Genetic Algorithms for syntactic and data-driven Question Answering on the Web

Alejandro G. Figueroa A.

Thesis Advisors:
PD Dr. rer. nat. Günter Neumann
Prof. Dr. Hans Uszkoreit

A Thesis presented for the degree of
Master in Speech Science and Language Technologies

LT-Lab - D.F.K.I.
Department of Computer Science
University of Saarlandes
Saarbrücken
July 2006

Dedicated to....

Through my short life, my Father and I liked to talk tall to each other and tried to solve the problems of the whole world in a couple of hours. One can imagine, any conversation which people usually hold while they are on the booze on the streets, in bars. I secretly relished the thought of being alone talking with him, I guess he also particularly savoured those precious moments. I always treasured the abiding memory of our time together.

My father utterly dedicated his entire life to doing research into mathematics, he was constantly pointing out the egocentric world of the research communities around the world. One of my most cherished possessions is one of those fleeting moments. We were sitting in our conventional family living room, and he suddenly started pointedly talking about how their colleagues often referred to each other in their publications. My father was an old-fashioned researcher. By old-fashioned, I mean, lucky owners of a different outlook on science and life, like Nikola Tesla or Albert Einstein. He was always confidently expecting that his work were a real contribution to science and people. By old-fashioned, I also mean that he was an outspoken person, he was always sharply criticizing his work and the work of others. As a natural consequence, he was always between the thin line of love and hate with his peers.

That night, I came up with the wacky and exciting idea of annoying him and I brought up the controversial topic of the usefulness of the current research around the world. Obviously, my deliberate intention was to directly question the usefulness of his work. Contrary to my expectation, the straight answer I received that night was: "Well, I live with the fervent hope that someone else, someday, will do something useful from my work; one day, probably after I depart this life. For the moment, it only feeds my ego.". Later, he added: "Nowadays, authorship seems to be more important than contribution". That night the conversation ended focusing on the two sides of the coin of the authorship and the impact of publications.

Taking into account these inherent attributes of our human nature, I truly dedicate my small work to the families, friends and all the people behind the scenes who help researchers to present their work -with an unknown impact- around the world. I strongly believe that those persons are like small heroes of science, like my mother. If their contribution to science would be published, there would be no journal or conference suitable to assess their work, and not enough sheets of papers, which could contain the necessary words for explaining how they solve their every day life problems. I dedicate this work then, to my gutsy mother, who patiently misses me badly while I am away. Maybe there is no mark or publication that could pay me for all the time that I have missed looking at her sweet eyes. Sometimes, when I fondly

gaze at the sky, starring at the stars unblinking, they repeatedly remind me of the sparkles in her puffy and velvety eyes, when I left. I also dedicate my work to my father who died while I was pursuing my studies here. To whom I eternally thank for all his invaluable pieces of advice.

I would like to dedicate my work to my friends, who tremendously helped me to start this long journey: Gonzalo, Arturo and Alejandro. Last but not least, I also thank God for making it possible to bring me to the land, where sapphires come from its rocks, and its dust contains nuggets of gold. No bird of prey knows this hidden path, no falcon's eye has ever seen it.

July 14, 2006

Genetic Algorithms for syntactic and data-driven Question Answering on the Web

Alejandro G. Figueroa A.

Submitted for the degree of Master in Speech Science and Language Technologies
June 2006

Abstract

This thesis describes a question answering system, which takes advantages of Genetic Algorithms for extracting answers from web snippets. These GA learn the syntactic alignment between pairs {sentence, answer} obtained from past QA cycles in order to identify and extract the most promising answers to new natural language questions. The answer extraction strategy is strongly data-driven using only language specific stop-lists and thus, the whole approach has a high degree of language independency. In this thesis, ideas of how to add linguistic processing to this data-driven search are also discussed. The strategies were assessed with different sets of pairs {question, answer}. Results show that this approach is promising, especially, when it deals with specific questions.

Declaration

The work in this thesis is based on research carried out at the LT-Lab Research Group, in the Language Technology Lab, Saarbrücken, Germany. No part of this thesis has been submitted elsewhere for any other degree or qualification and it all my own work unless referenced to the contrary in the text.

Acknowledgements

First, I would like to thank all the researchers, who contributed to the background of this thesis with their work. This thesis is grounded on some of my work developed as a research assistant in the QUETAL Group in the LT-Lab at DFKI[1]. This work is briefly introduced in section 2.2 and it has been accepted for publication in the 5th International Conference on Natural Language Processing. For this reason, this section has been written together with my advisor. The reference can be found at [52]. Secondly, some parts of section four, some ideas and conclusions came from a published work that will be presented at the next conference on Knowledge and Experts Systems, the exact reference can be found at [53]. Thirdly, extracts of chapter one and chapter four are included in an "accepted work under revision" on the Journal of Expected Systems (see [55]). Fourthly, section 2.1 is part of a work submitted to the 5th Mexican International Conference on Artificial Intelligence (see [54]). Fifthly, the endings of chapters five through eight are still under discussion between my supervisor and me. Lastly, I would like to thank those who prepared this template.

[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 projects `Quetal` (FKZ: 01 IW C02) and `HyLaP` (FKZ: 01 IW F02).

Contents

List of Figures

List of Tables

Introduction

This thesis presents a question answering system, which takes advantage of the redundancy existing on the Web in order to extract answers from web snippets. This extraction process is guided by a purpose-built Genetic Algorithm, which learns patterns from previously annotated tuples {question, sentence, answer} and aligns these patterns in order to readily identify answers to new natural language questions.

The answer extraction strategy is strongly data-driven, and it takes advantage of only language specific stop-lists and thus, it guarantees a high degree of language independency. In this system flows strategies from different fields: Linguistics, Machine Learning and Artificial Intelligence. Results show that this approach is promising, especially, when it deals with questions aiming for a location or name of persons. Our approach lessens the dependence upon external lexical resources such as lists of locations, names, etc.

Main Contribution

This work presents a data-driven approach to question answering which takes advantage of syntactical distributional patterns for discovering answers to new natural language questions on the web. These syntactical distributional patterns are directly learnt from the relative position of words with respect to the *expected answer type* from previously annotated pairs {sentence, answer}. These patterns are aligned with sentences presented in retrieved snippets in order to extract answer candidates to new questions. This alignment is performed by purpose-built *Genetic Algorithms* (GA), which efficiently test the most promising alignments.

Additional Contributions

This work also presents two other contributions: (a) a baseline for Question Answering on the Web based largely on a well-known metric for measuring the power of terms as an index, this metric is *Term Frequency - Inverse Document Frequency*, (b) a strategy for balancing the contribution of data-driven and linguistic processing to the answering process.

Improvements

The main drawback to the model presented in this work is that different expected answer types behave in a similar way. It is therefore perfectly clear that this model will not be able to deal efficiently with all kinds of questions and languages. Consequently, semantic processing is a key tool for the answer extraction process, specially, for tackling this problem head-on.

Another drawback to our strategies is that *Genetic Algorithms* do not absolutely guarantee to test the best individual while they are extracting answers. That is, the possibility exists that GA will not detect the answer on the text. The impact of this disadvantage is mitigated by the large-scale redundancy of the Web.

Thesis organization

This thesis is organized as follows: chapter one to three go over theoretical foundations and relevant previous work, chapter four to seven clearly present the new strategies, and chapter eight and nine show results and draw conclusions.

Each chapter focuses its attention on a special issue: chapter one describes the relevant state of the art concerning the Web Question Answering problem, chapter two describes at a greater length strategies for learning the syntactical behaviour of words from raw text, chapter three goes over the foundations of *Genetic Algorithms* (GA), chapter four discusses evaluation issues and the design of the *Baseline*, chapter five presents our Web Question Answering System and the Core Genetic Algorithm, chapter six describes a data-driven improvement to our system, chapter seven deals at a greater length with the enrichment of the extraction process by adding linguistic processing, chapter eight shows and discusses experimental results, and chapter nine draws some conclusions.

Conclusions

This thesis focuses special attention on data-driven methods for Question Answering on the Web. Two important issues are pointed out in this work: a learning model and an alignment heuristic.

Chapter 1

State of the Art

In the last three decades, substantial advances in different areas of computer science have had a significant impact on our every day lives. Thirty years ago, computers could not do most of the demanding tasks that they currently do (i.e. *Image Processing*, *Internet*, etc), because they were enormous, slow and expensive electronic devices. At that time, nobody imagined that almost everybody could have one at home and/or at work. In the last years, the remarkable reduction of their size has made it possible to not find them only everywhere, but to bring them everywhere as well. In addition, the vast improvement in the speed has also contributed to this impact, Computers are much faster and the scope of applications grows every day. Nowadays, we find applications that cover tasks like *calendars*, *word processors*, *movie players*, *networking*, *and much more*. Today, there is no room for doubt Computers are a necessary tool in our lives, due to their coverage of applications and reasonable price.

The rapid increase in the use of computers and storage capacity led us to connect them in such a way that users were able to easily transfer information from one computer to the other. In the beginning, small home-oriented, low-speed networks were developed, whereas today we find computers connected to each other around the whole world in what we know as *internet*.

Internet is a tremendous source of information, which demands computers to process a huge amount of data from all over the world in an efficient way. Every time a user has a need for a particular piece of information, a computer -or a set of them- must match his/her request with the right source. However, this matching task involves dealing with many challenges such as understanding a particular user request, choosing relevant documents and/or picking reliable sources. This is far from being a trivial task, matching the need of the user with the right source means dealing with information sources in many languages and in many formats: *web-sites*, *documents*, *web documents*, *videos*, *pictures*, *etc.*

1.1 Question Answering Systems

Question Answering Systems (QAS) try to find answers to natural language questions submitted by users, by looking for answers on a set of available information sources, which can be spread on a single machine or all over the internet. Broadly speaking, QAS have two major components [16]:

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 involves the efficient indexing of documents and the design of a fast algorithm that computes snippets. The latter has to do with identifying correctly the answer to the request of the user on the previously selected set of documents. For the efficiency sake, extracting answers from snippets is clearly desirable, in that way, QAS avoid downloading and processing a large amount of documents. Certainly, this is not an easy task. On the one hand, snippets provide: (a) localized contextual paragraphs that are highly related to the query, (b) these localized contextual paragraphs express ideas and concepts by means of different paraphrases, which consist of morphological, semantical, orthographical and syntactical variations of these ideas and concepts [31], which makes possible to find a paraphrase where the answer is easily identified. 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 more difficult and dependent on 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 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 answer ("14 March 1879" ⇔ "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*" ⇔ "*physics nobel prize Albert Einstein*"). Furthermore, snippets three and four are semantic variations ("*physics nobel prize Albert Einstein*" ⇔ "*German-American physicist*"). The last two snippets show breaks inserted deliberately by the search engine. 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.*").

The trend of QAS is to start by analyzing the query, in order to select an adequate strategy for answering the question [7, 9, 13, 23]. This initial phase is called *Query Analysis*. There are different approaches to *Query Analysis*, but in most cases it 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, and this categorization 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 [21], in contrast to other approaches, where the number is restricted to a few, but broad and general categories [23]. In [21], the EAT falls into one type out of a typology of 185 types: *Abstract*, *Semantic*, *Relational*, *Syntactic*, etc. In [23], *five* types of questions were identified: *Factoid*, *List*, *Other/Definition*, *Inferred Based*, *Semantics in Text*. Here is a brief summary of each of the categories in [23]:

1. **Factoid:** The answer or the paragraph of the answer is identified by simply keyword matching: "*When was Sting born?*".
2. **List:** The answers are obtained by processing multiple sources of documents: "*Where can I find a Mc Donalds in Europe?*".
3. **Other/Definition:** The answer solely depends on the context of previous questions: "*When was Eric Clapton born?, Where?*".
4. **Inferred Based:** are factoid questions that need deep processing for extracting the answer from the paragraph: "*Who invented the Radio?*".
5. **Semantics in Text:** are questions that can only be answered by means of deep processing: "*How did Adolf Hitler died?*".

Usually, other categories are sub-categories of these five general categories. However, the answer does not necessarily need to belong to only one class, it could be a member of many classes. Consequently, in some approaches, the *expected answer type* is viewed as a distribution over different possible categories [12], where some classes are more likely than others for some sorts of questions.

Somehow, the strategy for answering a question is determined by the category it belongs. The EAT guides the passage and sentence selection, which is later ranked according to a set of features [20, 23]. For example in [9], the answer extraction schema is based on the EAT. If the EAT aims for a name entity, they determine some lexical and syntactical clues from the query in order to use them in the answer extraction module. If it does not aim for an entity, it determines the pattern associated to the answer along with some semantical relations between words in the query and the possible answer. These kinds of approaches disclose another important issue on *Query Analysis*, it does not only provide the EAT, it also provides of 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, in which the answer can easily be identified. Normally, QAS extract paraphrases at the sentence level [21]. The rules for identifying paraphrases can be written manually or learnt automatically [10, 21], and they can consist of pre-parsed trees [21], or simple string based manipulations [10]. Paraphrases are learnt by retrieving sentences that contain previously known question-answer pairs. For example in [21], anchor terms (like "Lennon 1980") are sent to the Web, in order to retrieve sentences that contain query and answer terms. Patterns are extracted from this set of sentences, and their likelihood is computed in proportion to their redundancy on the web [7]. In both cases, the new set of retrieved sentences is matched with paraphrases in order to extract new answers. Another advantage of a huge set of paraphrases [10] is that they considerably decrease the need for deep linguistic processing like: anaphora resolution, uncovering complex syntactical or semantical relations, synonym resolution, etc. In some cases, it reduces the extraction to a pattern matching by means of regular expressions [21].

Redundancy is an important tool for *open-domain question answering systems*. In [10], they systematically explored the correlation between the performance of QAS and the number of snippets. They concluded that the performance of their system sharply increases until fifty snippets, increases slower from 50 to 200 snippets, peaks at 200 snippets, and flatters and falls off slowly above 200 snippets. The major drawback to these kinds of systems is that it is hard to find a massive redundancy on domain specific topics. Hence, linguistic processing is still the core of *domain-specific question answering systems*. In [22], they 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 word reordering matching (by a set of rules or syntactical transformations), they are also considered as different syntactical variations and mapped to the same logical representation. From this representation, called *Minimal Logical Form* [24], they extracted answers by means of a logical proof. As a result, they observed that domain-specific QAS must deal with unknown specific lexicon, abbreviations and acronyms, and for this reason, linguistic processing is still a vital issue. On all sides, redundancy is crucial for both types of QAS. But, the more specific the engine is, the more linguistic processing

it needs. A large-scale redundancy also provides of a way of validating whether an answer is correct or not, and identifying unreliable sources readily.

In more practical terms, strategies based on paraphrases perform better when questions aim for a name entity as an answer: *Locations*, *Names*, *Organizations*. But, they perform poorly when they aim for *Noun Phrases* [21]. Due to the huge amount of paraphrases, statistical methods are also used for extracting answers. In [16], a statistical strategy which scores a given sentence and a substring of the sentence, that is likely to be the answer, according to the query is presented. 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 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 on 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. Hence, the open research questions are: For which sorts of questions is linguistic processing more appropiate?, How can QAS know a priori, how hard is to find the answer for a given question?. These questions can be summarized in: When is it appropiate to use deep processing, statistical based approaches and strategies based on distributional patterns (like frequency counts, n-grams, etc)?.

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 combining different kinds of strategies, in order to re-rank answers, has taken off. In QA jargon, this re-ranking step is know as *answer validation*. In [21], 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, negative feedback, etc. Results show that a good strategy for combing answer extractors can considerably improve the overall performance of QAS (see also [23]).

Question answering systems restarted to catch the attention of research groups, when the *American Institute of Science and Technology* (NIST) introduced the Question Answering Track in the *Text REtrieval Conference*[1] (TREC). Since 1999, this track takes place every year, and during this track a challenge between different QAS around the world is held: *Carnegie Mellon University* [32], *IBM T.J. Watson Research Center* [33], *Microsoft Research* [34], *MIT Computer Science and Artificial Intelligence Laboratory* [35], *University of Amsterdam* [36], *University of Edinburgh* [37], *University of Sheffield* [38], amongst many others. In order to compare the performance and the efficiency of different systems, TREC provides a set of questions and a target corpus, from where QAS are challenged to extract answers. This corpus and the set of questions vary from year to year. TREC also provides

[1] `http://trec.nist.gov/`

answer patterns which are used for measuring the exactness of answers discovered by QAS. As a logical consequence, the TREC corpus has become an invaluable set of question-answer pairs, that are used as a common ground to evaluate QAS. Some samples of factoid questions from the TREC corpus are:

Who was the first American in space?

The answers in the corpus are: "*Alan Shepard*" or "*Shepard*". Another illustrative example:

Who was elected president of South Africa in 1994?

The answers are "*Nelson Mandela*" or "*Mandela*". Questions can be more complex and do not necessarily aim at a single word or an entity. For instance:

Why can't ostriches fly?

One of the answers provided by TREC is "*wings that are too small to keep them aloft*". The strategies for dealing with the TREC challenge widely differ from one team to the other. The team of the *University of Edinburgh* presented his QED system. QED classified the EAT in twelve categories: *reason*, *manner*, *color*, *location*, *definition*, *count*, *measure*, *date*, *location*, *name*, *abbreviation*, and *publication*. This system takes advantage of deep linguistic processing such as *Categorial Grammar* and *Discourse Rhetorical Theory* (DRT). The answers are extracted by the unification of DRT representation of the query and selected passages. At the answer validation step, they used Google API[2] for improving the accuracy of the final rank of answers. QED also takes advantages of alignment algorithms for expanding the set of answers of list questions [39, 40]. The system of the *University of Amsterdam* (XQuesta) uses senses of Wordnet[3] for determining the EAT and a Name Entity Recognizer for dealing with factoid questions. The *MIT Question Answering System* takes advantages of resources like: Yahoo, Google, Wikipedia[4], Yahoo or Google as a source of hypernyms and synonyms, etc. This system also uses Wikipedia for answering list questions, and their question analysis tool identifies relative clauses.

TREC focuses special attention on the English Question Answering task, whereas the Cross Language Evaluation Forum (CLEF) deals with the Multilingual and Cross-Lingual tasks. The former task consists essentially in finding answers within collections of documents in the language of the query prompted by the user. The latter attempts to find answers in pairs of queries and a collection of documents of different languages. CLEF[5] has built a framework for testing, tuning and evaluating *Information Retrieval* and *Question Answering* systems operating on European Languages in both monolingual and cross-lingual contexts. This framework consists predominately of eight collections which contain news articles in eight different languages: *Dutch*, *English*, *French*, *German*, *Italian*, *Russian*, *Spanish* and *Swedish*.

[2] `http://www.google.com/apis/`
[3] `http://wordnet.princeton.edu/`
[4] `http://en.wikipedia.org/`
[5] `http://www.clef-campaign.org/`

In this corpus, variations of languages are also taken into account, this means it makes allowances for Portuguese from Brazil and Portugal, US and British English and Swiss French. This is an additional factor, because significant differences in orthography and lexicon across these pairs of languages exist [56]. QAS must be therefore robust enough to cope with these variants. The sorts of question provided by CLEF and TREC slightly differ. In particular, the Question Answering CLEF 2005 (QA@CLEF-2005) considered three sorts of question:

1. **Temporally Unrestricted Factoid** questions aim for answers such as address locations, persons, measures, etc. For example: "*Who invented the paper clip?*" (John Vaaler).

