5$ and $k_2 - k_1 > 0$ then the leftmost word to the left is is removed. If $r > 0.66$, the end position of the answer candidate is changed in a similar way as the initial position, taking into account that $0 \leq k_2 - k_1 < len(S_s)$. In brief, if the position of the answer strictly observes a specific pattern (i. e. subjects in some languages), mutation tries to discover another sentence by chance, in which this pattern matches an *n-gram* that is more likely to be the answer, mutation also arbitrarily test other *n-grams* within the same context.

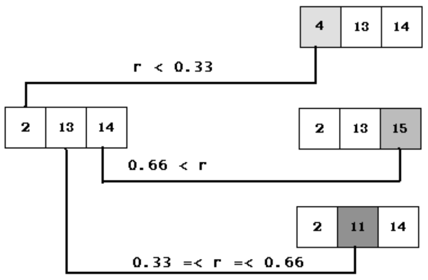

Figure 2: GAQA Mutation operator.

Crossing over two individuals $\{s_1, k_{11}, k_{12}\}$ *and* $\{s_2, k_{21}, k_{22}\}$ consists in exchanging their genes by computing the next values:

$$\alpha_1 = min\{k_{11}, k_{21}\},\ \alpha_2^* = min\{k_{12}, k_{22}\}$$
$$\alpha_3^* = max\{k_{11}, k_{21}\},\ \alpha_4 = min\{max\{k_{12}, k_{22}\}, len(S_1)\}$$

Offspring are generated by setting their *phenotype* in the following way: $\{s_1, \alpha_1, \alpha_4\}$ and $\{s_2, \alpha_3^*, \alpha_2^*\}$. These offspring aim principally for testing the context of the current parents, in which α_1 and α_3^* are their left boundaries, α_2^* and α_4

are their right boundaries. A concrete example are the parents *"Tesla in 1896"* and *"by Guglielmo Marconni"* extracted from the sentences $S_1 =$ *"The real inventor of the radio was Nikola Tesla in 1896."* and $S_2 =$ *"The radio was really invented by Guglielmo Marconni."* respectively. Their corresponding chromosome representation are {1,9,11} and {2,6,8}. Hence, the alpha values are $\alpha_1 = 6$, $\alpha_3^* = 9$, $\alpha_2^* = 8$ and $\alpha_4 = 11$, thus offspring are given by {1,6,11} and {2,9,8}. In words, the first offspring seeks to enlarge the *n-gram* corresponding to the first parent and the second to shrink in length the second parent. The size of this enlargement and reduction are integrally related to the location of *n-grams* within parents. By and large sentences with different length are normally crossed-over. For this reason, the operator must always check to see whether the right boundary of the enlargement α_4 fits the limits of the first offspring ($\alpha_4 \leq len(S_1)$). Another vital thing is abnormal individuals (i. e. {2,9,8}), an abnormal individual comes from the reduction and is due exclusively to the complete absence of overlap between the position of their parents within sentences ($k_{12} < k_{21} \vee k_{22} < k_{11}$), this absence directly derives the fulfilment of the inequality $\alpha_3^* > \alpha_2^*$. Accordingly, the operator must check to see if it is necessary to swap the values for α_3^* and α_2^* and if the new right boundary is within the limits of the sentence S_2. The new values for α_2^* and α_3^* are therefore computed as follows:

$$\alpha_2 = min\{max\{\alpha_3^*, \alpha_2^*\}, len(S_2)\},\ \alpha_3 = min\{\alpha_3^*, \alpha_2^*\}$$

In the illustrative example, $\alpha_2 = 8$ and $\alpha_3 = 8$, thus the new offspring is {2,8,8}. Hence, crossing over these two example parents generates the following two offspring: *"radio was Nikola Tesla in 1856"* and *"Marconni"*. The fitness of offspring is eventually computed according to equation 5.

Indeed, not changing the gene concerning the index of the sentence and keeping substrings from parents in offspring are a sort of inversion strategy, because it brings offspring closer to their parents and avoid the fact that they can move from one region to another, playing an explorative instead of an exploitative role. GAQA selects individuals at the end of each iteration according to their fitness value by means of a *proportional mechanism*. GAQA takes advantage of this selection strategy as a way of dealing directly with individuals which represent *stop-words* and contain query terms. Due to their high frequency and ambiguous syntactical behaviour, the probability that some contextual words provide them with alignment significantly increases, giving the misleading impression of answers, taking over offspring and passing on over generations. Since these individuals are undesirable, GAQA recombines all individuals in the population and selects the next population from these generated individuals and their parents afterwards, this way GAQA accounts for a larger population from which it can choose desirable individuals and the best still pass from one generation to the other. As a logical consequence, p_m and p_c are set to one, avoiding a fine adjustment to these parameters. The size of the population is $I = 20$ and it runs $T = 25$ iterations, this means it can theoretically tests at most 500 different individuals. It is worth to observe experimentally that about 50 different individuals are really tested during the whole search because stronger individuals gradually take over populations, hence, these values are rightly interpreted as a state in which the population finally converges.

The *implicit parallelism* of GA helps to quickly identify the most promising sentences and strings according to query keywords and the syntactic behaviour of the EAT. This approach differs fundamentally from a traditional query keyword matching ranking in: (a) GAQA do not need to test all sentences and/or strings, because GA quickly find cue patterns that guide the search. In GAQA, these patterns are indexes of the most promising sentences, or some regular distribution of the position of the answer within sentences. (b) these cue patterns weigh the effect of query terms and the likelihood of the answer candidate to the EAT. (c) the fitness of the answer candidate is calculated according to these cue patterns, causing answer candidates with more context are relatively stronger individuals and more likely to survive. (d) due to fact that stronger individuals survive, these cue patterns lead the search towards the most promising answer candidates. As a result, GAQA tests principally the most promising strings. In fact, it is especially important to only consider promising individuals, because GAQA must check if the context of each occurrence of an answer candidate matches the acquired model.

4.2 GA for Syntactic Contextual Alignment (GASCA)

In GAQA, promising answers are subsequently discovered by means of the simple word by word alignment defined in equation 5. These promising individuals gradually take over populations as long as generations go by. GASCA aims specifically for improving the context matching of these promising individuals by performing the full word by word alignment described in section 3.2.2. GAQA interacts directly with GASCA, every time it generates a new offspring, by sending an answer candidate B^* and the set of sentences S. GASCA returns consequently the new fitness according to equation 8. Algorithm 2 shows the flow of GASCA, which in line three replaces the answer candidate string with w_0 and in line eight picks individuals of the next generation according to the *proportional mechanism*.