2. **Temporally Restricted Factoid** questions are also factoid questions, but they assume one of the following three temporal restrictions:

   (a) A **Date**: "*Who won the soccer world championship in Germany in 2006?*" (Italy).

   (b) A **Period**: "*Who won the Wimbledon Grand Slam seven times between 1993 and 2000?*" (Pete Sampras).

   (c) A **Event**: "*Who stopped the Pete Sampras eight wins in a Row of Wimbledon?*"(R. Kracijek).

3. **Definition Questions** address exclusively organizations and people: "*Who is Roger Federer*?" (current best Tennis Player).

Since Systems that take part into the CLEF competition must handle resources associated to different languages, the complexity of their architecture dramatically increases. Looking upon results of the QA@CLEF-2005 track, DFKI LT-Lab obtained the best results for a pair of the two most spoken languages: *English* and *German*. In order to determine the type of the question, this System [57] starts by analyzing the query. The question type is used by the system controller for picking an adequate answering strategy. In the case of factoid questions, this System extracts answers at the sentence level by identifying the following answer types: PERSON, NUMBER, ORGANIZATION and LOCATION as well as DATE. In the case of temporally restricted questions, one of the most interesting aspects of this answering strategy is that they formally split the query into two sub-queries [60]. The first sub-query refers to the "*timeless*" proportion and the second to the temporally restricted part. Later, answers to the restricted part of the query are used for constraining the "*timeless*" part. In the case of the definition questions, linguistics patterns are used for distinguishing descriptions: *appositions* and *abbreviation-explanation*. Eventually, this System takes advantage of the Web in order to validate answers. Currently, making allowances for the multilinguality of Web for extracting or validating answers is often used by Question Answering Systems [37, 58]. For the cross-lingual tasks, questions are translated into the language of the collection of documents by means of several translation engines, the well-formedness of translated queries is then assessed by a linguistic parser. The most well-formed queries were used for extracting answer afterwards.

Evaluation is a crucial point in QAS. Even though, it is possible to determine which is the best system in coping with a given set of questions and corpus (normally, the TREC corpus), it is extremely difficult to assess which system is better than other in the overall sense. For starters, the linguistic phenomena on natural language documents is not yet well understood. For this reason, it is unclear how to properly assess the complexity of answering a particular question. Secondly, QAS have several components and modules that make difficult to sharply distinguish the contribution of each of them. Furthermore, the increasing use of machine learning techniques makes this task even harder, because they usually aim for being independent on the language and the corpus as well as the set of questions, thus, it is hard to infer if a correlation between their performance and a particular kind of question, corpus or language exists.

To sum it up, assessing QAS is not a trivial task, it has to do with evaluating complex software architectures and the uncertainty of the target phenomena. Therefore, it is not desirable that an evaluation focus on only one score value. The absence of exhaustive evaluations is an undesirable problem of the research in Question Answering Systems.

1.2 Conclusions

In this chapter, the main features of *Question Answering Systems* were introduced. In particular, two main components were discussed: *Question Analysis* and *Answer Extraction*. The discussion focused attention on the significance for the question answering task of the *expected answer type*, redundancy and paraphrases as well as the trade-off between different strategies.

Lastly, this chapter also highlighted the topic of the evaluation of Question Answering Systems and the TREC and CLEF competitions.

Chapter 2

Acquiring Syntactic Categories

The most commonly used document representation is known as the *Vector Space Model* (VSM) [17]. Here, a document D is represented as a vector in a space in which each dimension is associated with the frequency of one word w_i in the dictionary W.

$$D = (freq(w_1), freq(w_2), \ldots, freq(w_\omega)) \in \Re^\omega$$

In this representation, some grammatical information is lost because the order of words and punctuation is ignored leading to broken phrases [18]. For example, "*Albert Einstein*" is split into "*Albert*" and "*Einstein*" without representing their syntactic relation. This model also does not take into account the role of words as modifiers in their local context, or as suppliers of the predicate or argument of the main proposition being expressed.

The role of a word in a text is given by its *syntactic category* (i.e., *noun*, *verb*, *adjective*). From the statistical viewpoint, *syntactic rules* involve distributional patterns, whereas in linguistics, *distributional analysis* is referred to as the study of syntactic properties that are in essence distributional. Even though *distributional analysis* tries to model this syntactic phenomena, it is well-known that it can not deal with the semantic phenomena presented on natural language text.

This chapter is organized as follows: section 2.1 describes two approaches for learning the syntactic behavior of words on unstructured text, section 2.2, describes two approaches for using the inferred syntactic behavior of words in question answering, and section 2.3 draws some conclusions.

2.1 Learning Syntactic Categories from raw text

Many efforts have been put in modelling the syntactic behavior of words using unsupervised mechanisms [1,2]. In these two approaches [1,2], each word $w_i \in W$ is represented by two vectors, called *syntactic context vectors*. The dimensions of the first vector $\phi^l(w_i)$ represent how often the other words in W appear immediately to the left of w_i, whereas the second vector $\phi^r(w_i)$ follows a similar strategy for words that appear immediately to the right.

To illustrate, consider the next two sentences: "*The thermometer was invented by Galileo*" and "*The zipper was invented by Judson*". The *syntactic context vectors* of these sentences are sketched in the following matrices[1]:

	by	Galileo	invented	Judson	the	thermometer	was	zipper
by	0	0	2	0	0	0	0	0
Galileo	**1**	0	0	0	0	0	0	0
invented	0	0	0	0	0	0	2	0
Judson	**1**	0	0	0	0	0	0	0
The	0	0	0	0	0	0	0	0
thermometer	0	0	0	0	1	0	0	0
was	0	0	0	0	0	1	0	1
zipper	0	0	0	0	1	0	0	0

Table 2.1: *Left syntactic context vectors.*

	by	Galileo	invented	Judson	the	thermometer	was	zipper
by	0	1	0	1	0	0	0	0
Galileo	0	0	0	0	0	0	0	0
invented	2	0	0	0	0	0	0	0
Judson	0	0	0	0	0	0	0	0
The	0	0	0	0	0	1	0	1
thermometer	0	0	0	0	0	0	1	0
was	0	0	2	0	0	0	0	0
zipper	0	0	0	0	0	0	1	0

Table 2.2: *Right syntactic context vectors.*

From tables 2.1 and 2.2, we realize that the matrix of the *right syntactic context vectors* is the transpose of the matrix of *left syntactic context vectors*. In table 2.1, we read that "*by*" appears two times to the right of "*invented*", and in table 2.2, "*invented*" appears two times to the right of "*was*". The main problem of the *syntactic context vectors* is that the degree of overlap can not be computed in the original vector space due to their sparseness. A simple similarity measure based on cosine can draw misleading classifications, even though the frequency of words is high. A good example is in [1]: "*a*" and "*an*" do not share any neighbours, because "*an*" appears whenever the sound of the next word starts with a vowel and "*a*" with a consonant, then the similarity is zero, but they have the same syntactic category.

In both approaches, they represented *syntactic context vectors* in another specially designed space, in which different syntactical categories show distinctions. Consequently, they found that *syntactic context vectors* of words contain the information about their syntactic behavior.

[1] Along the chapter, instructive values are in bold numbers.

2.1.1 Distinguishing Different Syntactic Categories

The first approach is due to Goldsmith and Belkin [2], who constructed a nearest-neighbor graph in which vertices represented words and edges pairs of words whose distribution in the corpus was similar. For this graph, they used the top 500 and 1000 frequent words. For each pair of words, the cosine of the angle of their *syntax context vector* was computed, and the 5, 10, 20 and 50 closest neighbors were selected. From this matrix, they built a canonical representation C, in which a value of zero was assigned to every element in the diagonal and wherever there was a zero in the original matrix, a value of one was assigned whenever a value was greater than zero in the original matrix.

They defined a diagonal matrix E, in which each value is the degree of each vertex. Then, they compute the normalized *laplacian* of $E - C$. The *laplacian* is a positive semi-definite symmetric matrix, therefore, all eigenvalues of the matrix are non-negative. The first and the second eigenvectors -corresponding to the lowest eigenvalues- derived from each *syntax context vector* were used to build a graphic representation of the syntactic behaviour of the words in the corpus. These vectors have a coordinate for each of the K most frequent words in the corpus. Eventually, they concluded that using these lowest-valued eigenvectors provides a good graphical representation of words, in the sense that words with similar left-hand neighbours will be close together in the graph.

Even though this strategy does not lead to a sharp distinction of syntactic categories, it can distinguish syntactically heterogeneous set of words [2]. The strategy was evaluated for two languages French and English. For English, the syntax category of many constituents (i.e., *non-infinitive verbs*, *infinite verbs*, *nouns*, *etc*) were correctly inferred. For French, other categories such as *female nouns*, *plural nouns*, *finite verbs*, *etc.* were clustered.

2.1.2 Acquiring Syntactic Behavior in presence of ambiguity

In [1], a model for the acquisition of syntactic categories from raw text in presence of ambiguity is introduced. In this model, called TAG SPACE, two matrices are built from the *syntactic context vectors* of the 250 most frequent words. The *Singular Value Decomposition* (SVD) was used for reducing the dimension of the two matrices and for solving the problem of sparseness of the data. The dimension of the matrices in the reduced space was 50 and they used the *group average agglomeration algorithm* for clustering.

In addition, this approach did not take advantage only of the *syntactic context vectors* of w_i like in [2], it also considered the *syntactic context vectors* of the preceding and following words. In this way, they were able to improve the accuracy of the learning process. This issue had a significant impact on the quality of results, because this approach was capable of clustering words having an ambiguous behavior.

As well as that, the accuracy of the learning process was also improved by a new kind of *syntax context vectors*, called *generalized context vectors*. These vectors were obtained by counting frequencies of classes of words -in the reduced space- that appeared to the left and to the right of each word. Furthermore, the performance was increased by assigning a special tag to pairs of classes that often occurred consecutively.

2.2 Learning Syntactic Categories for Question Answering

2.2.1 Syntactic Bonding/Chains of Related Words

In this approach, a document is a multi–set of all sentences which are extracted from all the N–best snippets returned by a search engine.[2] A vector–space document representation is proposed, based on the following binary variable:

$$X_{sik} = \begin{cases} 1 & \text{if the word } w_i \text{ is in the sentence } S_s \text{ at position } k \\ 0 & \text{otherwise.} \end{cases}$$

where $len(S_s)$ is defined as a function which returns the number of words in a sentence S_s. Then, the frequency of the word w_i in the document is given by:

$$freq(w_i) = \sum_{s=1}^{\sigma} \sum_{k=1}^{len(S_s)} X_{sik}, \quad \forall w, \ 1 \leq i \leq \omega \tag{2.1}$$

when w_j is a word in W, $1 \leq j \leq \omega$. For example, the document D="*John loves Mary. John kisses Mary every night.*" has two sentences determined by the dot. Considering that "w_1" is "*John*", then X_{111} will match the first occurrence of "*John*" and X_{211} the second. X_{s1k} takes the value of one for only this two occurrences. Therefore, $freq$("*John*") will be the sum of $X_{111} + X_{211} = 2$.

A document D is represented by the set of tuples:

$$D = \{< w_i, w_j, \epsilon, freq(w_i, w_j, \epsilon) >, \ \forall\, i, j, \epsilon, 0 \leq \epsilon \leq \Upsilon \ \wedge \ freq(w_i, w_j, \epsilon) > 0\}$$

where $freq(w_i, w_j, \epsilon)$ is the frequency of w_i with which it appears to the left of w_j, and ϵ is the absolute distance of their positions in the sentence:

$$freq(w_i, w_j, \epsilon) = \sum_{s=1}^{\sigma} \sum_{k=\epsilon+1}^{len(S_s)} X_{si(k-\epsilon)} X_{sjk} \tag{2.2}$$

For instance, $freq$("*John*", "*Mary*", 1) = 2 means that the pattern *John* * *Mary* was observed 2–times in the document D. $\Gamma(w_i, w_j, \epsilon, v) : W \times W \times N \times N \rightarrow \{0, 1\}$ is defined as a function that returns one if the $freq(w_i, w_j, \epsilon)$ is equal to v, otherwise it returns zero. Using this notation, it is defined:

[2]Very simple rules for mapping a snippet to a stream of sentences are used, basically standard punctuation signs as splitting points: colon, semicolon, coma, and dot.

$$G(v) = \sum_{i=1}^{\omega} \sum_{j=1}^{\omega} \sum_{\epsilon=1}^{\Upsilon} \Gamma(w_i, w_j, \epsilon, v) \tag{2.3}$$

$G(v)$ determines the amount of pairs of words that occur v times in the document. In the example, the only tuple that occurs two times is *John* * *Mary*, then $G(2) = 1$.

2.2.2 Ranking Sentences

A sentence S_s in a document is ranked by means of a specially designed matrix M. This matrix is constructed from the tuples in D in the following way:

$$M_{ij}(S_s) = \begin{cases} freq(w_i, w_j, \epsilon) & \text{if } i < j; \\ freq(w_j, w_i, \epsilon) & \text{if } i > j; \\ 0 & \text{otherwise.} \end{cases}$$

w_i and w_j are two words in S_s, ϵ is the distance between w_i and w_j, ϵ=*abs*(*i*-*j*), $0 \leq \epsilon \leq \alpha$, and α=*len*(S_s). This matrix models the strength of the relation or correlation between two words w_i and w_j in a sentence S_s.

The following filtering rule reduces the size of the representation of D and the noise of long sequences of low correlated words:

$$\forall i, j \;\; M_{ij} \leq \zeta \Rightarrow M_{ij} = 0$$

where ζ is an empirical determined threshold. This rule allows to remove some syntactic relations of a word which are probably not important. For example, the English word *of* is a *closed class word* and as such will co-occur very often with different words at different positions. However, if it is part of a phrase like *The President of Germany*, the definition above allows us to keep *of* in the noun phrase, because it typically occurs with short distance in such specific syntactic construction.

Then, the **rank of a sentence** S_s is defined as follows:

$$rank(S_s) = \lambda_{max}(M(S_s))$$

where $\lambda_{max}(M(S_s))$ is the **greatest eigenvalue** of the matrix M constructed from the sentence S_s, see also [19]. This eigenvalue gives the amount of "*energy*" or "*syntactic bonding force*" captured by the eigenvector related with λ_{max}. Note that computing the eigenvalues for a small matrix is not a demanding task, and M is a matrix of size $len(S_s)$, which in case of snippets is small. There are two more aspects of M that is worths mentioning:

1. $\forall i \; M_{ii} = 0 \Rightarrow \sum_{\forall i} M_{ii} = 0 \Rightarrow \sum_{\forall f} \lambda_f = 0$.
2. $\forall i, j \; M_{ij} = M_{ji}$, the *spectral theorem* implies that $\forall f \; \lambda_f \in \Re$, and all eigenvectors are orthogonal.[3]

The second aspect guarantees that for each sentence S_s, the value for $rank(S_s)$ is a real number.

[3]The *spectral theorem* claims that for a real symmetric n-by-n matrix, like M, all its eigenvalues λ_f are real, and there exist n linearly eigenvectors e_f for this matrix which are mutually orthogonal.

Algorithm 1: extractPredictedAnswers

```
input : M,S_s
1  begin
2      predictedAnswers = S_s;
3      if numberOfWords(w_i) > 3 then
4          forall w_i ∈ S_s do
5              flag = true;
6              forall w_j ∈ S_s do
7                  if M_ijε > 0 then flag=false;
8              end
9              if flag then replace w_i with "*";
10         end
11         predictedAnswers = split(S_s,"*");
12     end
13     return predictedAnswers;
14 end
```

2.2.3 Extracting Predicted Answers

The matrix M contains the frequency of each pair of words of S_s, which appears in this sentence and which has the same distance in the whole document. Sequences of word pairs which frequently co–occur with same distance in M are interpreted as *chains of related words*, i.e., groups of words that have an important meaning in the document. This is important if we also consider the fact that, in general, snippets are not necessarily contiguous pieces of texts, and usually are not syntactically well–formed paragraphs due to some intentionally introduced breaks (e.g., denoted by some dots betweens the text fragments). The claim is that these chains can be used for extracting answer prediction candidates. Algorithm 1 extracts predicted answers from a sentence S_s. It aims to replace low correlated words with a star, where a low correlated word is a word in a sentence that has a low correlation with any other word in the same sentence. Sequences of high correlated words are separated by one or more stars. Thus, low correlated words in sentences define the points for cutting a sentence into smaller units.

In order to assess this answer prediction strategy, traditional answer extraction modules based on lexical databases and pattern matching as well as stop-lists were implemented. The set of questions aimed for a LOCATION, PERSON or DATE as an answer, which answers were extracted from the predicted answers. Experiments were carried out in four languages: English, German and Portuguese as well as Spanish. Results showed that this extracting schema works well for a language like English, for which exists a massive redundancy on the Web (see full details in [52]). In contrast to the other three languages, for which there is not yet a large-scale redundancy on the web and the other of words is more flexible.

2.3 Acquiring Syntactic Patterns for Question Answering

In this work, an agent-based approach to question answering was introduced, in which the syntactic behavior of a particular EAT is learnt by means of the *syntactic context vectors* of question-answer pairs obtained in questions previously answered. From here, the learnt *syntactic context vectors* were used for extracting new answers to the same or new questions.

Let Q^* be the set of all questions that triggered the question answering system which aims to the same EAT. A is the set of answers to the questions in Q^*. Each component ϕ_i of the *syntactic context vectors* of the EAT of Q^* is given by:

$$\phi_i^l(EAT) = sum_{\forall A_j \in A} freq(w_i, A_j, 0)$$

$$\phi_i^r(EAT) = sum_{\forall A_j \in A} freq(A_j, w_i, 0)$$

Where $freq(w_i, A_j, 0)$ is the frequency in which w_i occurs immediately to the left of A_j, the sum over all $A_j \in A$ gives the frequency of w_i to the left of the EAT, and $freq(A_j, w_i, 0)$ is the homologous to the right. Next, $\phi^l(EAT)$ and $\phi^r(EAT)$ provide the information of the role of the EAT in the local context. For the simplicity sake, ϕ^l and ϕ^r refer to *syntactic context vectors* $\phi^l(EAT)$ and $\phi^r(EAT)$ respectively. If we consider the example in section 2.1, $\phi^l(INVENTOR)$ and $\phi^r(INVENTOR)$ are given by:

	by	Galileo	invented	Judson	the	thermometer	was	zipper
ϕ^l	**2**	0	0	0	0	0	0	0
ϕ^r	0	0	0	0	0	0	0	0

Table 2.3: *syntactic context vectors* of the EAT PERSON/INVENTOR.

ϕ^r is the null vector, because of the fact that no word occurs to the right of the EAT PERSON/INVENTOR.

Then, the **Syntactic Likelihood of an answer** A' is computed as follows:

$$L(A') = \phi^l \phi^l(A') + \phi^r \phi^r(A') \tag{2.4}$$

Where $\phi^l \phi^l(A')$ is the sum of the product of each component of the *left syntactic context vector* of the EAT, whereas the *left syntactic context vector* of the answer A', $\phi^r \phi^r(A')$ is the homologous to the right. Every answer is measured according to the amount of its context words in the snippets that match the context words of the EAT and the strength of this matching is according to their frequencies. The context words, which are assumed to occur more often in the context of the EAT have a stronger relationship with the EAT, and therefore, are stronger indicators for scoring a new answer.

Consider a document consisting of the following sentence: "*The kevlar was invented by Kwolek.*". The next two tables illustrate the *syntactic context vectors* of this document:

	The	kevlar	was	invented	by	Kwolek
The	0	0	0	0	0	0
kevlar	1	0	0	0	0	0
was	0	1	0	0	0	0
invented	0	0	1	0	0	0
by	0	0	0	1	0	0
Kwolek	0	0	0	0	**1**	0

Table 2.4: *Left syntactic context vectors* of the document.

	The	kevlar	was	invented	by	Kwolek
The	0	1	0	0	0	0
kevlar	0	0	1	0	0	0
was	0	0	0	1	0	0
invented	0	0	0	0	1	0
by	0	0	0	0	0	1
Kwolek	0	0	0	0	0	0

Table 2.5: *Right syntactic context vectors* of the document.

Then, the likelihood of each word to the EAT is given by the following table:

	The	kevlar	was	invented	by	Kwolek
$\phi^l\phi^l(A')$	0	0	0	0	0	2
$\phi^r\phi^r(A')$	0	0	0	0	0	0
Total	0	0	0	0	0	**2**

Table 2.6: *syntactic context vectors* of the EAT PERSON/INVENTOR.

The only word that contributes to the likelihood is "*by*" -when it is to the left to the EAT-. The only match occurs with the occurrence of "*by*" to the left of "*Kwolek*", as a result, it is the only word with likelihood greater than zero.

Experiments suggest that this likelihood is strongly affected by the data sparseness. However, the aim of the approach is not to cluster words to uncover their syntactic categories. The model assumes that every $A_j \in A$ has the same syntactic behavior in the local context of the answer, and therefore, the main interest was in the likelihood of A' in this context.

The strategy was assessed with a set of questions in English that aim for a LOCATION as an answer. This learning strategy was able to identify locations that are normally difficult to identify for traditional answer extraction strategies, some examples are: *Where is bile produced?* (liver), *Where is the Sea of Tranquility?* (moon), *Where is the volcano Olympus Mons located?*(mars), *Where does chocolate come from?*(Cacao), *Where is the Gateway Arch?* (Jefferson National Expansion Memorial).

2.4 Conclusions

In this chapter, two techniques for the unsupervised learning of the syntactic behavior of words in raw text were introduced. Both techniques are based on two vectors that model the local context of words. Also, two approaches that exploit syntactical information for answering natural language questions were introduced. One take advantage of the *syntactic context vectors*, and the other make inference from the relative position of words in text.

On the whole, the syntactic behaviour of words is a key issue in Question Answering Systems, especially, while the system is identifying answer candidates.

Chapter 3

Genetic Algorithms

Genetic Algorithms(GA) are computational models or algorithms, proposed by Holland [3], which are inspired by the natural selection process described by Charles Darwin. Charles Darwin in his book *On the Origin of Species by Means of Natural Selection, or The Preservation of Favoured Races in the Struggle for Life* stated:

- Species reproduce in order to survive, when they reproduce, they tend to produce offspring that are more suitable to the environment.
- As a result of the lack of resources, individuals compete with each other causing some of them to not survive.
- Due to some desirable genome, individuals that are more suitable to the competitive environment are more likely to survive.
- If the environment changes, the gnome of individuals will change in order to adapt to the new conditions of the environment.

The idea of the natural selection process is that each individual takes part in a fierce and cut-throat competition for resources and attracting mates. On the one hand, those individuals which are successful will have a relatively larger number of offspring. On the other hand, weaker individuals will produce a relatively smaller number of offspring. Then, individuals in the next generation will have a greater number of genes that come from stronger individuals. In that way, they will become more suited to their environment in the long term.

By mimicking this natural selection process, GA are able to solve real world search and optimization problems. A simple web search can show the enormous number of applications to which GA have been applied. In GA, each solution is like an individual in the population, and the value of how suitable (good) each solution is to the environment (problem) is given by a fitness score. Due to the fact that highly scored individuals have greater opportunities to reproduce, they spread their genes throughout the population and genes of weaker members tend to disappear gradually over the time. This selection mechanism guarantees that the GA will search the most promising areas of the search space, while they are evolving the population. Therefore, they will tend to converge to the optimal solution, or in the worst case, to a high quality solution in a short time.

There are many implementations of genetic algorithms in the literature, but it is still the case that most of the theory refers to the genetic algorithm introduced by Holland in 1975, which in some papers [51] is referred to as *the canonical genetic algorithm*. In this chapter, the *canonical genetic algorithm* is reviewed in order to explain the more essential and interesting features of GA. In this chapter, *The Schemata Theorem* is also discussed at a greater length, which is the traditional explanation for the outstanding performance of GA.

3.1 The Canonical Genetic Algorithm

The *Canonical Genetic Algorithm* (CGA) is the basis of almost all other implementations of genetic algorithms. CGA have predominantly concentrated the most important amount of theoretical research. In CGA, each individual is represented by a sequence of zeros and ones. For example, figure 3.1 shows an individual representing the number 178:

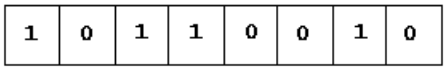

Figure 3.1: Binary Representation.

In the literature, this representation is commonly referred as *Binary Representation* of a *genotype* or *chromosome*. Individuals in GA are not narrowly restricted to binary numbers. However, normally, it is necessary to design specialized chromosomes to deal with the different characteristics of each specific problem which the GA target. Let us consider that the selection process of girls for promoting a new product during the next summer, a chromosome representing the conspicuous and central features of what an employee could look like:

	genes/genotype	Phenotype
Chromosome	height (cms)	180
	weight	65
	hair-color	red
	eyes-color	blue

Table 3.1: Non-binary chromosome representation.

Phenotype is the value or the information necessary for building an organism or individual. The fitness of an individual entirely depends on the *phenotype*. In GA, there are also other components that are highly dependent on the problem: The *goal or fitness function* and the *operators*.

Normally, GA are used to solving hard problems, where there is not known deterministic algorithm that can solve them in polynomial time. The difficulty of these kinds of problems, usually of a combinatorial nature, relies on the fact that the value of each variable has an important impact on the value of other variables, forcing these implicit effects to be considered while an algorithm is trying to solve the problem. In practical terms, a decision in the future is severely constrained by decisions at previous steps and the overall profit depends crucially on the sequence of decisions. This interaction is called *epistasis* and GA tackle it while they are trying to solve the problem. GA are also used for problems where algorithms exist that can solve them in polynomial time, but GA can solve them faster. Time is a critical issue for some real time applications, where the optimal solution is not strictly necessary, but being fast and closer to the optimal solution is enough to solve the problem.

The **Fitness function** measures how good a solution is, by returning a number proportional to the "utility" of a given *phenotype*. For example, if we consider that a utility function is equal to the square value of the phenotype $f(x) = x^2$, the fitness for the chromosome on our example would be $f(178) = 31684$. In real applications, we seldom find that the quality of a solution depends only on one factor, usually, a set of fitness functions is necessary to evaluate the quality of an individual. For instance, consider the *Vehicle Routing Problem*:"*A set of trucks must deliver goods to a set of clients visiting each of them only once, the chief of the company is interested in doing this by reducing the cost of the delivery and the number of trucks*". These kinds of problems, where we look for "*trade-offs*" instead of single solutions, are referred to as **Multi-objective Problems** [6] and GA are also capable of solving these kinds of problems.

The goal function is the most critical component in the performance of GA [4]. On the one hand, we desire a smooth and regular fitness function, so that chromosomes with reasonable fitness are close in the space to the ones with slightly better fitness [4], although it is normally not possible to build such a function. On the other hand, it must be efficient as every individual in the population of each generation will be evaluated, therefore the computational resources needed to compute the fitness value is a crucial point in the design of GA.

In practice, there are two well-known problems regarding the goal function [5]:

1. **Premature Convergence** occurs when GA converge to a highly local optimal, but not the optima, due to some genes of highly fit individuals which gradually take over the population. The capability of the GA of escaping from this local optima is given by mutation. There are also two other strategies for dealing with this undesirable situation: (a) changing the selection rule, (b) compressing the range of the fitness in order to prevent any "over-fit" individual from taking over the population [4].

2. **Slow finishing** is when the average fitness of the population is high, and the difference in the fitness of individuals is small. Consequently, there is an insufficient gradient in the fitness function to push the GA towards the maximum and it converges to a local optima [4]. This problem is tackled in a similar way as the premature convergence.

These two drawbacks are due to the fact that offspring of too highly fit individuals are strongly favoured, while GA are selecting its population for the next generation. These offspring normally belong to the same region of the space of their parents, and their fitness and/or genes differ slightly from the fitness value and/or genes of their parents.

The way that GA build the individual of the next generation is called **Reproduction** or **Recombination mechanisms**, and the instruments that perform this recombination are called **Operators**. Normally, and also in CGA, recombination mechanisms are not applied to all individuals in the population; usually, they are randomly selected according to their fitness value. The two most common recombination mechanisms are *crossover* and *mutation*:

1. **Crossover** The probability of crossing over (p_c) two individuals is between 0.6 and 1, usually 0.8. There are many ways of crossing over two individuals, sometimes specialized operators are designed according to a particular problem. In CGA, this operator is called *Single-point crossover*. Here, individuals are cut at a random point generating two heads and two tails, and tails are exchanged afterwards.

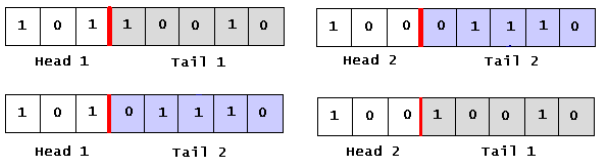

Figure 3.2: Single Point Cross Over.

When individuals are cut at two different random points, the operator is called *2-point crossover*. In this case, individuals are interpreted as a loop, in which the end and the beginning can simultaneously belong to the segment that will be exchanged.

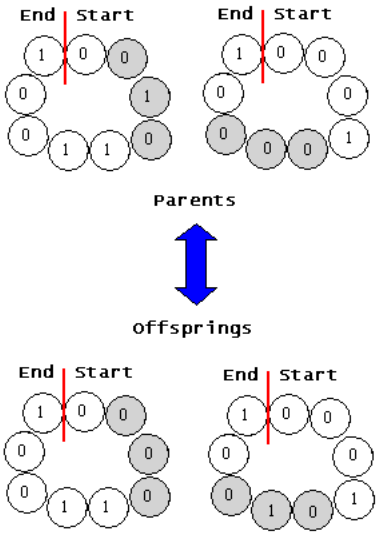

Figure 3.3: 2-Point Cross Over.

2. **Mutation** The probability of mutating (p_m) an individual is lower than 0.1. Mutation consists in changing the value of a gene of an individual. In CGA, it means flipping the value from one to zero, or from zero to one. This operator helps the GA to tackle the problem of premature converge by trying new genes.

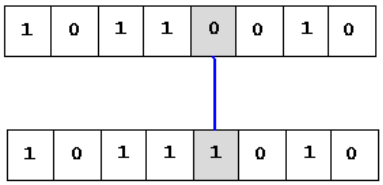

Figure 3.4: Mutation.

Both operators offer a trade-off between *exploitation* and *exploration* of the search space. *Exploitation* is due to cross over and it aims for testing new zones of regions where individuals of the current population belong. The act of jumping from one region to another is called *exploration*, and in GA, this role corresponds to the mutation operator. Hence, in GA jargon, people talk about the one-to-one correspondence between *exploitation* and cross-over, *exploration* and mutation. Consequently, tuning parameters of GA has to do with setting their exploitative and explorative power. In the literature of CGA, we find many standard parameter settings. Two of them are summarized in table 3.2:

Parameter	DeJong and Spears [41]	Grefenstette [42]
Population Size	50	30
Number of Generations	1000	-
p_c	0.6	0.9
p_m	0.001	0.01

Table 3.2: Two well-known GA parameter settings.

In most cases, GA with standard set of parameters perform satisfactory. Sometimes, due to the characteristics of the problem or the demand upon performance, manual tuning is necessary. Manual tuning is always a demanding task, because it involves carrying out a significant number of experiments in order to find a good set of parameters. This is one of the main disadvantages to the design of GA.

GA choose individuals for the next generation by means of a mechanism called **Selection**. Like the natural selection of species, the selection mechanism assigns a probability of survival to each individual proportional to its fitness, this strategy is also know as *stochastic sampling with replacement*. The most common selection strategies are:

Elitist A set of the best individuals of a generation are passed to the next generation. Normally, only the best individual is passed from generation to generation. There is also the possibility of replacing an individual - usually the worst or a random one- with the best individual found during the whole search process.

Proportional The selection is performed proportionally according to their fitness value, in this way, there is a higher probability, but not a certainty, that the best individuals will pass on to the next generation.

Rulet individuals are selected randomly according to the difference between their fitness and the fitness of their competitors.

The flow of CGA is shown in algorithm 2. The execution of GA is split into two phases: **selection** and **recombination**. The selection creates an intermediate population which is recombined and parents are replaced with their offspring in the next generation. Each of these cycles is called iteration. Lines two and three build the initial population, normally, by some random generation process. Lines four to 25 show the cycle of a generation in GA. Line five selects individuals for the next generation, line six to thirteen aim at the recombination mechanisms. Here, cross over takes place with a probability p_c, and parents are replaced with new individuals in the new population (line 11). After crossover (lines 14 to 21), GA apply mutation with a probability p_m. The new offspring replaces its parent in the new population (line 19). Lines 22 to 24 check if a stop condition is fulfilled. Line 26 returns the best individual.

Algorithm 2: Canonical Genetic Algorithm

```
1  begin
2      Generate initial population
3      Compute fitness of each individual
4      while not finished do
5          Select individuals for next generation
6          for population size / 2 do
7              if random() ≤ p_c then
8                  Select two individuals for reproduction
9                  Generate offspring by crossing over
10                 Compute fitness of the two offspring
11                 replace parents with offspring
12             end
13         end
14         for population size do
15             if random() ≤ p_m then
16                 Select an individual for mutation
17                 Generate offspring by mutating
18                 Compute fitness of the offspring
19                 Replace parent with offspring
20             end
21         end
22         if population has converged then
23             finished:=true
24         end
25     end
26     return BestFound;
27 end
```

3.2 The Schemata Theorem

The clear advantages of GA over other search techniques are that they are well founded and robust as well as have been tested on a wide range of problem areas.

Many research has concentrated on understanding the remarkable performance of GA. The most well-know explanation is called **The Schema Theorem** which is due to Holland [3]. According to Holland, a schema is a pattern of gene values, which represents some region of the space that contains ones or zeros at certain places. Schematas are represented by the alphabet {0,1,#}, where "#" can be anything. For example, the individual 01011111 is a member of regions 01####1# , 0##1###1, ##0##11#, etc.

The schemata theorem claims that opportunities of reproduction of an individual solely depend on its fitness function, which is computed according to its *phenotype*. Thus, schematas which give a better fitness pass through one generation to the other.

In this propagation strategy, cross-over suffers from two drawbacks that considerably slow the rate of evolution:

1. Good individuals are tested many times over generations, this means evaluating the goal function many times for the same individuals.

2. Offspring can move from one region to another, playing an explorative instead of an exploitative role.

The probability that offspring move from one region to the other mainly depends on the distance between ones and zeros that define the region. For instance, cutting at the second position is the only way that the individual 01###### could leave its region, but independently on the gene where the individual 0######1 is cut, the risk from leaving the region of its parents exists. Hence, block of adjacent ones and zeros, called *compact blocks*, are most likely to be propagated into next generations with a probability proportional to the fitness of individuals that carry them.

In natural genetics [43], *inversion* is the operation that sometimes rearranges genes so to bring offspring closer to their parents. Then, the new building blocks are more compact and less subject to being broken up by crossover [43]. If they specify a region of high average fitness, then the less compact blocks are automatically replaced by more compact ones, because they lose few offspring.

In GA, different individuals with several genes explore the search space in parallel, this *implicit parallelism* is one of the accepted explanations for the outstanding performance of GA and one of the advantages over other search techniques. Holland claimed that the optimal way of searching the space is assigning a number of reproductive trials to individuals proportional to their relative fitness. In this way, relatively better genes have more chances to be in next generations than relatively weaker genes.

Implicit parallelism allows to efficiently explore a large region of the search space using only few individuals, and the success of an individual is not only due to one gene, but rather to the block. This propagation strategy is also the major drawback, because evolution is inductive, that is, it does not aim at best individuals, it aims for running away from adverse circumstances. Therefore, the population can reach a steady state where all individuals are away from unfavourable circumstances, in the same way that GA are trapped in a suboptimal solution. In GA, it is said that when 95% of the population share the same value on the genes, the population **converge**, and premature convergence is due to the underlying assumption of the schemata theorem that GA handle an infinity population.

3.3 Conclusions

In this chapter, foundations and important features of the Genetic Algorithms were introduced. This chapter focused especially on the *Canonical Genetic Algorithm*, which was presented by John Holland in 1975.

This chapter also addressed the *schemata theorem*, which is the traditional explanation for the dazzling and outstanding performance of *Genetic algorithms* in many real-life problems.

Chapter 4

Web Question Answering: Baseline and Evaluation

Question Answering Systems (QAS) extract answers to natural language questions from raw text. This answering task consists essentially of two major steps: retrieving promising documents and determining which strings are most likely to be the answer. Without a shadow of doubt, the performance of both components have a significant impact on the overall performance of question answering systems.

In most cases, natural language questions represent some sort of close relations amongst entities, where the answer is the missing part of this relation: one or more entities, or the kind of relation that entities hold [29]. To illustrate this, consider the following relation between two sorts of entity:

Inventor *invented* ***Invention***

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

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

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****?*

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 query, and the overall performance of QAS has to do with their success in coping with different kinds of relations and entities.

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 may find uncountable variations that express the same concepts and ideas. For instance, entities can be written in many different forms (i.e. "*Alexander Graham Bell*", "*Graham Bell*", "*Bell*", "*Alexander G. Bell*","*The inventor of the telephone*", etc) and many words can signal at the same relation (i.e. "*invented*","*was developed*","*discovered*","*was created*", etc.). In deed, the linguistic phenomena presented in natural language corpus is much more complex than considering only possible variations of entities and relations. It involves reference resolution, complex syntactical and semantical relations, morphology, inference, world knowledge, etc. All things considered, the answer to a particular question can be easily extracted from one corpus, on the contrary, extracting the answer to the same question from another corpus can be an extremely difficult task. Furthermore, trying to clearly identify that there is no answer on the corpus can be even harder, which, this is not an unusual case. Consequently, QAS try to effectively deal with questions that aim at some sorts of entities and relations.

All in all, in order to attempt to properly assess the efficiency of QAS, three important issues must be taken into account: (a) metrics, (b) the contribution of the system, and (c) the complexity of the task. In the first issue, different metrics aim for evaluating different aspects of systems. For this reason, a metric must be chosen according to the aspect of interest: *speed*, *precision*, *recall*, *quality of the answer*, *etc.* In the second issue, the architecture of QAS consists merely of many specialized components that are combined in different ways in order to deal with questions aiming at different kinds of relations and entities. As a logical consequence, measuring accurately the contribution of each individual component is a laborious and hard task. In particular, if we consider that some QAS make stochastic decisions while they are searching for an answer. In the third issue, the performance of QAS is highly dependent on the complexity of the underlying linguistic phenomena in the corpus. On the one hand, sometimes purpose-built algorithms are not necessary to extract some answers from a given corpus, which can be extracted by means of general-purpose answer extractors. On the other hand, when general-purpose techniques are not enough to rank answer candidates, the use of more sophisticated strategies is necessary. Given the fact that we need to accurately assess the contributions of sophisticated systems, it can be concluded that a baseline is necessary.