Algorithm 2: GASCA

```
   input: $T$,$S$,$P_l$,$P_r$,$I$,$p_m$,$p_c$,$w_0$,$B^*$
1  begin
2      t←0;BestFound ← $\emptyset$;
3      S←replace($S$,$B^*$,$w_0$);
4      AC[0] ← createInitialPopulation($I$,$S$);
5      EvaluatePopulation(AC[0],$P_l$,$P_r$,$S$);
6      while $(t < T)$ do
7          t++;
8          AC[t]←selectPopulation(AC[t-1]);
9          CAC←Crossover(AC[t],$p_c$);
10         MAC←Mutate(AC[t],$p_m$);
11         evaluatePopulation(AC[t],$P_l$,$P_r$,$S$,$w_0$);
12     end
13     return $BestFound$
14 end
```

One fundamental aspect of this interaction strategy is the cached fitness. Every time GASCA calculates the fitness of an individual which has not previously sent by GAQA, it stores its new fitness in a cache. This cache brings about a sharp reduction in the processing time, because best answer candidates are sent several times from GAQA. However, GASCA does not absolutely guarantee that every time it runs, it will compute the same best alignment. But, the reason to favour this cache at the expense of quality of solutions is three-fold: (a) GASCA optimizes the context for each occurrence of the answer candidate, this means several runs can slightly differ in the fitness of some contextual alignments (equation 7), but these increases and decreases have a marginal impact, because the sum in equation 8 considerably lessens these minor differences in the final fitness, (b) the optimal alignment is not strictly necessary, but being fast and closer to the optimal alignment is enough to solve this problem, and (c) parameters were mainly manually set by inspecting their stability after several runs. These parameters are as follows: the population size is $I = 10$, the number of generations $T = 30$,$p_c = 0.8$, and $p_m = 0.1$.

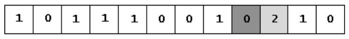

Figure 3: GASCA Chromosome.

The *chromosome* (figure 3) is based mainly on a binary representation of a sentence, where a value of one is assigned in every case that a word is considered in the alignment, otherwise zero. The gene corresponding to w_0 is replaced with two genes that are natural numbers and correspond to the offsets. The individual in figure 3 represents the aligned sentence $S_1^{'}$ =*"The Helicopter was invented by* w_0 + + *in"*[5], and its offsets are *"0"* and *"2"* for its left and right contexts respectively. GASCA creates an initial solution by computing a random number for each word in the sentence $S_s^{'}$. For every random number greater than 0.5, the value of the corresponding word is one, otherwise is 0. If any of the adjacent words of the answer is plugged to one, then a random value for the offset is computed, otherwise it is assumed to be zero. This random number is selected from 1 to $len(S_s^{'})$, that is, the length of the sentence is used as a bound for the offset. Mutation randomly changes a value of a gene to an individual by choosing two random numbers (r_1, r_2) between 0 and 1. If $r_1 \leq 0.5$, the left context of the answer candidate is changed. If $r_2 > 0.5$ and the word to the left of the answer candidate is considered in the alignment (the gene is one), then the offset is changed. The new value for the offset is a random number between 0 and the length of the sentence. Otherwise, the value of a random gene is flipped from one to zero or vice versa. If $0.5 < r_1 \leq 1$, the right context of the answer candidate is changed in a similar way. In short, mutation aims specifically for finding out a word by chance, which displaces the contextual alignment.

Crossing over two individuals consists in cutting them at two randomly selected points. One cutting point is picked from each context. In order to avoid checking the consistency of the offset in offspring: (a) Instead of tails, heads are exchanged afterwards, and (b) words next to the answer candidate are not considered as cutting points. In figure 4, the aligned sentences $S_1^{'}$ and $S_1^{''}$ =*"The First Helicopter was invented in 1939 by* w_0*"* are crossed over at the past tense verb *"invented"* and the noun *"Kyiv"*. Offspring represent the next two aligned sentences: *"The First Helicopter was invented by* w_0 + + in" and *"The Helicopter was invented in 1939 by* w_0*"*.

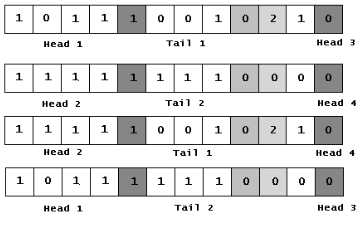

Figure 4: GASCA Cross Over.

[5]The corresponding S_1^* is *"The First Helicopter was invented in 1939 by* w_0 *in Kyiv"*.

In GASCA, one main advantage of the coding and reproduction mechanisms is that the left and right context converge independently. If one context is properly aligned, their corresponding phenotypes will pass on over generations, because individuals which carry these genes will have a better fitness and will be for this reason more likely to survive. As a consequence, this independence leads to simultaneously discover evidence on both contexts, bringing about a quicker convergence and causing a substantial reduction in the number of tested ungrammatical sentences. This simultaneous convergence is due solely to the non-existence of epitasis between both contexts. A final remark is that, unlike conventional methods, GASCA discovers the best patterns, along with the answer, that are necessary to extract answers in this context. These patterns are traditionally hand-written or learnt and manually filtered beforehand.

4.3 Predication and Data-driven syntactic alignment (PreGA)

Due to the syntactical nature of the extraction process in GAQA and GASCA, a natural way of enriching this alignment is by adding semantics. PreGA attempts to enhance the performance of GAQA by adding external semantic knowledge. This section discusses two possible choices of semantic processing and describes PreGA in details.

4.3.1 Semantic Analysis: Predication

There are two possible ways of directly enhancing this extraction process by the addition of semantic knowledge: (a) corpus driven semantics like *Latent Semantic Analysis* (LSA) (Landauer et al. (1998)), (b) some linguistic out–of–the–box tools for semantic processing. Since answer candidates are subsequently discovered by properly aligning their context with contextual syntactic patterns, which are logically deduced from previously annotated {question, sentence, answer} tuples, external semantic processing is clearly encouraged, causing the robustness and performance of the system to be vastly improved. Accordingly, the absolute dependency upon the training data is markedly decreased. Another reason is that, on the one hand, the query is seen as a relation amongst many entities, on the other hand, LSA provides of semantic relations between pair of terms. Consequently, LSA seems not to be adequate to distinguish clearly the semantic relation amongst a set of entities (Kintsch (1998)). A final reason is the existence of tools like Montylingua[6], which computes a semantic representation of a raw text in English. It specifically extracts {subject, verb, objects} tuples, which are a predicate-argument representation of sentences in a given text.

The real and strong motivation behind the use of predication is that it provides a semantic relation between the predicate and arguments. In this representation, the predicate is clearly seen as the relation that entities hold, while arguments are interpreted broadly as entities. To illustrate, consider the next sentence "*Igor Sikorsky invented the helicopter in 1939 in Kyiv.*". MontyLingua provides the following predicate-arguments representation for this sentence:

invent(Igor Sikorsky,helicopter,in 1939,in Kyiv)

In this example, it is clear that the answer matches the subject (first argument), which can be clearly differentiated by its syntactic behaviour. Taking into account the predicate-arguments representation of the query, the search of its missing part can be interpreted clearly as the search for a sentence in the text with a similar predicate-arguments representation, in which, the argument (or the predicate) corresponding to the answer syntactically behaves like the EAT, and some other arguments and/or the

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

predicate in both predicate-arguments representations match. To illustrate this, consider the following query: "*Who invented the helicopter*?". The predicate-arguments representation provided by MontyLingua is as follows:

invent(Who,airship)

Comparing both predicate-arguments representations, it becomes clear that the subject "*Who*" matches the subject "*Igor Sikorsky*". At the same time, the large-scale redundancy of the Web steeply increases the probability of finding a rewriting of the query, where the answer can be easily identified by means of this alignment strategy.