In general, a **Baseline** is a traditional and well-known as well as effective strategy, which provides a construction line that helps to have a notion of the complexity of the task, in this way, it helps to discriminate the contribution of novel systems.

This chapter is organized as follows: section 4.1 describes a general overview of the Web question answering problem, section 4.2 brings up the topic of the design of a baseline, section 4.3, goes over the *Term Frequency - Inverse Document Frequency* metric, section 4.4, presents the proposed baseline, section 4.5, goes over an ad-hoc evaluation metric, and section 4.6, draws some conclusions.

4.1 Web Question Answering Problem

The Web Question Answering Problem can be viewed as the problem of searching for a set of strings that represent answers to a given natural language question Q. A string represents an answer to Q, when it is the missing member of the relation established by the query.

To begin with a formal description of the Web Question Answering Problem, consider S to be a set consisting 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* 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.*

Then, the **Question Answering Problem** consists in finding the *n-gram* that satisfies:

$$max \;\; K(B_{sk_1k_2}, Q) \tag{4.1}$$

Where K is a function which states how likely the *n-gram* represents an answer to Q. Essentially, 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 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:

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

Some *n-grams* extracted from S are: B_{112}=“*Igor Sikorsky*”, B_{223}=“*tea bag*”, B_{278}=“*Thomas Sullivan*”. Normally, $B_{sk_1k_2}$ can not contain a set of Q^* banned terms, which are usually from the query or it can not be in the stop list ϱ of the language:

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

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

The size of the search space 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:

$$NPA = \frac{N * \bar{\sigma} * \bar{\Upsilon} * (\bar{\Upsilon} - 1)}{2} \tag{4.2}$$

The factor $\frac{\bar{\Upsilon}(\bar{\Upsilon}-1)}{2}$ represents the number of possible *n-grams* of different length. This value results from an arithmetic series given by every sentence of length $\bar{\Upsilon}$ has one possible $\bar{\Upsilon}$*-gram*, two *(*$\bar{\Upsilon}$*-1)-grams*, until $\bar{\Upsilon}$ *uni-grams*. For simple values like N=10, σ=3, $\bar{\Upsilon}$=15, there are 3150 possible answers. Table 4.1 shows other different values for NPA.

N	10	30	50	100
NPA	3150	9450	15750	31500

Table 4.1: Number of possible answer candidates vs. number of snippets.

4.2 Discussion

Table 4.1 shows that the amount of strings increases proportionally to the number of sentences. It is perfectly clear that many of those strings hardly play the role of the missing part of the relation established by the query. Some of them are strings that a simple regular expression can easily filter, such as sequences of punctuation signs or math symbols. We can not then give credit for recognizing these strings to a sophisticated strategy where credit is not due. At the same time, the proportion of these strings is a primary measure of complexity of the task.

In each document exist words that clearly differentiate its content, these words are called indexes. The likelihood of a term as an index of a document gives the notion of its significance. On the one hand, if a term is not likely to be an index, it is not considered a truly representative of the document, because it does not discriminate its content, thus, a sophisticated strategy will be necessary to correctly identify its role or wether it is an answer or not. On the other hand, a general-purpose strategy could be necessary when the answer has a strong likelihood as an index. This is a plausible interpretation on grounds of: (a) if a term is likely to be an index of a document, it is highly probable that it provides the necessary localized context to readily distinguish the significance and the role of the word. (b) if a term is not likely to be an index, the document probably does not provide the necessary localized context in order to easily distinguish its significance and role. For instance: *rare terms* and *close class words*. In addition, each collection of documents has terms that serve as indexes, whose are used for differentiate one collection from the other. If a term is likely to be an index of a collection, it is highly possible that a subset of documents will provide enough localized context to unambiguously identify its role and significance or whether or not it is an answer. This localized context, implicitly determined by the query, consists principally of: paraphrasing, redundancy, relevant semantic context, uncovered syntactic relations, etc. In short, the idea is to exploit the likelihood of a term as an index of a collection in order to approximately estimate how difficult it is to sharply distinguish if it is the answer to a question or not.

In fact, it is crystal clear that it is a rough approximation, trying to determine precisely the complexity of extracting an answer from a collection is more complex than trying to determine exactly its likelihood as an index. The formulation of the query also contributes substantially to identify an answer readily on a corpus. It seems to be a reasonable approximation, when we only desire to properly understand the contribution of QAS or novel strategies.

In *Information Retrieval* (IR), indexing documents with respect to a collection involves three steps: computing the likelihood of each term as an index, removing *stop-words* and *stemming* [27]. In traditional IR, this likelihood is computed by means of a well-known technique named *Term Frequency - Inverse Document Frequency* (tdidf). *tdidf* provides an efficient and useful framework for ranking terms according to their power as an index of a document. As a logical consequence, the use of *tdidf* for building a baseline is encouraged.

Back to the point of assessing QAS, the next question that needs to be answered is: *What is a right answer?*. To begin with, we need to determine how likely it is that the answer provided by a question answering system fulfills the role of the missing part of the relation established by the query. For this purpose, a set of answer patterns usually comes along with the set of questions. These patterns represent variations of answers presented on the corpus. Thus, comparing answers with patterns is the most common way of assessing the likelihood of strings as answers. In second place, QAS can still find valid answer strings, which are not considered as answers in the set of patterns. This is an crucial issue when QAS based on the Web are assessed, because different variations of answers retrieved from the Web are not priori known. Four kinds of answer are considered:

1. **Exact Answers** match one of the patterns provided as answer patterns.
2. **Inexact Answers** do not match a given pattern, but they are orthographical variations of exact answers, or simply have a strong semantical relation to one exact answer, for instance, "*Where is the Gateway Arch*?", if the set of exact answers consists of:"*USA*", "*St. Louis*", some orthographical variations are:"*U.S.A.*","*ST. Louis*", and one semantical related answer is "*Jefferson National Expansion Memorial*". In this category, the addition of some context words is also considered, i.e.:"*in USA*".
3. **Correct Answers** are not considered in the set of possible answers, but they are related to one of the possible answers. But, this relation is not strong enough to unambiguously identify it as an inexact answer. Some correct answers for "*Where is Berlin?*" could be: "*Europe*","*Planet Earth*", "*near Postdam*", etc.
4. **Alternative Answers** are answers whose correctness depends on the user. For "*Who invented the radio?*", the Web provides two possible answers: "*Nikola Tesla*" and "*Guillermo Marconi*". This is usually due to different or conflicting opinions about the topic of the question.

In order to compare fairly with other systems, experiments must be carried out using exactly the same set of settings: *corpus*, *questions*, *answers*, *metrics*, *etc.* Exploiting intensively the Web as a target corpus makes this comparison even more difficult, because search engines do not assure that they will always provide the same set of snippets/documents for a particular request. Incidentally, the Web is changing and updating all the time.

Even more to the point, only few approaches describe the performance of their systems and components regarding different kinds of questions (see [14, 23, 29, 30]). In most cases, systems are compared considering only their overall performance on a target corpus-questions set. Comparing systems accurately is therefore not possible today, because an overall raw number does not provide enough precise information for being able to sharply distinguish which are the core and vital components of each system. Under these conditions, it is hard to readily distinguish where and how QAS can be improved.

Other Baselines[1] The problem of using search engines as a baseline is the fact that how they work is not well-known. Moreover, the content of the Web updates all the time and the supporting technology can suddenly change. Also, Search Engines take into account additional information while they are ranking documents such as cookies and favorite web-sites. Furthermore, not all search engines extract answers, most of them compute only small descriptions. Under these conditions, it is difficult to use them as a basis for analyzing systems or making an accurate comparison between different systems as well as measuring the contribution of novel strategies.

4.3 Term Frequency-Inverse Document Frequency

In *Information Retrieval*, the importance of a word within a document is measured according to two contexts: the source document and the collection of documents. The overall significance is usually combined as follows:

Source Document	Document Collection	Overall Weight
High	High	Very High
Low	High	Average
High	Low	Average
Low	Low	Very Low

Table 4.2: Term weighting schema.

Table 4.2 recalls **Zipf's law**. Zipf stated that the plot of the logarithm of the frequencies of a word in decreasing order is a straight line [28], and checked his hypothesis on the American Newspaper English [27]. Later, Luhn took advantage of

[1]Some unknown reviewers proposed the use of Google as a baseline. This paragraph aims for clarifying the reason why is not desirable to consider a search engine as a baseline.

Zipf's law for removing rare and non-significant words from documents, by setting two empirical thresholds. Luhn observed that the role of words for discriminating the content of a document reaches a peak somewhere between these two thresholds. In 1958, Luhn [25] stated that the frequency of words and their relative position within a sentence give a good notion of the significance of a sentence.

Nowadays, the ideas of Luhn are still used for measuring the importance of words and sentences in a collection of documents. Basically, the procedure consist of three steps: *stemming*, computing the significance and removing *close class words*. In the first step, 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: "*buy*","*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* [27], such as:"*neutralise*" and "*neutron*".

The **Term Frequency - Inverse Document Frequency** is a metric that tries to measure the significance of a word[2] w_i within a document d_j in the collection D:

$$tfidf(w_i, d_j) = \frac{freq(w_i, d_j)}{max_{\forall w_i \in W} freq(w_i, d_j)} * log\left(\frac{\mid D \mid}{nd(w_i, D)}\right) \tag{4.3}$$

Where $freq(w_i, d_j)$ is the frequency of the word w_i in the document d_j. Where $\mid D \mid$ is the size of the collection, W is the set of all terms that appear in the collection, and $nd(w_i, D)$ is the number of documents where w_i occurs. The first factor is a normalization of the raw frequency and makes the term frequency independent of the length of documents, and the second factor is the power of the term as an index of the collection [26].

w_i	$freq(w_i, d_j)$	$max_{\forall w_i \in W} freq(w_i, d_j)$	Weight
he	120	150	0.8
the	150	150	1
radio	35	150	0.233
invented	40	150	0.266
Marconi	15	150	0.1
Tesla	20	150	0.133

Table 4.3: Example of normalized term frequency in a document d_j.

Table 4.3 shows an example of the normalized term frequency for a set of six words on a document d_j. The most frequent word on d_j occurs 150 times. Table

[2]In IR, the question "*what is considered a word*?" has to do with the underlying language model. Usually, a word is a *unigram* [44]. But approaches that exploit further representations exist [45].

4.4, shows the *inverse document frequency* for the same set of words. The number of documents in the collection is 100.

w_i	$nd(w_i, D)$	$log\left(\frac{\mid D \mid}{nd(w_i,D)}\right)$	Weight
he	80	1.25	0.097
the	100	1	0
radio	50	2	0.3
invented	80	1.25	0.097
Marconi	28	3.57	0.55
Tesla	45	2.22	0.35

Table 4.4: Example of *Inverse Document Frequency* ($\mid D \mid = 100$).

Table 4.5, shows the final value for the *tdidf*. An overwhelming advantage of *tdidf* is that it is very fast to compute, because the *inverse document frequency* needs to be computed only once for the whole collection.

w_i	$tf(w_i, d_j)$	$idf(w_i)$	$tfidf(w_i, d_j)$
he	0.8	0.097	**0.0776**
the	1	0	**0**
radio	0.233	0.3	**0.07**
invented	0.266	0.097	**0.026**
Marconi	0.1	0.55	**0.055**
Tesla	0.133	0.35	**0.047**

Table 4.5: Example of *Term Frequency-Inverse Document Frequency*.

Lastly, *stop-words* or *close class words* are cut-off. **Stop-words** are highly frequent words, which do not carry any meaning. Some examples from an English *stop-list* are

do, the, will, sometimes, himself, everything, ...

Upon looking closer at table 4.5, one can easily notice that "*he*" is highly ranked, despite the fact it is a *stop-word*. Nowadays, implementations of *stop-lists* in many languages are available[3], which are used for natural language processing tasks.

[3] `htpp://www.unine.ch/info/clef/`

4.4 Baseline

The idea of a baseline[4] is to construct an algorithm which is a basis for constructing more sophisticated systems. With regards to the discussion in sections 4.2 and 4.3, our baseline is based largely on the t*erm frequency-inverse document frequency*. Each retrieved snippet is interpreted as one document, and the set of retrieved snippets as the collection of documents. Since we are dealing with web snippets, the formulae 4.3 is changed as follows:

$$tfidf(w_i) = \frac{freq(w_i)}{max_{\forall w_i \in W} freq(w_i)} * log\left(\frac{\mid D \mid}{nd(w_i, D)}\right) \tag{4.4}$$

Instead of counting the frequency on each document, the *idf* is weighted proportional to the normalized frequency of the word w_i in the whole collection. Essentially, due to the size of snippets, they are small pieces of text and the original normalized term frequency (equation 4.3) does not draw a distinction between words in each web snippet, all of them can be considered as an index of the document. However, we are not interested in the set of indexes of each document, but rather, aim for computing the likelihood of strings as indexes of the whole collection.

Algorithm 3: Baseline

```
   input  : C, stopList
1  begin
2  |  words = extractAllWords(C)
3  |  rank = tdidf(words)
4  |  rank = filterWords(rank,stopList)
5  |  rank = filterRareStrings(rank)
6  |  return rank;
7  end
```

The input of the baseline is the document collection C and a *stop-list*. Line 2 extracts all words[5] from the document. Line three looks for the most frequent word and computes the *tdidf* according to equation 4.4. Line four filters words that are in the given *stop-list*. The motivation of performing the filtering at this step is to consider *stop-words* while the baseline is computing the most frequent word. Line five filters strings that contains numerical and/or alphabetic characters. Here, all rare substrings are removed, leaving strings that are likely to be an answer. In this step, we remove strings that an answer extractor will easily identify as non answers. Eventually, line six returns the ranked answer candidates. Table 4.6 shows the final rank obtained by our baseline. In this rank, we consider only one document consisting of snippets glued together.

[4]Thanks to an unknown reviewer for providing some ideas for the design of this baseline.

[5]In the scope of this baseline, the underlying language model is a *unigram* model. Each *unigram* is a sequence of spaces separated by spaces or punctuation signs: coma, colon and semi-colon as well as dot.

w_i	$tf(w_i)$	$idf(w_i)$	$tfidf(w_i, d_j)$
radio	0.233	0.3	**0.07**
invented	0.266	0.097	**0.026**
Marconi	0.1	0.55	**0.055**
Tesla	0.133	0.35	**0.047**

Table 4.6: Ranking of strings - Baseline.

On the whole, the presented baseline ranks strings according to their likelihood as an index of the set of snippets. The baseline removes from the rank all strings consisting of *math symbols*, *html tags*, *special characters*, *etc.* The score of each string is computed by means of the *term frequency - inverse document frequency.*

4.5 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 [29]) of a system is the average value for a set Qu of Q_n questions:

$$MRR = \frac{1}{Q_n} . \sum_{\forall q \in Qu} \frac{1}{rank\_answer_q} \tag{4.5}$$

Let us consider the following results of a Question Answering System:

		Top one - Rank				
	Total	1	2	3	4	5
Questions	100	49	24	9	4	5
MRR	0.66	0.49	0.12	0.03	0.01	0.01

Table 4.7: Example of *Mean Reciprocal Rank.*

In the illustrative example, 91 out of 100 questions were answered. The MRR of the system is 0.66, and the contribution to the overall score of ranked answers: top is 0.49, while the top two is 0.12, the top three 0.03, the top four and the top five 0.01. It is important to remark that:

1. MRR measures the impact of an improvement on a system, but it does not provide an accurate comparison of two different systems. Because, a system with a lower MRR can deal with a set questions that another higher scored system can not deal with.

2. MRR does not describe the distribution of the rank of answers over the question set (i.e. *results per questions type*) . Hence, it is not possible to do a deep error analysis.

3. There is no difference in the score, if the system gives only wrong answers and no answer at all.

4.6 Conclusions

This chapter presents a discussion about the burning issues of metrics and evaluation of Question Answering Systems. This discussion focused attention on the choice of an evaluation metric and the design of a baseline for web-based Question Answering Systems. The selected evaluation metric is the *Mean Reciprocal Value*, and the baseline is founded on the *term frequency-inverse document frequency*.

Chapter 5

A Genetic Algorithm for Data-Driven QA on the Web

The increase of the amount of information on the Web has led search engines to deal with a huge amount of data as users have become retrievers of all sorts. Search engines are no longer focusing on retrieving relevant documents for a user's particular request, but are also providing other services (i.e., *Group Search*, *News Search*, *Glossary*). This has caused more complex user requests, a problem which is addressed by Question Answering (QA) systems. QA aims to answer natural language (NL) questions prompted by a user by searching the answer in a set of available documents on the Web. Question answering is a challenging task due to the ambiguity of the language and the complexity of the linguistic phenomena that can be found in NL documents [13].
Currently, the main kinds of questions to answer are those that look for name entities (i.e., *locations*, *persons*, *dates*, *organizations*). Nevertheless, QA systems are not restricted to these kinds of questions. They also try to deal with more complex questions that may require demanding reasoning tasks, as the system searches the answer.

The answering process usually starts analyzing the query [7,8] in order to determine the *Expected Answer Type* (EAT). The EAT allows the QA system to narrow the search space [9], while it is ranking documents, sentences or sequences of words in which the answer is supposed to be found. This set of likely answers is called answer candidates. In this last step, the QA system must decide which are the most suitable answers for the triggering query. Usual strategies for extracting and ranking answer candidates are based on frequency counting, pattern matching and detecting different orderings of the query words [7, 9, 10]. Answer extractors can also make use of *Machine Learning* techniques for learning contextual information from previously annotated question-answer pairs [12]. They also take advantage of linguistics tools or external knowledge sources [8, 13] such as *shallow parsers*, *stemmers*, *stop-lists*, *lexical databases* or deep processing using linguistic formalisms like HPSG [11]. However, it is still unclear how each technique contributes to deal with the linguistic phenomena that QA systems face while searching for the answer.

Recently, a number of web-based open-domain webQA have been developed based mainly on the redundancy of the Web [10,14,15]. Most of them try to answer factoid questions utilizing a tree-layers architecture:

1. conversion of NL questions to search engine specific queries.
2. interface to a public search engine for retrieving documents
3. extraction of answers from retrieved web pages

Some systems take advantage of the redundant information on the Web, in order to match the context of some answer candidates by looking at some paraphrases of the query [10]. But, it is not reasonable to consider all possible paraphrases of all possible queries, therefore research efforts have moved towards statistical methods, to take advantage of large amounts of available training data, which basically consists of pairs {question, answers} along with the textual context of the answer. In this way, systems learn the context of the possible answer candidates in a language independent and easy to re-train fashion [16]. This kind of approach uses different sorts of features extracted from the training data covering linguistic features from *n-grams* to parse trees. The question of selecting the learning features comes along with the question of how strong the dependency on the different kinds of questions is. Data-driven methods are probably enough for some questions like "*Who invented the telephone?*", whereas some knowledge base will be necessary to answer a question "*Why is it cloudy today?*". In this chapter, we concentrate on a data-driven approach, and like [16], we consider a two-layer web question answering system:

1. An *Information Retrieval* (IR) engine that retrieves a set of documents and/or sentences that may contain answers to a given question Q.
2. An answer extraction module that identifies and extracts a substring from a text snippet that is likely to be an answer to Q and assigns a score to it.

The goal is to develop a data-driven question answering system that: (a) uses primitive knowledge and poor knowledge processing components, and (b) automatically acquires the content and control information from previous {question, answer} pairs.

This chapter is organized as follows: section 5.1 presents an overview of our web question answering approach, section 5.2 describes our genetic algorithm for extracting answers, and section 5.3 draws some conclusions.

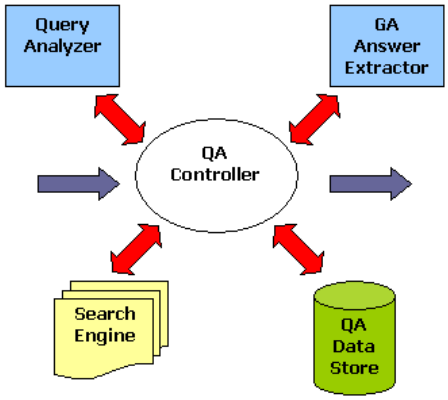

Figure 5.1: System Overview.

5.1 System Description

5.1.1 Overview

Figure 5.1 shows the major control flow of the data-driven web-based question answering system, called `GAQA`. When a natural language question Q triggers our system, the `QA-CONTROLLER` first sends it unmodified to the search engine[1] and retrieves the first N snippets.

Snippets are then normalized (algorithm 4) as follows: they are capitalized, and all HTML tags and accents (i.e. "ü", "ó", "Á", etc.) as well as rare and mathematical signs (i.e. "˜", "#", "+", "/", etc.) are removed.

In the next step, snippets are mapped to a set of sentences, by simply using the standard punctuation signs as splitting points: colon, semi-colon, dot and comma. The `QA-CONTROLLER` sends Q to the `QUESTION-ANALYZER`, which normalizes Q, in order to extract keywords from the query, which are strings delimited by spaces within Q. From now on, Q will refer to the normalized query.

[1] googleAPI: `http://www.google.com/apis/`

Algorithm 4: Normalization Algorithm

```
input : C
1  begin
2      normalizedSnippets = ∅
3      foreach c ∈ C do
4          ns = removeHTMLTags(c)
5          ns = removeMathSigns(ns)
6          ns = removeAccents(ns)
7          ns = removeRareSigns(ns)
8          ns = toUpperCase(ns)
9          normalizedSnippets = normalizedSnippets ∪ ns
10     end
11     return normalizedSnippets
12 end
```

The keywords serve as an index for the QA-CONTROLLER to extract the relevant {questions, sentences, answers} tuples from the QA-STORE. Later, the QA-CONTROLLER sends the tuples, the normalized query and snippets to the GA-ANSWER-EXTRACTOR. This component tries to extract word subsequences as answer candidates directly from snippet sentences. The underlying genetic algorithm uses a statistical model of the contextual cues determined from the selected tuple subset and terms from the normalized query Q to measure the fitness of answer candidates. It only uses a language-specific list of *stop-words* as an external resource [2]. No other external resources are used.

Next, the minor components of the GAQA will be described including: the QA-STORE, the QUESTION-ANALYZER and the QA-CONTROLLER. The GA-ANSWER-EXTRACTOR is described in detail in the next section.

The **QUESTION-ANALYZER** returns the language and the EAT of the question. This is performed by identifying a set of wh-keywords on the query:

[2]The stop-list from htpp://www.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils/stop_words is used. It contains 319 highly frequent close class forms,.

	Keywords
Date	Wann, When, Cuándo, Qué año, Cuando, What year, Welcher Tag, Que año, Que ano, What day, Welchem Jahr, Que ano, Welcher Jahr
Location	Where, Onde, Dnde, Donde, Wo, Woher, What city, In what city, What country, In what country, In what town, In what continent, In what planet, What town, What region, What continent, What planet, What area, In what area, In what region, In which city, In which country, In which town, In which continent, In which planet, In which region, In which area, Was ist die hauptstadt von
Person	Wer, Who, Quién, Quem, Quien, Whom, Wem, Wen, What is the name of, What man, What company, What enterprise, What woman, What person, What organization

Table 5.1: Some sample keywords.

The returned EAT is: LOCATION, DATE or PERSON, and the language: English, German, Spanish or Portuguese.

The **QA-STORE** consists of tuples {question, sentences, answers} obtained in previous `QA-CYCLE`. A `QA-CYCLE` is the complete process of answering a question. A example of a tuple is:

Who invented the Radio?	
Answer	Sentences
Nikola Tesla	Nikola Tesla invented the radio in... The radio was invented by Nikola Tesla
Guillermo Marconni	Guillermo Marconni invented the radio in... The radio was invented by Guillermo Marconni

Table 5.2: Sample `QA-STORE` tuple.

The tuples contained in the `QA-STORE` are the memory of results of past QA cycles.

The **QA-CONTROLLER** answers a question according to the flow in the following algorithm:

Algorithm 5: `QA-CONTROLLER`

input : $nlQuery$,N

```
1  begin
2      nQuery = normalizedQuery(nlQuery);
3      queryAnalisys(nQuery);
4      getStopList(language);
5      getSnippets(N,nlQuery);
6      getSentences(snippets,stopList);
7      getSelectedUnigram(selectedSentences,nQuery);
8      getQuestionAnswerPairs(selectedUnigram);
9      GA(selectedSentences,BDSentences,stopList,nQuery);
10     return BestFound;
11 end
```

The first step is to normalize snippets by removing all irrelevant symbols and capitalizing the string. The normalized string is sent in line three to the query analysis module which returns the language of the query, and then the appropiate *stop-list* is retrieved. Line five gets N snippets from the Web. Then, sentences that contain more than two *non-stop words* are selected. Line seven extracts a set of *uni-grams* from the set of selected sentences and the query. A selected *uni-gram* must occur at least two times in the set of sentences and not be in the *stop-list*. Later, the `QA-CONTROLLER` starts by replacing answers in every sentence -retrieved from the `QA-STORE`- with a special word w_0, in order to calculate the frequency of every word relative to w_0. Eventually, it sends the frequencies obtained in the previous steps and the selected sentences as well as the stop list to the `GA-ANSWER-EXTRACTOR`, which tries to find appropiate answer candidates.

5.2 The Genetic Algorithm for Extracting Answers

5.2.1 Acquiring the syntactic category of the EAT

Since the EAT and language of the new triggering question are computed by the `QUESTION-ANALYZER`, `GAQA` needs to find at least one substring in any sentence that has a similar syntactical behaviour than the EAT. For this reason, the `QA-CONTROLLER` extracts the {answer, sentences} pairs which aimed to the same EAT and language in previous `QA-CYCLES` from the `QA-STORE`. The answer with w_0 is then replaced in

every {answer, sentence} pair. Next, the following values are computed:

$$H_l(w_i, \epsilon) = freq(w_i, w_0, \epsilon) = \sum_{s=1}^{\sigma} X_{si(\tau-\epsilon-1)} X_{s0\tau} \tag{5.1}$$

H_l is the frequency which a word w_i is to the left of w_0 with ϵ word between them. $1 \leq i \leq \mid W \mid$, where W is the dictionary of all words in the snippets, and $\mid W \mid$ is the number of different terms in W. For the simplicity sake, it is assumed that the answer occurs only once in the sentence, and X is a binary variable which:

$$X_{sik} = \begin{cases} 1 & \text{if the word } w_i \text{ is in the sentence } S_s \text{ at position } k \\ 0 & \text{otherwise.} \end{cases}$$

Where $len(S_s) \geq k \geq 1$, S_s is the s-th sentence in σ, $1 \leq s \leq \sigma$, and τ is the position of w_0 in the sentence S_s, which can be computed by the following formula:

$$\tau(S_s) = \sum_{k=1}^{len(S_s)} k X_{s0k} \tag{5.2}$$

Table 5.3 shows the value of $\tau(S_s)$ for our working `QA-STORE` tuple. 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_s	$\tau(S_s)$
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 5.3: $\tau(S_s)$ for the illustrative `QA-STORE` tuple.

In a similar way, we compute the frequencies of w_i to the right of w_0 with ϵ word between them:

$$H_r(w_i, \epsilon) = freq(w_0, w_i, \epsilon) = \sum_{s=1}^{\sigma} X_{s0\tau} X_{si(\tau+\epsilon+1)} \tag{5.3}$$

In the illustrative example, the values for H_l and H_r are:

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

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

Table 5.4: H_l and H_r for the `QA-STORE` tuple.

Eventually, we compute the probabilities P_r and P_l respectively:

$$P_l(w_i, \epsilon) = \frac{H_l(w_i, \epsilon)}{freq(w_i)} \tag{5.4}$$

$$P_r(w_i, \epsilon) = \frac{H_r(w_i, \epsilon)}{freq(w_i)} \tag{5.5}$$

The following tables show the values for P_l and P_r 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 5.5: P_l and P_r for the `QA-STORE` tuple.

Where frequencies $freq(w_i)$ are given by:

	invented	the	radio	in	was	by
$freq(w_i)$	4	4	4	2	2	2

Table 5.6: $freq(w_i)$ for the `QA-STORE` tuple .

5.2.2 The Genetic Algorithm

The GA is described in algorithm 6. 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_x (the crossover probability). P_l and P_l regard the syntactic model which measures the syntactic contribution of each aligned word to the syntactic fitness of the answer candidate. The algorithm is split into two major steps: Lines 2-4 shows the initialization of the population and lines 5-11, the evolution.

Algorithm 6: Core Genetic Algorithm

```
   input : T,S,P_l,P_r,I,P_m,P_x,ϱ
 1 begin
 2     t=0;BestFound = ∅;
 3     AC[0] = createInitialPopulation(I,S,ϱ);
 4     EvaluatePopulation();
 5     while (t < T) do
 6         t++
 7         CAC=Crossover(AC[t-1],P_x)
 8         MAC=Mutate(AC[t-1],P_m)
 9         EvaluatePopulation(AC[t],P_l,P_r)
10         AC[t]=selectPopulation(CAC,MAC,AC[t-1])
11     end
12     return BestFound
13 end
```

The components are described as follows:

- **Coding:** The *chromosome* is the tuple $\{K(B_{sk_1k_2}),s,\ k_1,k_2\}$, where the *genes* represent the fitness, the sentence and the boundaries of the *n-gram* in the sentence. In figure 5.2, the value 1.76 represents the fitness of the individual, 2 is the sentence index, and the last two numbers are the position of the boundary words of the answer candidate.

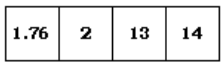

Figure 5.2: `GA-QA` Chromosome.

- **Create Initial Solution:** (Line three) A random sentence S is chose and therefore, we choose two random cutting points: 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*.

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

$$\left(\sum_{k=1}^{s(B^*,S_s)-1} \alpha(w_{sk}, Q) P_l(w_{sk}, k-1) + \sum_{k=e(B^*,S_s)+1}^{len(S_s)} \alpha(w_{sk}, Q) P_r(w_{sk}, k - e(B^*, S_s) - 1) \right) \tag{5.6}$$

- **Evaluate Population:** Line nine computes the "utility" of an individual. The idea is to construct a smooth and regular fitness function, so that chromosomes with reasonable fitness are close in the space to the ones with slightly better fitness. The fitness K of an answer candidate $B_{sk_1k_2}$ is given by the formula 5.6. When w_{sk} is the word at position k in the sentence S_s and $\alpha(w_{sk})$ is a special weight for every w_{sk} which is also in the query Q. The function K assigns high fitness to answer candidates that show a highly similar syntactic behavior with answers in the `QA-STORE`, with which they also share common query terms in the context. The function K particularly favours query terms by giving them a weight of $\alpha(w_{sk})$. In this way, we smooth the goal function in order to choose strings near query terms. In equation 5.6, $B^* = B_{sk_1k_2}$ and s and e are functions that return the start and the end positions of B^* respectively.

  Consider that the next question is sent to the search engine: "*Who invented the airship?*". For simplicity, let us assume we retrieve only one sentence: "*The airship was invented by Ferdinard von Zeppelin.*". When the string B^* = "*Ferdinard von Zeppelin*" is evaluated, we obtain two sentences of the `QA-STORE` that provide with alignment for this string (see table 5.2). Then, we have:

The	radio	was	invented	by	Nikola Tesla
The	radio	was	invented	by	Guillermo Marconni
The	airship	was	invented	by	Ferdinard von Zeppelin

Table 5.7: Sample of alignment.

The frequencies of "*the*" and "*invented*" is four, the frequencies of "*was*" and "*by*" is two (see table 5.6), then $K(B^*, Q)$ is given by:

$$K(B^*, Q) = 2 * 0.5 + 1 * 1 + 2 * 0.5 + 1 * 1 = 4$$

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

- **Select Population:** Line nine selects by means of a *proportional mechanism*. This mechanism selects individuals proportionally according to their fitness value. The best individuals amongst the populations at t and $t-1$ and individuals generated by the recombination mechanisms will have a higher probability of surviving in the next generation. Note that individuals which contain banned terms or belongs to the *stop-list* are not considered in the next generation.

- **Mutate:** Line eight randomly changes 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 $0 \leq k_2 - k_1 < len(S_s)$.

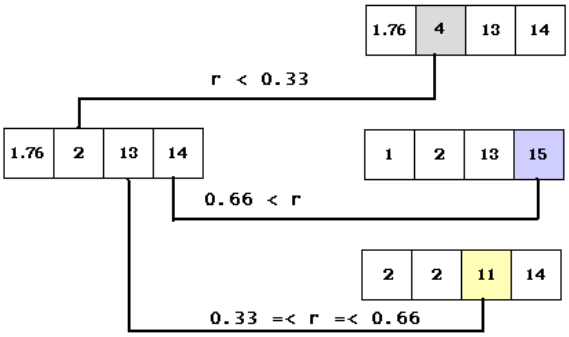

Figure 5.3: GA-QA Mutation operator.

- **Cross Over:** Two selected individuals

$$\{K(B^1_{s_1 k_{11} k_{12}}), s_1, k_{11}, k_{12}\}$$

$$\{K(B^2_{s_2 k_{21} k_{22}}), s_2, k_{21}, k_{22}\}$$

exchange their genes by computing the next values:

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

Where α_1 and α_3 are the left boundaries of the parents, and α_2 and α_4 are the right boundaries. Every time parents are crossed over, the operator must check if the right boundary is greater or equal to the new left boundary and if it is within the limits of the sentence.

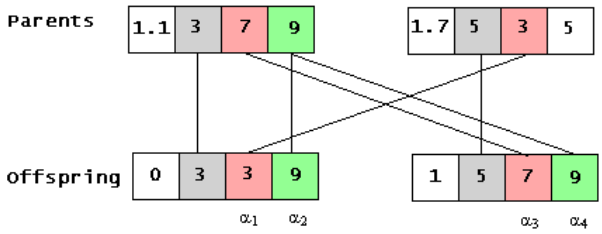

Figure 5.4: GA-QA Cross Over operator.

The following offspring are generating by exchanging their *phenotype*:

$$\{K(B^{1'}_{s_1\alpha_1\alpha_2}), s_1, \alpha_1, \alpha_2\}$$
$$\{K(B^{2'}_{s_2\alpha_3\alpha_4}), s_2, \alpha_3, \alpha_4\}$$

Normally, parameters of GA must be tuned. In GA-QA, p_m and p_c are set to one and the selection method is performed at the end of each iteration. The selection step is the only responsible step for choosing the individuals that will survive, because tuning the GA is an expensive task and parameters are normally computed for each instance of a problem. In this case, it should be done for at least for each type of question. This allows the best individuals to still pass from one generation to the other, but a larger population is taken into consideration. Individuals representing *stop-words* or query terms are not desirable, but due to the nature of the proposed recombination mechanisms, there is a high probability that offspring belong to a *stop-list* or to the set of terms of the query. Hence, selecting individuals from the three sets allows GA to have a larger set from which it can choose desirable individuals for the next generation.

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 simple query keyword matching ranking in: (a) GA 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 *expected answer type*. (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 patterns must be aligned with the context of each occurrence of an answer candidate. Thus, the particular importance of ignoring *stop-words*.

5.3 Conclusions

This chapter presents a web-based Question Answering System, which looks for answers to natural language questions on web snippets of text. This system is capable of learning the syntactic behavior of the *expected answer type* from previous annotated question-answer pairs, and takes advantage of a *genetic algorithm* in order to search for the most promising answer candidates on the retrieved web snippets.

Chapter 6

GA for Answer-Sentence Syntactic Alignment

Machine Translation aims for translating natural language texts from one language to another. Statistical Machine Translation (SMT) is the research field of Machine Translation which takes advantage of data in order to roughly translate texts. This data usually consists of a pair of sentences that are translations of one other [46]. In Machine Translation, statistical approaches are preferred to linguistic based approaches, because they can be easily re-trained to deal with several pair of languages.

Because of the rapid and significant increase in the use of multilingual information sources such as the *internet*, the user demand for machine translation engines, which can deal with several pair of languages, has rocketed up. For this reason, many companies are strongly motivated to develop easy re-trainable systems capable of coping with different pairs of languages. In particular, the translation systems of IBM focus on translating raw texts from English to French by taking advantage of translations from proceedings of the Canadian Parliament (see full details in [47]). IBM builds their systems in an upward spiral, that is, each new system tackles deficiencies of the previous one. In all these systems, sentences in both languages are aligned in a *word-by-word* fashion. By and large, a *word-by-word alignment* associates each word in the original sentence to a set of words in the target sentence. The next figure sketches an example of *word-by-word alignment*:

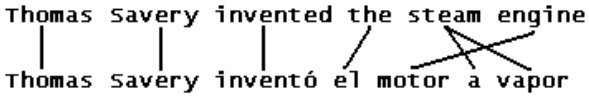

Figure 6.1: Machine Translation word-by-word sentence alignment.

In the illustrative example, the first sentence is in English: "*Thomas Savery invented the steam engine*", while the second is in Spanish: "*Thomas Savery inventó el motor a vapor*". Drawing attention to the complete set of features that a Machine Translation System considers while it is aligning sentences word by word, is beyond of the scope of this work (see full details in [46–48]). For the purpose of this work, the attention is focused on two important features of these models: *fertility* and *distortion*. The *fertility* of a word is the number of words that it generates in

the target sentence, and *distortion* has to do with the distribution of the position of words within the translated sentence, which is not necessarily uniform. In the example, the word "*steam*" generates two words "*a vapor*" and the order of the relation *adjective-noun* is inverted.

Question Answering Systems (QAS) aim for extracting answers to natural language questions from raw text. Some systems take advantage of the redundant information on the Web, in order to match the context of some answer candidates by looking at some paraphrases of the query [7, 9, 10]. 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. Therefore, research efforts have move towards statistical methods which take advantage of context information extracted from previously annotated tuples {question, sentence, answer} in order to match the context of new tuples [12] and extract answers afterwards. In the previous chapter, this contextual information is based largely on the inferred syntactic behaviour of answers presented on tuples which aimed at the same EAT and language. This inference is grounded on some observed distributional patterns of behaviour across words respecting the EAT. Accordingly, answer candidates were successfully found out by properly aligning words in new sentences with these distributional patterns. Nevertheless, the aligning success depends heavily on a strong similarity between patterns presented on new sentences and the training data.

Within the scope of this work observed distributional patterns of words respecting the EAT are interpreted as distributional patterns of words with respect to their translated sentences. A possible paraphrase is therefore seen as a possible translation. Consequently, a likely answer candidate to the EAT is interpreted as a likely translation to the model. Accordingly, *distortion* is seen as the capability of a paraphrase for swapping its constituents (i.e. *prepositional phrases*, *subject*, *etc*), and *fertility* is interpreted as the capability of words for occurring along with their modifiers, which do not contribute to unambiguously identify an answer candidate within the paraphrase. Broadly speaking, *fertility* aims for discriminating words that do not carry any substantial meaning for discriminating the answer.