One noteworthy intrinsic feature of our MontyLingua API is a cache, which stores previous predicate analysis and works in a similar way to `GASCA`, consequently, it also brings about a marked reduction in the processing time. This is a significant factor, because predicate analysis is also computer demanding.

4.3.2 The predicate-arguments and data-driven genetic algorithm

`PreGA` uses the same parameters and a similar *chromosome* representation to `GAQA`, but completely replaces some of its components. The partial replacement consists principally of a new mutation operator and a purpose-built goal function, which take advantage of *predicate analysis* in order to enrich the syntactical data-driven alignment of `GAQA`. This new mutation operator chooses a random sentence with a uniform probability. Then, a predicate analysis is performed to the picked sentence and one of its arguments is randomly selected afterwards. Only arguments consisting entirely of numbers and letters are taken into account. The sole purpose of this operator is to systematically explore the fitness of objects/arguments belonging to sentences on the document. Highly frequent arguments are consequently more likely to be selected. The fitness function of an answer candidate or individual B^* respecting to the query Q is given by:

$$K_p(B^*, Q) = K(B^*, Q) * K_l(B^*, Q) \tag{9}$$

Where $K_l(B^*, Q)$ is the "*linguistic fitness*" of the individual and $K(B^*, Q)$ is its fitness according to the annotated context (equation 5). This product allows to calculate $K_l(B^*, Q)$, only when $K(B^*, Q) > 0$. Accordingly, only individuals with some contextual evidence are further tested. $K_l(B^*, Q)$ is defined as follows:

$$K_l(B^*, Q) = \sum_{\forall S_s \in S : B^* \in S_s} \gamma(S_s, Q) *$$

$$\left(1 + \eta(S_s, Q) \sum_{o \in obj(S_s)} J(B^*, o)\right)\left(1 + \eta(B^*, S_s) \sum_{q \in obj(Q)} \sum_{o \in obj(S_s)} J(q, o)\right) \tag{10}$$

$\gamma(S_s, Q)$ is a binary variable, where its value is one whenever the verb of the sentence S_s matches the verb of the query, is otherwise zero. $\gamma(S_s, Q)$ considers only sentences where the verb of the sentence and query match, even though their senses are not the same. At this point, synonyms are not considered on the ground that `PreGA` trusts implicitly in the massive redundancy of the Web. $\eta(S_s, Q)$ is a binary variable, where its value is one whenever the subject of the sentence S_s and the query match, otherwise is zero. $\eta(B^*, S_s)$ is similar to $\eta(S_s, Q)$, but its value is one whenever the individual matches the subject of the sentence S_s. $obj(S_s)$ is a function which returns the arguments of the sentence, excluding the subject, and $obj(Q)$ is an homologous function for the query. Each argument within the predicate of the query is compared with each argument in the predicate of the sentence according to the *Jaccard* measure (Cohen et al. (2003)). The *Jaccard* measure J is a ratio between the number of terms that occur in both strings, and the total number of different terms in both sequences. For instance, consider B_1="Sikorsky" and B_2="Igor Sikorsky", $J(B_1, B_2) = \frac{|B_1 \cap B_2|}{|B_1 \cup B_2|} = 0.5$.

It is clear that the denominator is always greater or equal to the numerator, thus the value of J is between one and zero. $\eta(S_s, Q)$ and the sum of values of J aim for: (a) clearly differentiating the sense of the verb presented on the sentence S_s and the query Q, and (b) directly measuring the semantical bonding between the query Q and a particular sentence S_s on the text. In the instructive example, only one retrieved sentence is considered: *"Igor Sikorsky invented the helicopter in 1939 in Kiev."*, thus, the value for K_l is computed as follows:

$$K_l(\text{"Igor Sikorsky"}, \text{"Who invented the helicopter?"}) = 1 * (1 + 1 * 1) * (1 + 0 * 1) = 2$$

All in all, `PreGA` takes advantage of predicate analysis in order to enrich the alignment of annotated contextual patterns with the context of new answer candidates. This predicate analysis is performed as long as `PreGA` tentatively identifies similarities in both contexts ($K(B^*, Q) > 0$ in equation 5), consequently, the application of linguistic processing is carefully balanced and the strong dependency of the alignment on the annotated data significantly decreases. Moreover, as long as the gene representing the sentence index converges, `PreGA` saves computational time by taking advantage of the inherent cache provided by our MontyLingua API. Nevertheless, `PreGA` still remains heavy dependent upon patterns seen on the training data, while it is selecting a new answer string ($K(B^*, Q) > 0$ in equation 5).

5 Experiments

For the purpose of the assessment of our methods, experiments were carried out considering a set of six different relations and the three most common sorts of entity: PERSON, LOCATION, DATE as well. Additionally, our experiments also made allowances for a set of heterogeneous relations aiming at a particular kind of entity. Accordingly, this section fully describes the evaluation metric, data-sets, external resources and presents a discussion at a greater length of the obtained results as well as a baseline.

5.1 Evaluation Metric

In order to assess the performance of QAS, the standard metric of *Mean Reciprocal Rank* (MRR) is used. MRR rewards each answer according to its position in the ranking, by assigning each the inverse of its position. The MRR (see Lita and Carbonell (2004)) of a system is the average value for a set Qu of Q_n questions:

$$MRR = \frac{1}{Q_n} \cdot \sum_{\forall q \in Qu} \frac{1}{rank\_answer_q} \tag{11}$$

5.2 Experimental Settings

Table 5 shows six out of the seven considered data-sets, which were formally split into two sets: *training* and *testing*. The former was directly sent to an external system in order to retrieve training tuples. We tried to avoid manual annotations at all costs by taking into account right and wrong pairs. The latter set was separately sent to each individual method and our `Baseline`. Table 5 also shows the templates used for retrieving testing and training snippets as well as the size of the corresponding sets.

Data-set	Template	Total	Training	Testing
Inventors	*who invented the* {*invention*}?	272	87	185
Presidents	*who is the President of* {*country*}?	120	33	87
Composers	*who composed the* {*symphony*}?	180	80	100
Prime Ministers	*who is the Prime Minister of* {*country*}?	103	29	74
Locations	*where is the* {*monument/city*}?	120	37	73
Dates	*when was* {*person*} *born*?	4000	2160	145

Table 5: Different data-sets.

The last data-set is the standard **CLEF-2004** data-set, which refers to answers from 1994/1995 newspapers articles. The 169 who-questions were directly sent to our external system. Each right answered question was taken into account as a member of the training set and every unanswered question was considered as part of the testing set. The tuples where manually checked in order to effectively remove all pairs {sentence, answer} which contains a wrong answer. These who-typed questions consider several topics and wrong answers, even tough they are few, they can have a considerable impact on the model. Indeed, this manual annotation is a demanding task.