This chapter, in brief, presents a genetic algorithm for lessening the dependance on the similarity between patterns presented on new sentences and the training data. Specifically, the effects of *fertility* and *distortion* on the alignment of words respecting the answer candidate.

This chapter is organized as follows: section 6.1 discusses the syntactic answer-sentence alignment, section 6.2 presents a formal description of the problem and its complexity, section 6.3 describes the genetic algorithm for the syntactic answer-sentence alignment, and section 6.4 draws some conclusions.

6.1 Discussion

First, consider that our training data consists of the following {question, answer, sentence} tuples:

Who invented the Loudspeaker?	
Answer	Sentences
Chester W. Rice	Chester W. Rice invented the Loudspeaker. The Loudspeaker was invented by Chester W. Rice
Edward W. Kellogg	Edward W. Kellogg invented the Loudspeaker. The Loudspeaker was invented by Edward W. Kellogg

Table 6.1: Training data tuples.

In addition, consider the following new {sentence,answer} pair:

{*The geiger was really invented in 1913 by Hans Geiger*, *Hans Geiger*}

Next, aligning tuples in a *word-by-word* fashion respecting the answer looks like as follows:

			The The	Loudspeaker Loudspeaker	was was	invented invented	by by	Chester W. Rice Edward W. Kellogg
The	geiger	was	really	invented	in	1913	by	Hans Geiger

Table 6.2: Sample of alignment.

The only match is due to the preposition "*by*". Consequently, the syntactic fitness[1] of the answer candidate "*Hans Geiger*" is:

$$K(\text{“}Hans\,Geiger\text{”}, Q) = 1 * 1 = 1$$

Looking at table 6.2, it is crystal clear that words like "*really*" or prepositional phrases such as "*in 1913*" do not contribute significantly to enhance the fitness of the answer candidate "*Hans Geiger*" and seriously distort the alignment. In general, two phenomena can have noticeable impact on the alignment:

1. **Fertility** Some words are more likely than others to occur with its modifiers. Two examples are *noun-adjective* and *verb-adverb* combinations. As a consequence, it is not certain if they are likely to occur along with their modifiers in the training data or not. In the example, "*really*" does not occur along with "*invented*" in training tuples.

[1] This function is fully described in section 5.2. Q is the query "*Who invented the Geiger?*".

2. **Distortion** Some constituents like *Propositional Phrases* (PP) are more likely than others to occur in different orders within sentences or by being arbitrarily inserted. For instance: "*in 1913 by Hans Geiger*", "*by Hans Geiger in 1913*", "*a man called*", "*a man named*", *etc.*

As a result of mitigating the effects of *fertility* and *distortion*, the alignment is:

The	Loudspeaker	was	invented	by	Chester W. Rice
The	Loudspeaker	was	invented	by	Edward W. Kellogg
The	geiger	was	invented	by	Hans Geiger

Table 6.3: Sample of alignment.

Hence, the syntactic fitness of the new aligned sentence is given by:

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

To sum up, when the syntactic fitness of an answer candidate is assessed, the effects of *fertility* and *distortion* on the paraphrase must be taken into account. It is good to highlight that removing not necessary words leads to increase the syntactic fitness, because the alignment depends crucially on the distributional patterns presented on training tuples. Thus, many combinations must be tested in order to determinate exactly the best syntactic fitness for an answer candidate.

6.2 Answer-Sentence Syntactic Alignment Problem

Let us assume that we have a tuple: {sentence, answer}, in which the answer is a substring of the sentence. Consider S to be the sentence where the answer is already replaced with a special string w_0, and $\gamma = len(S)$ is the number of words in S. For example, $S =$"*The First Helicopter was invented in Kyiv by* w_0 *in 1909*", where the answer candidate is given by $w_0 =$"*Igor Sikorsky*" and $\gamma = 11$. Consider also a function $\tau(S)$ which returns the position of w_0 in the sentence S, and two other functions $P_l(w_i, \epsilon)$ and $P_r(w_i, \epsilon)$, which returns the likelihood that a word w_i occurs ϵ words to the left and to the right of w_0 respectively[2]. In the working example, $\tau(S) = 9$ and P_r and P_l are given by some external model ($1 \leq i \leq \gamma$).

Consider that words from S are removed in such a way that the syntactic fitness of the answer candidate is maximized, that is, that the remaining words maximize the likelihood of the answer candidate to the EAT. For the purpose of keeping track of words that remain on the aligned sentence S', consider the next binary variable:

$$Y_i = \begin{cases} 1 & \text{if the word } w_i \text{ is in the aligned sentence } S' \\ 0 & \text{otherwise.} \end{cases}$$

[2]These functions are described in details in section 5.2.1.

For instance, an aligned sentence $S^{'}$ is "*The * Helicopter was invented * * by* w_0 *in 1909*"[3], where $Y_2 = Y_6 = Y_7 = 0$ and $Y_1 = Y_3 = Y_4 = Y_5 = Y_8 = Y_{10} = Y_{11} = 1$.

The **number of remaining words** (NRW) between the word w_i and w_0 is defined as follows:

$$NRW(i) = \begin{cases} \sum_{k=i+1}^{\tau(S)-1} Y_k & \text{if } i < \tau(S). \\ 0 & \text{if } i = 0. \\ \sum_{k=\tau(S)+1}^{i-1} Y_k & \text{if } i > \tau(S). \end{cases}$$

In our working example, NRW(11)=NRW(5)=1. Since the goal is to find an alignment that maximizes the syntactic fitness of the answer candidate with respect to its context, the new fitness function takes into account the set of values for Y_i as follows:

$$K(S,Q) =$$

$$\sum_{i=1}^{\tau(S)-1} \alpha(w_{sk},Q) Y_i P_l(w_i, NRW(i) + \delta_l) + \sum_{i=\tau(S)+1}^{\gamma} \alpha(w_{sk},Q) Y_i P_r(w_i, NRW(i) + \delta_r) \tag{6.1}$$

This function K particularly equally favours query terms by giving them a weight of $\alpha(w_{sk})$. δ_l and δ_r are the left and right offset respectively. An offset is a translation of context terms. For instance, consider the following sentence $S^{'}$ to be an alignment of S: "*The * Helicopter was invented * * by + + +* w_0 *+ + in 1909*". In $S^{'}$, the offset are marked with a "+", thus, $\delta_l = 3$ and $\delta_r = 2$.

Sometimes constituents or words are inserted next to the answer candidate in sentences on the training data, which can distort the alignment. Offsets attempt to tackle this problem head-on. In the instructive example, $S^{'}$ can align a sentence in the training tuples, such as "*The Helicopter was invented by a man named* w_0 *in Kyiv in 1909*".

In short, removing words aims for improving the alignment when a smaller number of words within new sentences are desired, and offset a larger number of words. It is extremely clear that not all possible alignments are considered, but the number of possible alignments exponentially increases as long as the number of words in the sentence also increases.

The number of possible alignments Consider $\tau(S)$ to be the position where the answer candidate w_0 occurs. Then, $L_l = \tau(S) - 1$ is the number of words to the left of the answer candidate, and $L_r = \gamma - \tau(S)$ is the number of words to the right. All possible combinations of L_l words are given by $L_l!$. At this point, 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 $L_r!$. Incidentally, every

[3]The removed words are signaled by means of a star.

combination of words to the left can occur simultaneously along with any combination of words to the right.

Then, the total number of **Possible Alignments** (PA) is given by:

$$PA = L_l! * L_r!$$

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

$$PA = \sum_{j=0}^{L_l} j! * \sum_{j=0}^{L_r} j!$$

For our working example, the number of possible word alignments is 184936:

$$PA = \sum_{j=0}^{8} j! * \sum_{j=0}^{2} j! = 46234 * 4 = 184936$$

If word orderings are deliberately restricted to combinations that preserve their relative order. The number of possible combinations is:

$$PA = \sum_{j=0}^{L_l} \left(\begin{array}{c} L_l \\ j \end{array} \right) * \sum_{j=0}^{L_r} \left(\begin{array}{c} L_r \\ j \end{array} \right)$$

In the example, the number of possible alignments is 1024:

$$PA = \sum_{j=0}^{8} \left(\begin{array}{c} 8 \\ j \end{array} \right) * \sum_{j=0}^{2} \left(\begin{array}{c} 2 \\ j \end{array} \right) = 256 * 4 = 1024$$

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 value for the word next to w_0 is one. In the case of the left context, the number of new possible combinations is:

$$\Delta_l * \sum_{j=0}^{L_l - 1} \left(\begin{array}{c} L_l - 1 \\ j \end{array} \right)$$

Similarly, the number of new possible combinations due to the right context is:

$$\Delta_r * \sum_{j=0}^{L_r - 1} \left(\begin{array}{c} L_r - 1 \\ j \end{array} \right)$$

Eventually, the number of possible alignments is defined as follows:

$$PA = \left(\sum_{j=0}^{L_l} \binom{L_l}{j} + \Delta_l * \sum_{j=0}^{L_l-1} \binom{L_l-1}{j}\right) * \left(\sum_{j=0}^{L_r} \binom{L_r}{j} + \Delta_r * \sum_{j=0}^{L_r-1} \binom{L_r-1}{j}\right)$$

where Δ_l and Δ_r are upper bounds for their respective offset. Regarding the working example, if values for $\Delta_l = 5$ and $\Delta_r = 5$ are considered, then the number of possible combinations is 17024:

$$PA = \left(\sum_{j=0}^{8} \binom{8}{j} + 5 * \sum_{j=0}^{7} \binom{7}{j}\right) * \left(\sum_{j=0}^{2} \binom{2}{j} + 5 * \sum_{j=0}^{1} \binom{1}{j}\right)$$

$$= (256 + 5 * 128) * (4 + 5 * 2) = 896 * 14 = 12544$$

This result considers only one tuple {question, sentence, answer} consisting of ten 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 section 4.2. A reasonable estimate of the total number of **Possible Alignments** for a set of σ different sentences is:

$$PA = \frac{\sigma * \bar{\Upsilon} * (\bar{\Upsilon} - 1)}{2} * \left(\sum_{j=0}^{L_l} \binom{L_l}{j} + \Delta_l * \sum_{j=0}^{L_l-1} \binom{L_l-1}{j}\right) * \left(\sum_{j=0}^{L_r} \binom{L_r}{j} + \Delta_r * \sum_{j=0}^{L_r-1} \binom{L_r-1}{j}\right)$$

$\bar{\Upsilon}$ is the average number of words on a sentence. Thus, the factor $\frac{\bar{\Upsilon}(\bar{\Upsilon}-1)}{2}$ represents the number of possible answer candidates of different length on a sentence. In our illustrative example,

$$PA = \frac{25 * 14 * (14 - 1)}{2} * 12544 = 28537600$$

Then, the number of possible alignments for a set of 25 sentences is 28537600 ($\bar{\Upsilon} = 14$). Consequently, an efficient search algorithm is necessary to early detect and test promising alignments.

6.3 The GA for Answer-Sentence Syntactic Alignment

The GA for Answer-Sentence Syntactic Alignment, called `GA-ASSA`, is described in algorithm 7. The input for this algorithm are T the number of iterations, the tuple $\{S, w_0\}$ needs to be syntactically aligned, I the size of the population, P_m the probability of mutation and P_x the crossover probability. P_l and P_l regard the syntactic model which measures the syntactic contribution of each aligned word to the

syntactic fitness of the answer candidate. This algorithm is split into two major steps. Lines 2-5 shows the initialization of the population while lines 6-12 shows the evolution.

Algorithm 7: GA for Syntactic Answer-Sentence Alignment

input : T,S,P_l,P_r,I,P_m,P_x,w_0

```
1  begin
2      t=0;BestFound = ∅;
3      S=replace(S,w0);
4      AC[0] = createInitialPopulation(I,S);
5      EvaluatePopulation(AC[0],Pl,Pr,S);
6      while (t < T) do
7          t++
8          AC[t]=selectPopulation(AC[t-1])
9          CAC=Crossover(AC[t],Px)
10         MAC=Mutate(AC[t],Pm)
11         evaluatePopulation(AC[t],Pl,Pr,S,w0);
12     end
13     return BestFound
14 end
```

Line two initializes the variables and line three replaces the answer candidate string with w_0. Lines three and four create and evaluate the initial population of the `GA-ASSA`. Line eight picks individuals of the next generation which are crossed over afterwards (line nine). Mutation takes place next (line ten). Line eleven evaluates the new population, and line thirteen returns the best individual found during the whole search. The components are described as follows:

- **Coding:** The *chromosome* 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. Figure 6.2 shows a *chromosome*:

1	0	1	1	1	0	0	1	0	2	1	0

Figure 6.2: `GA-ASSA` Chromosome.

In this individual, "*0*" and "*2*" are offsets for the left and right contexts respectively. The *chromosome* in the figure represents the sentence:"*The Helicopter was invented by* w_0 *+ + in*".

- **Create Initial Solution:**(line four) For each word in the sentence S, a random number is computed. 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.

- **Mutate:** Line ten 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.

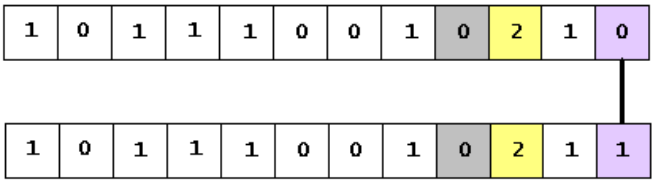

Figure 6.3: `GA-ASSA` Mutation.

- **Cross-Over:**(line nine) Two selected individuals are cut 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.

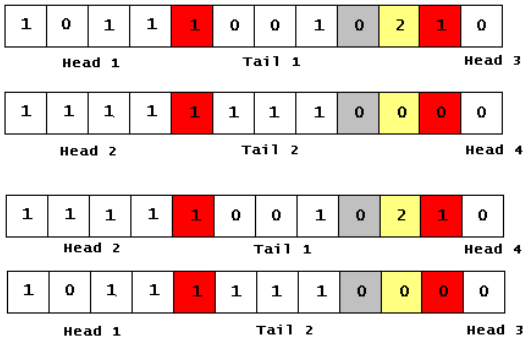

Figure 6.4: `GA-ASSA` Cross Over.

- **Evaluate Population:** Line eleven computes the "utility" of an individual according to equation 6.1. The weight $\alpha(w_i)$ has a value of two for all query terms, otherwise is one. P_l and P_r are parameters for the `GA-ASSA`, which are computed by means of the procedure described in section 5.2.1.

- **Select Population:** Line eight selects the best individuals of the previous generation according to the *proportional mechanism*.

6.4 Conclusions

This chapter deals at a greater length with the alignment of contextual patterns of new answer candidates with patterns of the *expected answer type* seen on previously annotated data. This chapter focus attention on the effects of different kinds of paraphrases on the alignment, specially, paraphrases consisting mainly of the addition and removal of meaningless -respect to the query- words .

In this chapter, the complexity of the answer-sentence syntactic alignment is discussed, and it presents a *genetic algorithm*, which aims for finding the best matching between the context of an answer candidate and the context of the *expected answer type*.

Chapter 7

PreGA: A Predicate-arguments and Data-driven Genetic Algorithm

Entities are connected to each other by means of relations. In some fresh and recent approaches [29], a question is seen as a relation amongst a set of entities, in which the answer is the missing part of this relation: at least one entity or the relation amongst them. In order to deal separately with different sorts of relations and entities, Question Answering Systems (QAS) take advantage of different strategies, which are usually combined.

In this thesis, a data-driven approach for extracting answers from the Web was presented. In this approach, the syntactic behaviour of the answer is learnt directly from the context of annotated {question, sentence, answer} tuples and used for guiding the search towards promising strings afterwards. In this search strategy, answers to new questions are tentatively identified by a *genetic algorithm* (GA), which properly aligns distributional patterns on annotated tuples with distributional patterns presented on new answer candidates. 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 match the previously annotated context, the answer will not be identified, even though it can be readily distinguished by means of some linguistic pattern.

Although, data-driven answer extractors get many inexact answers, they are 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 on linguistic processing, which extract exact answers, they demand more computational resources and are dependent on the language. All things considered, QAS are greatly encouraged to take advantage of both approaches, while they are attempting to cope with new questions prompted by users.

Given the fact that GA subsequently discover promising contextual 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. In this framework, the underlying assumption and primary motivation are the design of a specialized *goal function* and/or clever *reproduction mechanisms*, which aim for finely balancing the contribution of these two different approaches to the answering process. As a logical consequence, the quality and performance of QAS presented in this work, are improved.

This chapter is organized as follows: section 7.1 discusses the contribution of predicate analysis to the answer extraction process, section 7.2 presents the *predicate and data-driven genetic algorithm*, section 7.3 draws some conclusions.

7.1 Discussion

In this work, answer candidates are discovered by properly aligning their context with contextual syntactic patterns of the *expected answer type* (EAT), which are extracted directly from previously annotated {question, sentence, answer} tuples. Due to the nature of this extraction process, which is based mainly on syntactical patterns, a natural way of enriching this alignment is by adding semantic knowledge. There are two possible ways of directly enhancing this extraction process by the addition of semantic knowledge: (a) corpus driven semantics like LSA [49], (b) some linguistic out–of–the–box tools for semantic processing.

For the purpose of this work, the use of the latter is preferred to the former. Since GA already take advantage of data in order to logically deduce some syntactical patterns, the use of external semantic knowledge is clearly encouraged, causing the robustness and performance of the system to be vastly improved. Accordingly, the high dependency on the 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 [50]. A final reason is the existence of tools like Montylingua[1], 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 "*Henri Gif-*

[1] `http://web.media.mit.edu/~hugo/montylingua/`

fard invented the airship in 1852 in France.". MontyLingua[2] provides the following predicate-arguments representation for this sentence:

invent(Henri Giffard,airship,in 1852,in France)

In this example, it is crystal clear that the answer matches the subject, which can be clearly differentiated by its syntactic behaviour. This semantic representation along with the syntax behaviour of the *expected answer type* can help GA to sharply distinguish the answer on the text and significantly decrease the absolute dependence on patterns seen in the training data.

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 *expected answer type*, and some other arguments and/or the predicate in both predicate-arguments representations match. To illustrate this, consider the following query: "*Who invented the airship*?". The predicate-arguments representation provided by MontyLingua is as follows:

invent(Who,airship)

Comparing both predicate-arguments representations, it becomes perfectly clear that the subject "*Who*" matches the subject "*Henri Giffard*". 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.

7.2 The predicate-arguments and data-driven genetic algorithm

The `PreGA`[3] is fully described in algorithm[4] 8. The input for this algorithm are T the number of iterations, S the set of sentences, I the size of the population, P_m the probability of mutation and P_x the crossover probability. P_l and P_l regard syntactic models which measure the syntactic contribution of each aligned word to the fitness of an individual. ϱ is the *stop-list* of the language. This algorithm is split into two major steps: Lines 2-4 show the initialization of the population and lines 5-11, the evolution.

[2]In MontyLingua, the first argument is the subject.
[3]Many components of this *genetic algorithm* are already presented in this work.
[4]The structure of this genetic algorithm is similar to the genetic algorithm in chapter 5.

PreGA uses a similar *chromosome* representation to GAQA. The flow of the PreGA is as follows: line two initializes variables. Lines three and four create and evaluate the initial population of the PreGA, this creation is performed as in GAQA. Line seven randomly picks individuals of the next generation according to their fitness, which are crossed over afterwards (line eight). The selection process is similar to GA-ASSA, whereas cross-over is done as in GA-QA. Then, mutation takes place (line nine). Line ten evaluates the new population. Eventually, line twelve returns the best individual found during the whole search.