The external system[7] implements the approach presented in (Figueroa and Neumann (2006)). This system identifies answers by modelling the strength of the syntactic relations between words. Similarly to our GA-based methods, these syntactic relations are discriminated on the ground of the frequency of the relative positions of words within sentences. As well as that, this system extracts answers from web snippets, it supplies for this reason suitable context for the alignment of our GA-based strategies. In addition, this system makes use of some specialised modules for extracting answers, that is modules that account for traditional regular expressions and a lexical database of locations, namely WordNet[8]. The MRR values obtained by this external system are 0.69, 0.74 and 0.50 for English questions aiming at Dates, Locations and Persons respectively.

The back-end to the search engine was provided by Google API[9]. On the one hand, Google usually returns at least one snippet containing an answer (considering questions in table 5), on the other hand, Google is more likely to truncate sentences than other search engines such as MSN Search[10]. Google API differs from Google Search (the front-end) in the bias in favour of advertisements, cookies, history and personal profiles as well as other preferences.

Three additional remarks are: (a) the weights $\alpha(w_{sk}, Q)$ were set as in section 3.2.1, (b) snippets were tokenized by means of spaces and standard punctuation signs: colon, semicolon, coma, dot as well, and (c) the EAT was identified by simply Wh–keyword matching. Templates in table 5 were especially designed so to avoid any query analysis error.

5.3 Baseline

If a term is likely to be an index of a collection, it is highly possible that a subset of documents will provide enough localised context to unambiguously identify its role and significance. The idea is therefore to exploit the likelihood of a term as an index of a collection of snippets in order to approximately estimate how difficult it is to sharply

[7]This system is currently available at http://experimental-quetal.dfki.de/
[8]WordNet can be freely downloaded at http://wordnet.princeton.edu/
[9]http://code.google.com/apis/ajaxsearch/
[10]http://search.msn.com/developer

distinguish if it is the answer to a question or not. Our baseline is for this reason based on the *term frequency-inverse document frequency* (*td-idf*), which is the traditional metric of a term as index in *Information Retrieval*. But, since we are dealing with web snippets, *td-idf* is changed as follows:

$$tf - idf(w_i) = \frac{freq(w_i)}{max_{\forall w_i \in W} freq(w_i)} * log\left(\frac{\mid \varphi \mid}{nd(w_i, \varphi)}\right) \tag{12}$$

Where nd returns the numbed of snippets containing the word $w_i \in W$. Instead of counting the frequency on each snippet, the *idf* is weighted proportional to the normalised frequency of w_i in the whole collection. Essentially, due to the size of snippets, they are small pieces of text and the original normalised term frequency does not draw a distinction between words in each web snippet, all of them can be considered as an index of the snippet. However, we are not interested in the set of indexes of each snippet, but rather, aim for computing the likelihood of strings as indexes of the whole collection.

Algorithm 3: `Baseline`

```
  input: $\varphi$, $\varrho$
1 begin
2     words←extractAllWords($\varphi$);
3     rank←tdidf(words);
4     rank←filterWords(rank,$\varrho$);
5     rank←filterRareStrings(rank);
6     return $rank$;
7 end
```

The input of the baseline is the snippets φ and a *stop-list*. Line 2 extracts all words from the document. Line three looks for the most frequent word and computes the *td-idf* according to equation 12. Line four filters words that are in the given *stop-list*. Line five filters strings that contains numerical and/or alphabetic characters. In this step, strings that an answer extractor will easily identify as non answers are removed:*math symbols*, *html tags*, *special characters*. Eventually, line six returns the ranked answer candidates.

5.4 Results

The next table shows an overview of the obtained results:

Strategy	MRR	Total	1	2	3	4	5	AA
`Baseline`	0.376	403	137	92	78	42	41	13
`GAQA`	0.497	415	242	78	38	31	12	14
`GAQA+GASCA`	**0.512**	437	240	97	38	35	13	14
`PreGA`	0.373	344	155	101	33	23	10	22

Table 6: Overview of the results per strategy (out of 624 questions).

All data-sets consider a total of 713 questions. In 63 cases, an answer was found, which was not provided by the corpus (AA), for instance, in case of inventions, which their inventors are unclear such as "*The Radio*", the corpus[11] provides "*Guglielmo Marconi*", but we can easily find on the Web: "*Nicola Tesla invented the Radio*". In the CLEF

[11]http://corporate.britannica.com/press/inventions.html

corpus this ambiguity is more often, due to the fact that some answers are out of date and the retrieved snippets contain updated information. In 89 cases, no answer was manually found on the best 30 retrieved snippets (NAS). In some cases, the answer was in a large span of text which was intentionally replaced with a break. In other cases, the retrieved snippets contained a localised context in which the answer did not occur, due to a marked bias in favour of some words within the query with a strong likelihood as indexes of another collections of documents where the answer hardly occurs. For instance, the query "*Who invented the supermarket?*" retrieves strings like "*The supermarket giant claims the move will bring voice over Internet protocol . . . the man who really invented the Internet*". In this particular case, on-line advertisements of supermarkets have a strong influence over the terms "*invented*" and "*supermarket*". In actual fact, only the first five ranked answer strings were considered for calculating the MRR score. Our strategies still found answers to other questions in lower-ranked positions, which do not contribute to the final score. In particular, the question "*Who invented the Lego*?", two of our methods found an answer ranked at position six. As well as that, Alternative Answers (AAs) were not taken into account in the final MRR score.

It is worth to observe experimentally that our strategies are more likely to find *uni-grams* as answers than whole answer strings. In the case of who-typed questions, surnames are usually more frequent within web snippets than names or full names. Since our strategies make allowance for the alignment of every occurrence of an answer candidate, they are also inherently biased by frequency counting. Consequently, surnames tend to be naturally preferred to names and full names. A good example is in the set of Composers[12], "*Schubert*" is more likely than "*Franz Schubert*" or "*Franz*". In general, it is well-know that statistical oriented approaches often extract these kinds of inexact answer (see also Echihabi et al. (2004)).

`PreGA` performed as the `Baseline`. On the one hand, `PreGA` discovered answers to a lower number of questions, on the other hand, `PreGA` ranks right answers higher, it achieves therefore a better distribution of the answer rank. These are two sufficient reasons for their similar MRR scores. In addition, it is clear, `GAQA` and `GAQA+GASCA` outperforms `PreGA`. Furthermore, by considering only results upon table 6, it can be concluded that the flexible alignment of `GAQA+GASCA` performs slightly better than `GAQA` and their answer rank distributions are similar. Accordingly, they also finished with a similar MRR score.

Broadly speaking, the best systems that take part into the TREC competitions score an MRR value between 0.5 and 0.8. This score is computed over a wider variety of questions which are usually harder to answer. These systems therefore necessarily incorporate knowledge resources, specialised document retrieval, answer selection and validation. Under these concrete facts, results obtained by `GAQA` and `GAQA+GASCA` (0.497 and 0.512 respectively) seem to be positively encouraging. The following table shows results regarding each data-set:

[12]http://en.wikipedia.org/wiki/List_of_Composers_by_name

Corpus	Questions	NAS	`Baseline`	`GAQA`	`GAQA+GASCA`	`PreGA`
CLEF-2004	75	24	0.309	**0.387**	0.261	0.261
Inventions	185	28	0.421	0.502	0.452	**0.546**
Presidents	89	1	0.524	0.571	**0.629**	0.222
Prime Ministers	76	5	0.473	0.706	**0.714**	0.197
Composers	100	23	0.315	0.500	0.489	**0.584**
Locations	43	1	0.568	0.638	**0.684**	0.507
Dates	145	7	0.173	0.365	**0.450**	0.266

Table 7: MRR overview.

`GAQA+GASCA` achieved the best MRR score for four out of seven data-sets, while `PreGA` finished with the best score for two data-sets. In four data-sets, the `Baseline` outperformed `PreGA`. As a reasonable conclusion, the data-driven enrichment tended to perform better than our enhancement based mainly on *predication*. Given the highly variable MRR score achieved by `PreGA`, it can be concluded that the *predication analysis* provided by MontyLingua covers wider paraphrases that usually occur in some contexts/topics than others.

The approach presented on (Lita and Carbonell (2004)) scored a MRR value of 0.54 for a set of who-typed questions from the TREC 9, 10 and 11 corpus, and the evaluation was strict with respect to answer patterns provided by TREC. In (Dumais et al. (2002)), they obtained a MRR value of 0.45 for 500 TREC-9 questions. In (Charles et al. (2001)), a score about 0.5 was obtained for different configurations of their system. Their set of questions also aimed at names of persons, and the criteria for considering the contribution of correct answers to the MRR value is similar to ours. They also considered only names as correct answers, semantically related terms were not taken into account. Their ranking strategy seems to be likely to find full names as answers , takes advantage of simple syntactical patterns and their methods aim at a fixed corpus (TREC) as a target, this means that they know a priori how answers occur on the corpus. In contrast to our methods, which distinguish more strings as answers (not only full names), but they use the Web as a target. Hence, they do not know a priori how answers occur on the retrieved snippets. This intrinsic factor has an impact on the MRR values, but not on the real performance of the system. In more practical terms, we consider as a right answer the exact string match with the answer in the corpus, a more complete name description, only the surname or the name. Right answers do not make allowance for orthographical variations in order to reduce the ambiguity of the evaluation. Hyphens were removed.

In (Figueroa and Neumann (2006)), a data-driven strategy for extracting predicted answers from web snippets was presented. This strategy was evaluated by observing the distribution of answers within the rank of predicted answers. This method was properly assessed considering the set of questions provided by CLEF-2004, where it finished with a MRR value of 0.69 for questions aiming for a DATE as an answer, while it scored 0.74 for questions aiming for a LOCATION and 0.50 for questions aiming for a PERSON. Since this strategy takes advantage of a lexical database of locations and a set of regular expressions for dates during the answer extraction step, our results seems greatly motivating. More to this comparison, the answer extraction for the EAT PERSON is based mainly on identifying sequences of capital letters on predicted answers. Thus, it is similar in nature to our strategies. The MRR value for this kind of question was 0.5, which is lower than the values obtained by `GAQA` and `GASCA` for a similar sort of question. It is also important to comment that the CLEF-2004 is more heterogeneous than the set considered here, and the values achieved by our methods for the CLEF-2004 corpus are computed from the set of questions that the external system could not answer, and this external system is based mainly on this strategy, these values are for this reason evidently not comparable. But, it is perfect clear that our strategies successfully answered questions that this external system did not.

Regarding the set of locations[13], the approach presented on (Figueroa and Atkinson (2006)) scored an MRR value about 0.8 for a similar data-set and the Web as a target corpus. This approach takes advantage of external lexical resources. `GAQA` and `GAQA+GASCA` scored 0.638 and 0.684 respectively for a subset of the same set of questions. But, our methods are independent upon a lexical database of locations. In this type of entity, correct answers were the country or the city. With regards to questions aiming at a DATE[14], correct answers were considered: the year, the month and year or the full date.

	`Baseline`	`GAQA`	`GAQA+GASCA`	`PreGA`
`Baseline`	1	0.2899	0.2050	0.1878
`GAQA`	0.2899	1	**0.640**	0.380
`GAQA+GASCA`	0.2050	**0.640**	1	0.3038
`PreGA`	0.1878	0.380	0.3038	1

Table 8: Average Correlation Coefficient between each pair of strategies.

Another key issue relating to the evaluation of Question Answering Systems is the distribution of the rank of the answer achieved by different strategies. Since MRR does not show any distinction between different distributions of the answer, the correlation coefficient between each pair of ranks was computed. The average value for each pair of methods is shown table 8. It is fairly evident that this coefficient does not make a sharp distinction, but it helps to draw broad conclusions like the similarity between the ranks of `GAQA` and `GAQA+GASCA`. The following four tables describe results achieved by each strategy for each data-set:

Corpus	Questions	NAS	MRR	1	2	3	4	5	AA
CLEF-2004	75	24	0.309	9	6	5	3	2	2
Inventions	185	28	0.421	39	24	20	8	11	10
Presidents	89	1	**0.524**	32	17	12	4	3	0
Prime Ministers	76	5	**0.473**	20	15	12	5	4	0
Composers	100	23	0.315	13	14	8	3	4	0
Locations	43	1	**0.568**	16	8	7	3	1	1
Dates	145	7	0.173	8	8	14	16	16	0

Table 9: Results obtained by the `Baseline`.

Given the fact that the proposed `Baseline` ranks *uni-grams* according to an approximation of their likelihood as index of the set of retrieved snippets and this likelihood gives a simple notion of how sharp the role of a particular word within this set can be determined, it can be concluded that answers are more likely to play a role as an index, if the question aims for a President, Prime Minister or Location. Since our

[13]http://www.glasssteelandstone.com
[14]http://www.famousbirthdays.com/bday_123.html

methods are also more efficient in coping with these sorts of question, it seems that it is easier to identify the right answer on their corresponding collections of snippets. Contrary to dates, which do not usually play a role of index. It seems therefore to be more difficult to readily distinguish their role on the text, hence, whether they are answers or not. This drawback is usually lessened by means of purpose-built regular expressions, which are normally language dependent.

Corpus	Questions	NAS	MRR	1	2	3	4	5	AA
CLEF-2004	75	24	0.387	13	2	1	1	0	3
Inventions	185	28	0.502	64	12	7	5	1	10
Presidents	89	1	**0.571**	42	12	4	3	1	0
Prime Ministers	76	5	**0.706**	42	8	8	5	1	0
Composers	100	23	0.500	30	11	4	6	1	0
Locations	43	1	**0.638**	23	4	0	3	2	1
Dates	145	7	0.365	28	29	14	8	6	0

Table 10: Results obtained by the `GAQA`.

The table above shows results for `GAQA`. Best results were obtained for the data-sets regarding Presidents, Prime Ministers[15] and Locations. `GAQA` outperformed the `Baseline` on every data-set. Given the lower MRR scores achieved by `GAQA` on the CLEF and Dates data-sets, it can be concluded that the amount of training data is not the only significant factor for identifying answers on snippets readily. This meaningful difference in the score of CLEF data-set can be seen as a sharper difference between the contexts of the training and testing sets.