Algorithm 8: Predicate and Data-driven Genetic Algorithm.

input : T,S,P_l,P_r,I,P_m,P_x,ϱ

```
1  begin
2      t=0;BestFound = ∅;
3      AC[0] = createInitialPopulation(I,S);
4      EvaluatePopulation();
5      while (t < T) do
6          t++
7          AC[t]=selectPopulation(AC[t-1])
8          CAC=Crossover(AC[t],P_x)
9          MAC=Mutate(AC[t],P_m)
10         EvaluatePopulation(AC[t],P_l,P_r)
11     end
12     return BestFound
13 end
```

The novel components of PreGA are as follows:

- **Mutate:** (line nine) A random sentence is chosen 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.

- **Evaluate Population:**(Line ten) The fitness function of an answer candidate or individual B^* respecting to the query Q is given by:

$$K(B^*, Q) = K_d(B^*, Q) * K_l(B^*, Q) \tag{7.1}$$

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

[5] $K_d(B^*, Q)$ is computed as $K(B^*, Q)$ in section 5.2.2.

$$K_l(B^*, Q) = \sum_{\forall S_s \in S : B^* \in S_s} \gamma(S_s, Q) * \left(1 + \eta(S_s, Q) \sum_{o \in obj(S_s)} J(B^*, o)\right) \left(1 + \eta(B^*, S_s) \sum_{q \in obj(Q)} \sum_{o \in obj(S_s)} J(q, o)\right) \tag{7.2}$$

$\gamma(S_s, Q)$ 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:

$$\gamma(S_s, Q) = \begin{cases} 1 & \text{verb}(S_s)\text{=verb(Q);} \\ 0 & \text{otherwise.} \end{cases}$$

$\gamma(S_s, Q)$ 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:

$$\eta(S_s, Q) = \begin{cases} 1 & \text{subject}(S_s)\text{=subject(Q);} \\ 0 & \text{otherwise.} \end{cases}$$

$\eta(B^*, S_s)$ 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:

$$J(B_1, B_2) = \frac{\mid B_1 \cap B_2 \mid}{\mid B_1 \cup B_2 \mid}$$

The *Jaccard* measure 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="Giffard" and B_2="Henri Giffard":

$$J(\textit{Giffard}, \textit{Henri Giffard}) = \frac{1}{2} = 0.5$$

It is absolutely 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: "*Henri Giffard*

invented the airship in 1852 in France.", thus, the value for K_l is computed as follows:

K_l("*Henri Giffard*", "*Who invented the airship*?") = 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 ($K_d(B^*, Q) > 0$ in equation 7.1), consequently, the application of linguistic processing is carefully balanced and the strong dependency of the alignment on the annotated data significantly decreases. Nevertheless, `PreGA` still remains heavy dependent on patterns seen on the training data, while it is selecting a new answer string ($K_d(B^*, Q) > 0$ in equation 7.1).

7.3 Conclusions

This chapter introduces a *genetic algorithm*, which finely balances a data oriented approach and linguistic processing. This algorithm takes advantage of distributional contextual patterns of previously annotated tuples for guiding the search towards promising answer candidates. Once it detects these promising contexts, it performs a predicate analysis in order to establish the semantical bonding with the context of the query. This causes, the alignment to be semantically enriched, at the same time, it greatly reduces its strong dependency on the annotated data.

Chapter 8

Experiments

Through this work, questions are seen as a relation amongst entities, where the missing member of this relation is the answer. Hence, the quality of general-purpose Question Answering Systems has to do with how well they deal effectively with several kinds of question and entity. This work clearly presents three strategies for extracting answers to natural language questions from web snippets, and this chapter focuses special attention on the evaluation of these three presented strategies.

For the purpose of the general assessment of these methods, experiments were carried out considering carefully the three most common sorts of entity and a set of seven different relations. Accordingly, this chapter fully describes data-sets and parameters used for this evaluation as well as presents a discussion at a greater length of the obtained results.

8.1 Experimental Settings

Seven data-sets regarding different kinds of relation were particulary used in order to assess our strategies. Each data-set was formally split into two subsets: *training* and *testing*. For the simplicity sake, these subsets are referred as sets. The former was directly sent to our *webQA*[1] in order to retrieve tuples {question, sentence, answer}, which were used specifically for inferring syntactical distributional patterns afterwards (see section 5.2.1). The latter was separately sent to each individual method: `GA-QA`, `GA-QA`+`GA-ASSA` and `PreGA`. Here, it is good to pointedly remark three essential aspects of our strategies: (a) `GA-QA` is considered as described in chapter five, (b) `GA-QA`+`GA-ASSA` is a two-staged *genetic algorithm*, in which the first stage consists exclusively of a `GA-QA`, in the second stage, each obtained pair {sentence, answer candidate} is fully aligned by `GA-ASSA`, and (c) in the third method, the core *genetic algorithm* of `GA-QA` is simply replaced with `PreGA` according to the description in chapter seven. This partial replacement consists principally of a new mutation operator and a purpose-built goal function, which take advantage of *pred-*

[1] In the scope of this work, *webQA* is an external general-purpose Question Answering System, which aims for extracting answers from the Web. This system is used solely for retrieving training tuples.

icate analysis in order to enrich the syntactical data-driven alignment of `GA-QA`. In short, `GA-QA+GA-ASSA` is a data-drive improvement to `GA-QA`, whereas `PreGA` a sematic linguistic enhancement to `GA-QA`.

The detailed description of each data-set is as follows:

CLEF-2004 refers to answers from 1994/1995 newspapers articles. The 169 who-questions were directly sent to our *webQA* System in order to extract tuples {question, answer, sentence} from the Web. 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. The CLEF corpus considers rigorously questions on several topics and wrong answers, even tough they are few, they can have a considerable impact on the model. In deed, this manual annotation is a demanding task.

Inventions is a set of pairs {invention, inventor} provided by the Britannica Encyclopedia[2]. All inventions which theirs inventors are unknown were completely and permanently removed, and the list was split into two sets afterwards: *training* and *testing*. The first -alphabetical order- 87 inventions where used solely for training the model and the last 185 only for testing. In order to extract training tuples {sentence, answer} from the Web, each training pair {question, invention} was individually sent to our *webQA* by taking advantage of the following template:

$$\textit{who invented the } \{\textit{invention}\}?$$

In order to try to avoid a manual annotation at all costs, right and wrong pairs {answer, sentence} were used to train our methods. In this way, due principally to the redundancy and localized topic, the robustness of the system is also adequately tested. The testing set was individually sent to our strategies using the same template.

Presidents is a set consisting exclusively of 120 pairs {country, president}, which can be found on the on-line version of Wikipedia[3]. Accordingly, the first 33 tuples were used only during the training phase and the remaining 87 for the particular purpose of testing. The template is given by:

$$\textit{who is the President of } \{\textit{country}\}?$$

Like the set of inventions, right and wrong tuples were considered as training pairs {sentence, answer}. The testing set was also sent to our methods by means of the corresponding template.

[2] `http://corporate.britannica.com/press/inventions.html`
[3] `http://en.wikipedia.org/wiki/List_of_State_leaders.`

Symphonies is a set of pairs {composer, symphony} extracted from Wikipedia[4]. This set consists entirely of 180 symphonies, which were formally split into 80 tuples for training and 100 for testing. The procedure is similar to the previous set and the corresponding template is as follows:

who composed the {symphony}?

As the two previous sets, right and wrong tuples were taken into account as training pairs {sentence, answer}. Each testing tuple was individually sent to each of our methods by taking advantage of the corresponding template.

Prime Ministers is a set of pairs {country, prime minister} extracted from the on-line version of Wikipedia[5]. This set consists exclusively of 103 tuples, in which the first 29 pairs were used specifically for training and the remaining 74 solely for testing. The template is given by:

who is the Prime Minister of {country}?

Like previous sets, right and wrong tuples were particulary used as training pairs {sentence, answer}. The testing set was directly sent to our strategies using the corresponding template.

Locations is a set consisting entirely of 120 tuples {city, country} and {monument, city, country}, available on "Glass Steel and Stone, The Global Architecture Encyclopedia" [6], in which the first 37 pairs were used especially in the training phase and the last 73 only during the testing. The template look likes as follows:

where is the {monument/city}?

As previous sets, right and wrong tuples were considered as training pairs {sentence, answer}. Each testing tuple was separately sent to our methods by means of the corresponding template.

Dates is a set of pairs {person, birthday} available on "Famous Birthdays"[7]. This set consists only of over 4000 tuples, from which the first 2160 pairs where used specifically for training and the next 145 for the particular purpose of testing. The template is as follows:

when was {person} born?

[4] `http://en.wikipedia.org/wiki/List_of_symphonies_by_name.`
[5] `http://en.wikipedia.org/wiki/List_of_State_leaders.`
[6] `http://www.glasssteelandstone.com`
[7] `http://www.famousbirthdays.com/bday_123.html`

Like previous sets, right and wrong tuples were taken into account as training pairs {sentence, answer}. The testing set was individually sent to our strategies by taking advantage of the corresponding template.

It is good to merely highlight three fundamental aspects of the experiments. First, the `Baseline` was used exactly as fully described in chapter four, this means without additional modifications. Secondly, data-sets were deliberately selected in order to properly test three sorts of entity: PERSON and LOCATION as well as DATE. In addition, these selected data-sets focus essentially on several who-typed questions in order to empirically test different kinds of relation. Thirdly, the `Baseline` has to tune any parameter, whereas our strategies based largely on *genetic algorithms* necessarily need to tune their parameters:

`GA-QA` 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. The reader should consider that as the time goes by stronger individuals gradually take over the populations, so the number of different strings is considerable lower. In practice, about 50 different individuals are really tested during the whole search, hence, these values are rightly interpreted as a state in which the population finally converges. Similarly, P_m and P_x were set to one on the ground of the explanation in section 5.2.

`GA-ASSA` makes allowance for typical parameter settings, where mutation usually ranges over 0.001 to 0.1 and cross-over normally from 0.8 to 0.95:

Parameter	Value
Population Size	10
Number of Generations	30
p_c	0.8
p_m	0.1

Table 8.1: Parameters for `GA-ASSA`.

Since `GA-ASSA` runs every time that `GA-QA` finds a pair {sentence, answer candidate}, the population size and the number of generations must be low values. Despite of considering low values (10 and 30 respectively), the answering time is grossly unreasonable (see section 8.2). As well as that, explorative values for the probability of mutation and cross-over were specially selected in order to empirically and directly test a larger number of different alignments.

`PreGA` consists primarily on modifications to `GA-QA`, it takes then advantage of the same configuration.

It is well-known that *genetic algorithms* belong to the class of random search algorithms, they do not therefore absolutely guarantee to return always the same rank after finalizing the evolution of the population. But, they achieve a stable output after a number of runs proportional to the size of the search space. Parameters of `GA-ASSA` were mainly manually set by inspecting their stability after several

runs. The selected parameter setting seems to be utterly reasonable, if the size of the search space is carefully considered (see sections 4.1 and 6.2). However, this configuration could not be particulary useful for another kinds of question. A fined tuning of parameters would be adequate for each instance of a *genetic algorithm*, this means for each question -or at least each type of question- and its corresponding set of snippets. Without a shadow of doubt, high tuning is not a desirable option.

The *stop-list* is the only additional resource that our methods use. The language dependency is due only to our Query Analysis tool which is not used in the learning process or during the answer extraction phase, it is used solely to retrieve the right set of pairs {sentence, answer} from our *webQA* System, which attempts to answer a wider variety of questions in different languages, there is for this reason a complete independence of our model on the language or type of answer.

8.2 Results

The next table shows an overview of the obtained results:

Strategy	MRR	Total	1	2	3	4	5	AA
`Baseline`	0.376	413	137	92	78	42	51	13
`GA-QA`	0.497	401	242	78	38	31	12	14
`GA-QA+GA-ASSA`	**0.512**	437	240	97	38	35	13	14
`PreGA`	0.373	277	155	101	33	23	10	22

Table 8.2: 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). In 89 cases, no answer was manually found on the best 30 retrieved snippets. 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 localized 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. The reader can look at some results upon tables A.1 to A.15 and quickly realize that 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 (table A.1).

It worths 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 symphonies, "*Schubert*" is more likely than "*Franz Schubert*" or "*Franz*" (table A.8). In general, it is well-know that statistical oriented approaches often extract these kinds of inexact answer (see also [21]).

`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 crystal clear, `GA-QA` and `GA-QA+GA-ASSA` outperforms `PreGA`. Furthermore, by considering only results upon table 8.2, it can be concluded that the flexible alignment of `GA-QA+GA-ASSA` performs slightly better than `GA-QA` 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, specialized document retrieval, answer selection and validation. Under these concrete facts, results obtained by `GA-QA` and `GA-QA+GA-ASSA` (0.497 and 0.512 respectively) seem to be positively encouraging.

The following tables show results regarding each data-set:

Corpus	Questions	NAS	`Baseline`	`GA-QA`	`GA-QA+GA-ASSA`	`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**	0.203
Symphonies	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 8.3: MRR overview.

In the following tables, NAS stands for the number of questions in which there was no answer within the retrieved snippets (a total of 89 cases). `GA-QA+GA-ASSA` 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*. Due to the heterogene-

ity of the corpus, there is not enough contextual information/evidence for `GA-ASSA` (on snippets and in the training set) to enrich considerably the alignment. `GA-QA` obtained for this reason the best performance for the CLEF-2004 set of questions. Consequently, the amount of redundancy is a decisive factor in our strategies. Also, 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. A direct comparison between the performance of the `Baseline` and `PreGA` gives also a general and simplistic notion of the quality of the proposed `Baseline`.

The approach presented on [29] 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 [10], they obtained a MRR value of 0.45 for 500 TREC-9 questions. In [20], 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 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. Regarding the set of locations, the approaches presented on [53, 54] scored an MRR value about 0.8 for a similar data-set and the Web as a target corpus. These approaches take advantage of external lexical resources. `GA-QA` and `GA-QA+GA-ASSA` scored 0.638 and 0.684 respectively for a subset of the same set of questions. But, our methods are independent on a lexical database of locations.

In [52], 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 `GA-QA` and `GA-ASSA` 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 our *webQA* could not answer, and our *webQA* is based mainly on the strategy presented

on [52], these values are for this reason evidently not comparable. But, it is perfect clear that our strategies successfully answered questions that our *webQA* did not.

Another key issue relating to the evaluation of Question Answering Systems, extensively discussed in chapter four, 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 in the following table:

	`Baseline`	`GA-QA`	`GA-QA+GA-ASSA`	`PreGA`
`Baseline`	1	0.2899	0.2050	0.1878
`GA-QA`	0.2899	1	**0.640**	0.380
`GA-QA+GA-ASSA`	0.2050	**0.640**	1	0.3038
`PreGA`	0.1878	0.380	0.3038	1

Table 8.4: Average Correlation Coefficient between each pair of strategies.

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 `GA-QA` and `GA-QA+GA-ASSA`. This similarity can also be thoroughly inspected on tables A.1 to A.15. The other pairs seem not to be enough correlated to draw a valid conclusion, we can say therefore that they seem to contribute to the answering task in a different way. This last conclusion is also supported by their individual similarity to the `Baseline` (see tables A.1 to A.15).

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
Symphonies	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 8.5: 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 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 the role of dates 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
Symphonies	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 8.6: Results obtained by the `GA-QA`.

The table above shows results for `GA-QA`. Best results were obtained for the data-sets regarding Presidents, Prime Ministers and Locations. `GA-QA` outperformed the `Baseline` on every data-set. Given the lower MRR scores achieved by `GA-QA` 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 can be seen as a sharper difference between the contexts of the training and testing sets. Since results are quite similar, it can be interpreted broadly that the contextual information on the who-typed question of the CLEF corpus is less heterogenous than the corpus concerning dates. On the one hand, dates regard people from different contexts: *job*, *work*, *and much more*. On the other hand, the amount of different paraphrases for expressing the birthday of a person [59] is smaller than for expressing answers on the general-purpose who-typed questions of the CLEF corpus. Then, nothing else can be firmly stated.

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
Symphonies	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 8.7: Results obtained by the `GA-QA+GA-ASSA`.

The table above shows results for the two-staged `GA-QA+GA-ASSA`. Apart from the CLEF data-set, all results are better than the presented `Baseline`. For three data-sets `GA-QA` outperforms `GA-QA+GA-ASSA`: CLEF, Inventions and Symphonies.

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.

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	0.203	0	22	6	4	0	0
Symphonies	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 8.8: 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 not prefect clear if MontiLingua can deal efficiently with all possible paraphrases presented on snippets. A manual inspection leads to believe that it does not. Secondly, ungrammatical snippets seriously distort the *predication analysis*. On the whole, `PreGA` can not fully analyze sentences with intentional breaks, `PreGA` can not therefore easily discover right answer strings to some kinds of question on retrieved snippets.

Other key issue in experiments is to define whether an answer is correct or not. In fact, alternative answers to the one provided by the corpus exist. For instance, in case of inventions, which their inventors are nor clear such as "*The Radio*", the corpus provides "*Guglielmo Marconi*", but we can easily find on the Web: "*Nicola Tesla invented the Radio*". In the CLEF corpus this ambiguity is more often, due to the fact that some answers are out of date and the retrieved snippets contain updated information. Every time the system extracted a right answer, but it is not in the set provided by the corpus, it was labelled as AA (Alternative Answer). These answers were not taken into account in the final MRR score. 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, for instance, in the case of the inventor of the oil, we consider the following strings as answers: "*Edwin Laurentine Drake*", "*Edwin Drake*", "*Drake*" and "*Edwin*". Right answers do not make allowance for orthographical variations in order to reduce the ambiguity of the evaluation. The hyphens were removed. With regards to questions aiming at a DATE, correct answers were considered: the year, the month and year or the full date. For instance, "*1977*", "*AUGUST 1977*" or "*31 AUGUST 1977*". Regarding the data-set of locations, correct answers were the country or the city.

For example, "*Where is the Mozart statue?*", strings considered as right answers were "*SALZBURG*" and "*AUSTRIA*" (table A.13).

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 *genetic algorithms* can start looking for answers in another region of the answer space where they will not find the answer. In addition to that, some answers are missed because of intentionally inserted breaks, which seriously distort the alignment. This is also a reason of the better performance of `GA-QA+GA-ASSA`.

But, it is essentially important to duly note that this approach has its limitations. When we try to answer questions that aims a LOCATION or a DATE, the system can not readily distinguish the *expected answer type* in the syntactic alignment. This is due to the fact that the syntactic behaviour of dates and locations in English -as in another languages- is similar, both are indeed locations one in time and the other in the space. Another thing is that exists a larger amount of variations for expressing the same date than the same location (see example in section 1.1). These 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, at least a lexicon or string pattern matching is necessary.

8.3 Time Performance

On the one hand, results obtained by systems are a crucial issue when their quality is assessed. On the other hand, the amount of resources that systems require in order to achieve a certain quality level is also a critical issue to take into consideration when systems are evaluated. The following table shows the average execution time for each strategy vs. data-set (all questions):

Corpus	`Baseline`	`GAQA`	`GA-QA+GA-ASSA`	`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
Symphonies	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 8.9: Average execution time for for each strategy vs. data-set (milliseconds).