Corpus	Questions	NAS	MRR	1	2	3	4	5	AA
CLEF-2004	75	24	0.261	8	7	1	3	1	2
Inventions	185	28	0.452	52	20	5	6	4	11
Presidents	89	1	**0.629**	46	10	9	3	3	0
Prime Ministers	76	5	**0.714**	41	16	2	4	0	0
Composers	100	23	0.489	28	15	1	5	3	0
Locations	43	1	**0.684**	26	1	4	0	1	1
Dates	145	7	0.450	39	28	16	14	1	0

Table 11: Results obtained by the `GAQA+GASCA`.

The table above shows results for the two-staged `GAQA+GASCA`. Apart from the CLEF data-set, all results are better than the presented `Baseline`. For three data-sets `GAQA` outperforms `GAQA+GASCA`: CLEF, Inventions and Composers. The huge difference between the data-set of dates and presidents shows that the alignment can be highly sensitive to intentional breaks on snippets and the addition of contextual words such as adverbs and adjectives.

[15]http://en.wikipedia.org/wiki/List_of_State_leaders

Corpus	Questions	NAS	MRR	1	2	3	4	5	AA
CLEF-2004	75	24	0.261	8	7	1	3	1	2
Inventions	185	28	**0.546**	69	9	3	2	2	19
Presidents	89	1	0.222	0	31	8	4	2	0
Prime Ministers	76	5	0.197	0	22	6	4	0	0
Composers	100	23	**0.584**	41	5	3	2	0	0
Locations	43	1	**0.507**	17	4	2	2	3	1
Dates	145	7	0.266	20	23	10	6	2	0

Table 12: Results obtained by the `PreGA`.

The table above shows the contribution of *predication* to the alignment. Results show that *predication* did not improve substantially the performance of the system. According to this table, results split data-sets into two different groups where the difference can be interpreted as a result of the quality of the *predication analysis* performed by MontyLingua. Due to two determining factors, MontyLingua can not readily identify the predicate and/or arguments. First, it is unclear if MontiLingua can deal efficiently with all possible paraphrases presented on snippets, especially relative clauses. Secondly, ungrammatical snippets seriously distort the *predication analysis*. On the whole, `PreGA` can not fully analyse sentences with intentional breaks, `PreGA` can not therefore easily discover right answer strings to some kinds of question on retrieved snippets.

But, it is important to duly note that our approach has its limitations. When our methods try to answer questions that aim at a LOCATION or DATE, they can not readily distinguish the learnt syntactic behaviour of the EAT in retrieved snippets, because both behave similarly in English (as in another languages), both are indeed locations one in time and the other in the space. This is a crucial aspect, when our system aims for a birthplace or birthday, which usually co-occur within the same span of text. Another thing is that exists a larger amount of variations for expressing the same date than the same location (see example in the first section). This number of variations has a vital impact on the fitness function, while the system is considering the occurrences to be aligned. Hence, we can explain the low performance on the data-set regarding dates. In this case, a date normalisation strategy is necessary.

Corpus	`Baseline`	`GAQA`	`GAQA+GASCA`	`PreGA`
CLEF-2004	23.5	1909	41995	7550
Inventions	76.12	1609	37870	30826
Presidents	29.31	5293.95	103040.8	10084.24
Prime Ministers	27.66	6468.9	123169.67	11632.9
Composers	70.14	2511.16	58943.7	32203
Locations	27.66	6468.9	121356.67	11632.90
Dates	**81.16**	**99232.17**	**813696.6**	**123647**

Table 13: Average execution time for each strategy vs. data-set (milliseconds).

Another inescapable fact is that our strategies missed some answers, because the goal function properly evaluates individuals, but if the answer is in expressed in a way which was not learnt by the system, the answer is then missed or it is highly possible missed. If the answer is in a learnt form, but its frequency is low and the contextual evidence is not strong enough to guide the search, then our methods can start looking for answers in another region of the answer space where they will not find the answer.

This drawback has to do with the quality of the training data and it is an inherent disadvantage to data-driven approaches.

All strategies demanded more time for answering date-questions, because of the large amount of training data. Results suggest that the deep answer-context alignment can be used for increasing the accuracy of the answer extraction stage, but at some point, it turns to require a huge amount of computational resources. Results also suggest that ad-hoc linguistic processing improves the accuracy and performance of our methods. But, if this linguistic support is not adequate, data-driven approaches tend to perform better. Overall, data-driven approaches obtain a good performance, but they need a huge amount of **different** contextual paraphrases and computational resources in order to sharply identify the answer from its context.

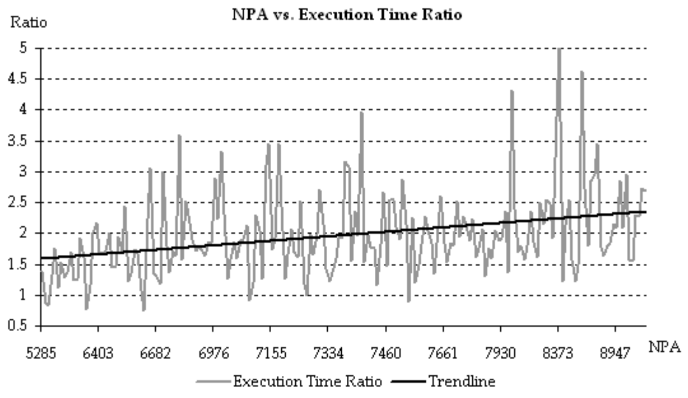

Figure 5: NPA versus Execution time ratio.

In addition, an extra baseline was implemented in order to measure the contribution of GA to speed the search up. This new baseline performs an exhaustive search through the whole answer space, evaluating the fitness of individuals according to equation 5. This baseline was compared with GAQA by fetching $N = 50$ snippets for each question in the test set of *Prime Ministers*. This comparison assists us in checking the following four aspects. First, the respective estimate of NPA is 5500 (equation 6), while its average empirical value was 7481.57 with a standard deviation of $\pm$ 855.84. Second, the average execution time ratio of this new baseline to GAQA is 1.97 with a standard deviation of $\pm$ 0.69. Since, GAQA and the new baseline share the same goal function, the increase showed in figure 5 corresponds directly to the grown in size of the answer space. Then, the larger the answer space is, the faster GAQA is with respect to the baseline. Third, GAQA did not discover an answer ranked top five by the new baseline for the following two countries: "*Japan*" and "*Niger*". Lastly, the MRR value of Google for this test set was 0.447. This value concerns the rank of the first **snippet**, within the top five, containing the correct answer, and it is noticeably lower than the value obtained by GAQA and GAQA+GASCA as well as the Baseline. All things considered, GAQA provides a faster search that finds right answers and misses few ones.

6 Future work

One possible way of enhancing our methods is by the addition of morphology knowledge to the search. To illustrate this, let us think on the following sentence: *"The sunglasses were invented by James Ayscough."*:

The	radio	was	invented	by	Nikola Tesla
The	radio	was	invented	by	Guillermo Marconni
The	sunglasses	were	invented	by	James Ayscough

Table 14: Effects of morphological inflections on the alignment.

when GAQA aligns this sentence with tuples on the QA-STORE (see table 1), it discovers enough contextual evidence to consider *"James Ayscough"* to be a promising *n-gram*. But, the interesting fact here is that the word *"were"* does not contribute to the alignment, bringing about a decrease in the value of the fitness from four to three (see section 3.2.1). In many NLP tasks, words are stemmed in order to avoid counting several morphological inflections of the same term as occurrence of different words. This allows their real occurrences to be reflected. Normally, this process consists of splitting the term into its *stem* and *ending*. For example, the *stem* *"kiss"* can be found in words like *"kiss"* and *"kissed"* (kiss+ed) as well as *"kisses"* (kiss+es). Then, any occurrence of one of this three variations is considered as an occurrence of the *stem* of the word (*"kiss"*). But, identifying and removing endings of some words is only a rough approximation, because some irregular verbs like: *"be"*,*"go"*, etc, inexorably leads the task to use additional linguistic knowledge. It is worth to highlight that two different words can share the same *stem*, such as: *"neutralise"* and *"neutron"* (van Rijsbergen (1979)). In our instructive example in table 14, the alignment after morphological analysis is as follows:

The	radio	be	invent	by	Nikola Tesla
The	radio	be	invent	by	Guillermo Marconni
The	sunglasses	be	invent	by	James Ayscough

Table 15: Alignment of stemmed words.

Morphology analysis aims for reducing the variability of natural language texts, and hence has a stronger impact upon the alignment of sentences in languages with a richer Morphology such as German and Spanish. For example, the word combination *"was invented"* can be translated into Spanish as *"fue inventado"* or *"fue inventada"*. Fortunately, tools, like MontyLingua, that perform this morphological analysis exist for several languages and are available for public usage on the Internet[16]. Accordingly, in the working example in table 15, the fitness after morphology analysis is four.

With regards to multilinguality, it is positively encouraging to extend our strategies to other languages, especially free-order languages, because this extension would consist chiefly of replacing the QA-STORE and the stop-list. Obviously, the performance in the new language will rely upon its redundancy on the Web.

Another possible way of enriching our search is by making use of syntactic infor-

[16]http://members.unine.ch/jacques.savoy/clef/index.html

mation such as part-of the speech (POS) tags. A *part-of speech*[17] is a lexical category that defines the specific syntactic or morphological behaviour of a word. Common categories include *nouns*, *verbs*, *adjectives* and *adverbs*, among others. POS tagging maps then each word to a particular part-of speech on the grounds of its definition and context. But, this mapping is extremely hard, because a vast number of word-forms are ambiguous, and thus having a list of pre-defined mappings is not enough. Like morphology analysis, tools[18] that automatically tag natural language texts exist (i. e. MontyLingua for English). The POS tags for our illustrative example are sketched in table 16:

The	radio	was	invented	by	Nikola Tesla
DT	NN	VBD	VBN	IN	NNP NNP
The	radio	was	invented	by	Guillermo Marconni
DT	NN	VBD	VBN	IN	NNP NNP
The	sunglasses	were	invented	by	James Ayscough
DT	NNS	VBD	VBN	IN	NNP NNP

Table 16: Sample of a tagged alignment.

In the example, it can be observed that the answer follows a concrete pattern (NNP NNP), which can be inferred from the `QA-STORE`. This inferred pattern can be used for filtering answer candidates, reducing hence the search space, and improving consequently the performance. Additionally, the reason for the misalignment can be deduced from table 16: the lexical category NNS (plural noun) differs from NN (singular noun) and brings about an inflection in the past tense of the verb (VBD) "*be*". Similarly to words, lexical categories can also be aligned, equally their distribution can be learnt by observing their relative position to the EAT. Here, multiobjective optimisation can perform a "*bi-dimensional*" search, where one function evaluates answer candidates at the word level and the other at the POS level. However, these relative POS distributions can also neatly be incorporated into equation 5 as the weights $\alpha(w_{sk}, Q)$.

To illustrate another possible way of adding POS tags to the search, let us recall the sentence S_1 presented in section 3.2.2: "*The helicopter was really invented in 1939 by Igor Sikorsky in Kyiv*". The POS tagged version of S_1 is as follows:

The/DT helicopter/NN was/VBD really/RB invented/VBN in/IN 1939/CD by/IN Igor/NNP Sikorsky/NNP in/IN Kyiv/NNP

`GASCA` can speed up the search by including POS information in its operators. For example, the mutation operator of `GASCA` could foresee that some lexical categories, like RB, are more unlikely to supply alignment, and selects therefore genes and their phenotypes according to a probability distribution that models this likelihood, while it is mutating an individual. One last remark regarding POS tags, web snippets contains truncated sentences, and thus the syntactic analysis carried out in these sentences is quite inexact. But, if sentences in the `QA-STORE` are also taken from web snippets, errors in the syntactic analysis would also be properly aligned, causing a mitigation of the unfortunate effects of truncation upon POS tagging.

[17]http://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html
[18]http://nlp.stanford.edu/software/lex-parser.shtml

GASCA can also take advantage of additional syntactic information such as *parse trees*. A *parse tree* is a tree that represents the syntactic structure of a sentence according to some formal grammar. Figure 6 shows the *parse tree* for our working sentence S_1 introduced in section 3.2.2. In this tree, the *Propositional Phrase* (PP) "*in 1939*" is in a frame signalling that it provides misalignment. Therefore, GASCA can speed up the search by swapping the value of a group of genes corresponding to constituents, such as PPs, instead of single words. Additionally, the QA-STORE can also consist of *parse trees*, and a GA-based strategy would accordingly align trees instead of words to discover new answers.

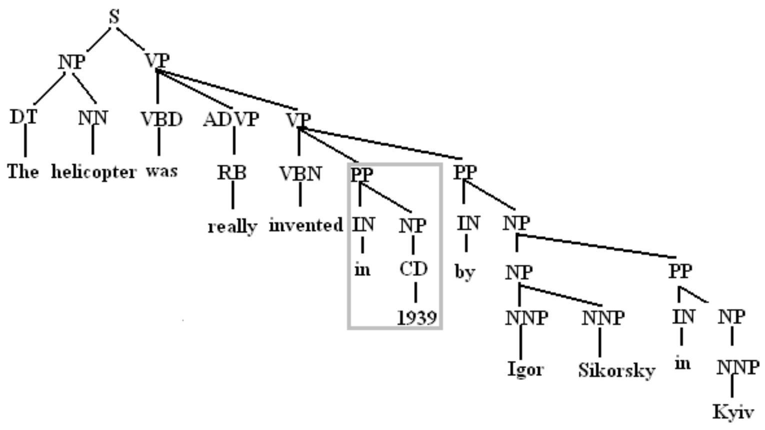

Figure 6: Parse Tree for S_1.

Since GASCA and PreGA got better results on different data-sets, it is greatly encouraging to take advantage of the ideas of (Tiedemann (2005a,b); Day et al. (2006)) for selecting answering strategies according to a set of given features extracted from the query. Furthermore, these ideas can also be extended to rerank and validate answers. These kinds of task are somehow hard to do in question answering systems (for example, see (Moldovan et al. (2004))).

Our strategies achieved good results taking advantage only of simple codifications, this is a promising factor, because genes can be added in order to improve the answer extraction process. In our methods, all occurrences of an answer candidate are equally fit, they do not draw any distinction between the contribution of different sentences to the final fitness, though these occurrences are inserted into completely different contexts. This distinction can help to give major priority to high fit sentences, causing a finer balance of the application of ad-hoc linguistic processing. Another extension in the gene representation is due to GASCA, the addition of two genes that signal the degree of compactness of the aligned blocks of words could be used particularly to distinguish the level of reliability of both contexts. In PreGA, the coding can be further exploited in order to answer more complex questions or perform more specific linguistic processing such as deep parsing.

7 Conclusions

Results obtained by GAQA shows that it is possible to extract answers from the Web to natural language questions by properly aligning previously annotated contexts with new sentences, in such a way that, this alignment lessens the dependence upon external lexical resources and it is robust to noisy training data, therefore, it substantially reduces the need of manual annotations or pre-defined ranked patterns. Specifically, results suggest that our strategies can cope with specific questions, specially those questions whose answers are inserted into contexts, for which do not exist a large amount of morpho-syntactical variations, this way the probability of matching patterns seen on training data and the context of new sentences increases. Additionally, results also show that `GAQA` can distinguish answers to some kinds of question (i. e. composers), for which the collection does not provide the necessary context to discriminate their role within snippets, this is, answers that are more difficult to be easily identified.

The improvement achieved by `GASCA` and `PreGA` shows that the heavy dependence upon the training set can be mitigated by: (a) aligning patterns in a more flexible fashion, and (b) balancing the contribution of linguistic processing to the answering process. A final remark is due to the processing time required by our strategies, our system is fully implemented in Java, time can be therefore substantially decreased by changing the programming language.

References

Aycinena, M., Aycinena, M., and Mulford, D. (2003). An evolutionary approach to natural language grammar induction. *Stanford CS 224N Natural Language Processing*.

Belkin, M. and Goldsmith, J. (2002). Using eigenvectors of the bigram graph to infer grammatical features and categories. In *Proceedings of the Morphology/Phonology Learning Workshop of ACL-02*.

Chalendar, G. D., Dalmas, T., Elkateb-Gara, F., Ferret, O., Grau, B., Hurault-Planet, M., Illouz, G., Monceaux, L., I., I. R., and Vilnat, A. (2003). A noisy-channel approach to question answering. *NIST Special Publication SP*.

Charles, L., Gordon, C., Cormack, V., and Lynam, R. (2001). Exploiting redundancy in question answering. *Journal of the American Society of Information Science*, 41(6):391–407.

Chen, J., Ge, H., Wu, Y., and Jiang, S. (2004). Question answering combining multiple evidences. In *TREC 2004*.

Coello, C. A. (2004). An introduction to evolutionary algorithms with applications in biometrics. In *Proceedings of the International Workshop on Biometric Technologies: Special Forum on Modeling and Simulation in Biometric Technology*, pages 51–67. BT'2004.

Cohen, W. W., Ravikumar, P., and Fienberg, S. E. (2003). A comparison of string distance metrics for name-matching tasks. In *IIWeb 2003*, pages 73–78.

Day, M., Lu, C., Ong, C., Wu, S., and Hsu, W. (2006). Integrating genetic algorithms with conditional random fields to enhance question informer prediction. In *Proceedings of the IEEE International Conference on Information Reuse and Integration (IEEE IRI 2006)*, pages 414–419.

Dumais, S., Banko, M., Brill, E., Lin, J., and Ng, A. (2001). Data-intensive question answering. In *proceedings of the tenth Text REtrieval Conference (TREC 2001)*.

Dumais, S., Banko, M., Brill, E., Lin, J., and Ng, A. (2002). Web question answering: is more always better? In *Proceedings of SIGIR-2002*.

Echihabi, A., Hermjakob, U., Hovy, E., Marcu, D., Melz, E., and Ravichadran, D. (2004). How to select an answer string? *Advances in Textual Question Answering*.

Echihabi, A. and Marcu, D. (2003). A noisy-channel approach to question answering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 16–23.

Figueroa, A. and Atkinson, J. (2006). Micai 2006: Advances in artificial intelligence. *LNAI*, 4293:985–995.

Figueroa, A. and Neumann, G. (2006). Advances in natural language processing. *LNAI*, 4139:423–434.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Co.

Holland, J. H. (2005). Genetic algorithms: Computer programs that "evolve" in ways that resemble natural selection can solve complex problems even their creators do not fully understand. http://www.econ.iastate.edu/tesfatsi/holland.GAIntro.htm.

Keller, B. and Lutz, R. (1997). Evolving stochastic context-free grammars from examples using a minimum description length principle. In *Workshop on Automata Induction Grammatical Inference and Language Acquisition, Nashville, Tennessee*. ICML-97.

Kintsch, W. (1998). Predication. *Cognitive Science*, 25:173–202.

Landauer, T. K., Foltz, P. W., and Laham, D. (1998). Introduction to latent semantic analysis. *Discourse Processes*, 25:259–284.

Lita, L. and Carbonell, J. (2004). Unsupervised question answering data acquisition from local corpora. In *Proceedings of the Thirteenth Conference on Information and Knowledge Management (CIKM 2004)*.

Moldovan, D., Harabagui, S., Clark, C., Bowden, M., Lehmann, J., and Williams, J. (2004). Experiments and analysis of lcc's two qa systems over trec 2004. In *TREC 2004*.

Mollá, D., Schneider, G., Schwitter, R., and Hess, M. (2000). Answer extraction using a dependency grammar in extrans. *Traitement Automatique de Langues (T.A.L.), Special Issue on Dependency Grammar*, 41(1):127–156.

Neumann, G. and Sacaleanu, B. (2006). Clef 2005. *LNAI*, 4022:429–438.

Otto, E. and Riff, M. C. (2004). Towards an efficient evolutionary decoding algorithm for statistical machine translation. *LNAI*, 2972:438–447.

Rinaldi, F., Dowdall, F., Kaljurand, K., Hess, M., and Mollá, D. (2003). Exploiting paraphrases in a question answering system. In *Proceedings of the second international workshop on Paraphrasing*, volume 16.

Savary, A. and Jacquemin, C. (2000). Reducing information variation in text. ELSNET Summer School.

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

Tiedemann, J. (2005a). Improving passage retrieval in question answering using nlp. In *Proceedings of the 12th Portuguese Conference on Artificial Intelligence (EPIA)*, pages 73–78.

Tiedemann, J. (2005b). Integrating linguistic knowledge in passage retrieval for question answering. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 939–946.

van Rijsbergen, C. J. (1979). *Information Retreival*. Butterworths.