All strategies demanded more time for answering questions which aimed for a DATE, because the syntactical distributional model of the EAT took into account a larger amount of training data. In general, `GA-QA+GA-ASSA` takes more time in answering questions, on the other hand, they rank correct answers higher. Results suggest that the deep alignment of answer-context 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, due to the amount of training data or the complexity of the alignment. Although, this alignment is performed by robust heuristics which do not aim for discovering the best context-answer alignment, but rather one closer to the best. The next table shows a ratio respecting the `Baseline` of each strategy and data-set:

Corpus	`Baseline`	`GAQA`	`GA-QA+GA-ASSA`	`PreGA`
CLEF-2004	1	80.44	1851.53	326.58
Inventions	1	195.06	3770.02	349.07
Presidents	1	193.74	3764.41	366.78
Prime Ministers	1	247.77	4804.46	439.14
Symphonies	1	36.28	841.82	498.82
Locations	1	21.68	497.62	414.55
Dates	1	1248.85	10261.67	1550.1354

Table 8.10: Time ratio between the different strategies and the baseline.

This ratio gives a valid comparison independent on other tasks running at the same time on the servers. These results suggest that we can simply replace the deep data-motivated alignment with some ad-hoc linguistic processing and get better results than using only data-driven approaches, for instance: the data-sets of inventions and symphonies. On the other hand, if this linguistic support is not adequate, data-driven approaches tends to perform better. Overall, data-driven approaches obtain a good performance, but they need a huge amount of different contextual paraphrases in order to sharply identify the answer from its context.

8.4 Conclusions

This chapter discusses at length the assessment of the strategies introduced through this work. In general, two major conclusions can be drawn: (a) results show that data-driven approaches are a useful strategy for Question Answering Systems, and (b) the balance between data-driven approaches and ad-hoc linguistic processing is a vital issue relating to the potential improvement of the answer extraction phase.

By and large, it can be concluded that linguistic processing is still a necessary tool for domain-specific Question Answering Systems, and *Genetic Algorithms* seem to be a proper tool for dealing with the complexity of the alignment problem presented in this work. At this point, the reader should remember that our methods do not use a lexicon for identifying words boundaries.

Chapter 9

Conclusions and Further Work

This work presents a data-driven approach to question answering which takes advantage of syntactical distributional patterns for discovering answers to new natural language questions on the Web. These syntactical distributional patterns are directly learnt from the relative position of words with respect to the *expected answer type* on previously annotated pairs {sentence, answer}. These patterns are aligned with sentences presented on retrieved snippets in order to extract answers candidates to new questions. This alignment is performed by a purpose-built *Genetic Algorithm* (GA).

Experimental results suggest that the presented methods can cope with specific questions, specially with those questions whose answers are inserted into contexts, for which do not exist a large amount of morpho-syntactical variations. By means of a representative training set of pairs {context, answer}, they can readily identify answers. Results also suggest that at some point, if we want our strategies to achieve a high accuracy, they need to use more computational resources in order to learn a massive number of different patterns and align them with new sentences. As a logical consequence, the use of ad-hoc linguistic processing is still strongly encouraged.

On the one hand, by using this ad-hoc processing, systems lose their property of being easily re-trainable and language independent. On the other hand, they substantially increase their accuracy, when they aim at specific questions. With all these things in mind, our methods seem to be adequate for language independent domain specific systems however, where they can be trained with a reasonable amount of several paraphrases, and take advantage of ad-hoc linguistic processing.

The advantages of using GA for extracting answers are: (a) they mainly test good individuals while they are searching for the answer, (b) during this search, they find syntactical clues which are good indicators for balancing the linguistic processing. Looking closer to the coding of our GA, the reader can easily realize that they implicitly rank sentences extracted from the snippets, because the second genotype of the chromosome represents the sentence number. According to the schemata theorem, the phenotype of the gene should converge to its best - the one who adapts best to the environment- as the iterations go by. It is therefore a reliable indicator for applying purpose-built linguistic processing, like *predication* at

the sentence level. Here, another advantage of GA was exploited. They provided a framework (recombination mechanisms and goal function) for inherently implementing data-driven and linguistic motivated answer extraction strategies.

The main drawback to GA is that they do not absolutely guarantee to test the best individual. This drawback can be mitigated by increasing the number of iterations, which means using more computational resources. Due to the characteristics of the search space, setting the parameters of our methods are not a relevant issue. Even though, explorative parameters were deliberately selected for our GA, the population quickly converged. More to the point of the search space, `GA-ASSA` shows that taking advantage of *Genetic Algorithms* for data-driven question answering is a promising research field.

Another important conclusion has to do with the proposed model for acquiring syntactical distributional patterns of the *expected answer type* from raw text. On the one hand, it provides an adequate framework for designing a goal function, which can properly and subsequently discover answers. On the other hand, the main drawback to this model is that different expected answer types behave in a similar way. For this reason, the use of external knowledge is highly motivated. Consequently, the language portability of our approach as an open domain system is still an open question. Another thing is the fact that our methods were trained with wrong annotated tuples. It can then be concluded that our model is robust to noisy training data (when systems aim for answering specific questions) and the need of annotations can be substantially reduced.

To sum it up, the performance of data-driven methods has not only to do with the amount of data, it also has to do also with diversity of paraphrases presented on the training set. In this way, systems can take advantage of the redundancy presented in several information sources in a more proper way.

As a further work, we can propose the use of this alignment function for anaphora resolution in the snippets. It is clear that snippets returns many paraphrases of the same sentences extracted from independent documents. A syntactic alignment can be used, in order to discover some pronouns that can clearly be inferred from another snippet. This can bring about an increment of the precision of the answer extraction process of a Question Answering engine or the retrieval precision of a document retrieval system.

Bibliography

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

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

[3] Holland, J.,H. *Adaptation in Natural and Artificial Systems*, MIT Press, 1975.

[4] Beasley, D., Bull, D., R., Martin, R., R. *An overview of genetic algorithms: Part 1, fundamentals*, University Computing, 15(2):58-69, 1993.

[5] Beasley, D., Bull, D., R., Martin, R., R. *An overview of genetic algorithms: Part 2, Research Topics*, University Computing, 15(4):170-181, 1993.

[6] Coello, C. *Recent trends in Evolutionary Nultiobjective Optimization*, Evolutionary Multiobjective Optimization: Theoretical Advances And Applications, pp. 7–32, Springer-Verlag, London, 2005.

[7] Dumais, S., Banko, M., Brill, E., Lin, J., Ng, A. *Data-Intensive question answering*, In proceedings of the tenth Text REtrieval Conference (TREC 2001), November 2001, Gaithersburg, Maryland.

[8] Rijke, M., Monz, C. *Tequesta: The University of Amsterdam's Textual Question Answering System*, NIST Special Publication SP, 2002.

[9] De Chalendar, G., Dalmas, T., Elkateb-Gara, F., Ferret, O., Grau, B., Hurault-Planet, M., Illouz, G., Monceaux, L., Robba I., Vilnat A. *The question answering system QALC at LIMSI: experiments in using Web and WordNet*, NIST Special Publication SP, 2003.

[10] Dumais, S., Banko, M., Brill, E., Lin, J., Ng, A. *Web question answering: is more always better?*, Proceedings of SIGIR-2002, 2002.

[11] Keselj, V. *Question Answering using Unification-based Grammar*, In advances in Artificial Intelligence, AI 2001, volume LNAI 2056 of Lecture Notes in Computer Science, Springer, Ottawa, Canada, June, 2001.

[12] Lita, L., V., Carbonell, J. *Instance-based question answering: a data driven approach*, In Proceedings of EMNLP, 2004.

[13] Chen, J., Ge, H., Wu, Y., Jiang, S. *Question Answering Combining Multiple Evidences*, In Proceedings of TREC, 2004.

[14] Clarke, C., Cormack, G., V., Lynam, T., R., Li, C., M., McLean, G., L. *Web reinforced question answering (multitest experiments for trec 2001)*, In Proceedings of TREC, 2005.

[15] Ramakrishnan, D., Paranjape, D., Chakrabarti, S., Bhattacharyya, P. *Is question answering an acquired skill?*, In WWW, 2004.

[16] Echihabi, A., Marcu, D. *A Noisy-Channel Approach to Question Answering*, In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, July 2003, pp. 16–23, 2003.

[17] Salton, G. *"The SMART retrieval system". Experiments in Automatic Document Processing, Prentice Hall*, Englewood Cliffm, NJ, 1971.

[18] Shawe-Taylor, J., Cristianini, N. *"Kernel Methods for Pattern Analysis"*. Cambridge University Press, 2004: 143–153, 2004.

[19] Deerwester, S., C., Dumais, S., T., Landauer, T., K., Furnas, G., W., Harshman, R., A. *Indexing by Latent Semantic Analysis*, Journal of the American Society of Information Science, volume 41, number 6, 1990, pp. 391–407

[20] Charles, L., Gordon C., Cormack, V., Lynam, R. *Exploiting Redundancy in Question Answering*, Journal of the American Society of Information Science, volume 41, number 6, 990, pp. 391–407

[21] Echihabi, A., Hermjakob, U., Hovy, E., Marcu, D., Melz, E., Ravichadran, D. *How to select an answer String?*, Advances in Textual Question Answering, Kluwer, 2004

[22] Rinaldi, F. , Dowdall, F., Kaljurand, K., Hess, M., Mollá, D. *Exploiting paraphrases in a Question Answering System*, In Proceedings of the second international workshop on Paraphrasing, Volume 16.

[23] Moldovan, D., Harabagui, S., Clark, C., Bowden, M., Lehmann, J., Williams, J. *Experiments and Analysis of LCC's two QA Systems over TREC 2004*, TREC 2004, 2004.

[24] Mollá, D., Schneider, G., Schwitter, R., Hess, M. *Answer Extraction using a Dependency Grammar in ExtrAns*, Traitement Automatique de Langues (T.A.L.), Special Issue on Dependency Grammar, 41(1):127-156.

[25] Luhn, H., P. *The automatic creation of literature abstracts*, IBM Journal of Research and Development, 2, pp. 159–165, 1958.

[26] Robertson, S. *Understanding Inverse Document Frequency: On theoretical arguments for IDF*, Journal of Documentation, volume 60, number 5, 2004.

[27] van Rijsbergen, C., J. *Information Retreival*, Butterworths, 1979.

[28] Zipf, H., P. *Human behaviour and the principle of the least effort*, Addison-Wesley, Cambridge, Massachusetts, 1949.

[29] Lita, L., Carbonell, J. *Unsupervised Question Answering Data Acquisition From Local Corpora*, In Proceedings of the Thirteenth Conference on Information and Knowledge Management (CIKM 2004), Washington, DC, USA, November 8-13, 2004.

[30] Monz, C. *From Document Retrieval to Question Answering*, IILC Dissertation Series DS-2003-4, *Institute for Logic, Language and Computation*, University of Amsterdam, 2003.

[31] Savary, A., Jacquemin, C. *Reducing Information Variation in Text*, ELSNET Summer School, pp. 145–181, 2000.

[32] Nyberg, E., Frederking, R., Mitamura, T., Bilotti, M., Hannan, K., Hiyakumoto, L., Ko, J., Lin, F., Lita, L., Pedro, L., Schlaikjer, A. *JAVELIN I and II at the TREC 2005*, In Proceedings of the Fourteenth Text REtrieval Conference (TREC 2005), 2005.

[33] Chu-Carroll, J., Czuba, K., Duboue, P., Prager, J. *IBMs PIQUANT II in TREC 2005*, In Proceedings of the Fourteenth Text REtrieval Conference (TREC 2005), 2005.

[34] Cucerzan, S., Agichtein, E. *Factoid Question Answering over Unstructured and Structured Web Content*, In Proceedings of the Fourteenth Text REtrieval Conference (TREC 2005), 2005.

[35] Katz, B., Marton, G., Borchardt, G., Brownell, A., Felshin, S., Loreto, D., Louis-Rosenberg, J., Lu, B., Mora, F., Stiller, S., Uzuner, O., Wilcox, A. *External Knowledge Sources for Question Answering*, In Proceedings of the Fourteenth Text REtrieval Conference (TREC 2005), 2005.

[36] Ahn, D., Fissaha S., Jijkoun, V., Müller, K., de Rijke, M., Tjong Kim Sang, E. *Towards a Multi-Stream Question Answering-As-XML-Retrieval Strategy*, In Proceedings of the Fourteenth Text REtrieval Conference (TREC 2005), 2005.

[37] Ahn, K., Bos, J., Curran, J., R., Kor, D., Nissim. M., Webber, B. *Question Answering with QED at TREC-2005*, In Proceedings of the Fourteenth Text REtrieval Conference (TREC 2005), 2005.

[38] Gaizauskas, R., Greenwood,. M., A., Harkema,H., Hepple, M., Saggion, H., Sanka, A. *The University of Sheffields TREC 2005 Q& A Experiments*, In Proceedings of the Fourteenth Text REtrieval Conference (TREC 2005), 2005.

[39] Gotoh. *An Improved algorithm for matching biological sequences*, In Journal of Molecular Biology, number 162, pp. 705–708, 1982.

[40] Wei, K. *Improving Answer Precision and Recall of List Questions*, MSc thesis, School of Informatics, University of Edinburgh, UK, 2005.

[41] DeJong, K.,A., Spears, W.,M. *An Analysis of the Interacting Roles of Population Size and Crossover in Genetic Algorithms*, In Proceedings of the First Workshop Parallel Problem Solving from Nature, Springer-Verlag, Berlin, 1990. pp. 38-47.

[42] Grefenstette, J.,J. *Optimization of Control Parameters for Genetic Algorithms*, IEEE Trans. Systems, Man, and Cybernetics, Vol. SMC-16, No. 1, Jan./Feb. 1986, pp. 122–128.

[43] Holland ,J. *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`.

[44] Zhai, C., Lafferty, J. *A Study of Smoothing Methods for Language Models Applied to Information Retrieval*, ACM Transactions on Information Systems, Vol. 22, No. 2, April 2004, Pages 179-214.

[45] Song, F., Croft, W., B. *A General Language Model for Information Retrieval*, in Proceedings of the 1999 ACM CIKM International Converence on Knowledge and Management, Kansas City, Missouri, USA, November 2–6, 1999.

[46] Kay, M., Röscheisen, M. *Text-Translation Alignment*, Computational Linguistics, Volume 19, Number 1, 1993.

[47] Brown, P., F., Pietra, S., A., Pietra, V., J., Mercer, R., L. *The Mathematics of Statistical Machine Translation: Parameter Stimation*, Computational Linguistics, Volume 19, Number 2, 1993.

[48] Otto, E., Riff, M., C. *Towards an Efficient Evolutionary Decoding Algorithm for Statistical Machine Translation*, MICAI 2004, LNAI 2972, 2004, pp. 438-447.

[49] Landauer, T. K., Foltz, P. W., and Laham, D. *Introduction to Latent Semantic Analysis*, Discourse Processes, 25, 1998, 259–284.

[50] Kintsch, W. *Predication*, Cognitive Science, 25, 1998, 173–202.

[51] Whitley, D. *A Genetic Algorithm Tutorial*, `http://samizdat.mines.edu/ga_tutorial/ga_tutorial.ps`.

[52] Figueroa, A., Neumann, G. *Language Independent Answer Prediction from the Web*, In Proceedings of the FinTAL 5th International Conference on Natural Language Processing, August 23–25 in Turku, 2006, Finland, LNAI 4139, pp. 423–434.

[53] Figueroa, A., Atkinson, J. *Molecular Sequence Alignment for Extracting Answers for Where-typed Questions from Google Snippets*, In Proceedings of the 10th International Conference on knowledge-based & Intelligent Information & Engineering Systems, Bournemouth, 9-11 October, United Kingdom, 2006.

[54] Figueroa, A., Atkinson, J. *Using Syntactic Distributional Patterns for data-driven Answer Extraction from the Web*, Submitted to MICAI-2006, Mexico, 2006.

[55] Figueroa, A., Atkinson, J. *Molecular Sequence Alignment for Extracting Answers for Where-typed Questions from Google Snippets*, Accepted under revision in Journal of knowledge-based & Intelligent Information & Engineering Systems, 2006.

[56] Di Nunzio, G., M., Ferro, N., Jones, G., J., F., Peters, C. *CLEF 2005: Ad Hoc Track Overview*, Working Notes for the CLEF 2005 Workshop, 21-23 September, Vienna, Austria, 2005.

[57] Sacaleanu, B., Neumann, G. *DFKI's LT-Lab at the CLEF 2005 Multiple Language Question Answering Task*, Working Notes for the CLEF 2005 Workshop, 21-23 September, Vienna, Austria, 2005.

[58] Neumann, G., Xu, F. *Mining Natural Language Answers from the Web*, In International Journal of Web Intelligence and Agent Systems, Volume 2, Number 2,2004, 123–135.

[59] Ravichandran, D., Hovy, E. *Learning Surface Text Patterns for a Question Answering System*, In Proceedings of the ACL Conference, 2002.

[60] Basten, R. *Answering open-domain temporally restricted questions in a multilingual context*, Master's thesis, University of Twente and LT-Lab DFKI, 2005.

Appendix A

Sample Results

The following tables show samples of results obtained by the strategies presented in this work. The letter "*R*" on the headings of some columns stands for the word "*Rank*", and its value is the rank position of the corresponding answer string. Due to the size of the margins of a page and the length of some answer strings, tables do not contain all results, but representative samples were carefully selected.

All cases, in which there was no answer on the corresponding retrieved snippets, are not shown in the tables. In case of the CLEF question set, it worths to highlight that only representative provided correct answers were considered, if the reader want to know all possible right answers to a particular question, please refer to the original CLEF-corpus.

		Baseline		GAQA		GA-ASSA		PreGA	
Invention	*Inventor*	Answer	R	Answer	R	Answer	R	Answer	R
door, revolving	Theophilus von Kannel	THEOPHILUS	5	THEOPHILUS	1	THEOPHILUS	1	THEOPHILUS	1
dynamite	Alfred Nobel	ALFRED	1	NOBEL	1	NOBEL	1	NOBEL	1
elastic, fabric	Thomas Hancock	HANCOCK	8						
electronic mail (e-mail)	Ray Tomlinson	TOMLINSON	11	TOMLINSON	10				
elevator, passenger	Elisha Graves Otis	ELISHA	1	OTIS	1	OTIS	1		
engine, jet	Sir Frank Whittle	FRANK	1	FRANK	1	FRANK	1	WHITTLE	1
engine, steam	Thomas Savery	THOMAS	2	THOMAS	1	THOMAS	1		
escalator	Jesse W. Reno	RENO	1	RENO	1	RENO	1	RENO	1
facsimile (fax)	Alexander Bain	BAIN	4	ALEXANDER	1	ALEXANDER BAIN	1	ALEXANDER BAIN	1
film, photographic	George Eastman	EASTMAN	5			EASTMAN	3		
flask, vacuum (Thermos)	Sir James Dewar	DEWAR	2	JAMES	1	DEWAR	1	DEWAR	1
foods, frozen	Clarence Birdseye	BIRDSEYE	1	BIRDSEYE	3	BIRDSEYE	2		
Fresnel lens	Augustin-Jean Fresnel	AUGUSTIN	7	AUGUSTIN	1	AUGUSTIN	1		
fuel cell	William R. Grove	GROVE	2	GROVE	1	GROVE	1	GROVE	1
Geiger counter	Hans Geiger	HANS	2	HANS	1	HANS	1		
guitar, electric	Les Paul	PAUL	1	PAUL	1	PAUL	1	LES PAUL	1
helicopter	Igor Sikorsky	SIKORSKY	1			IGOR SIKORSKY	1	SIKORSKY	1
holography	Dennis Gabor	GABOR	1	GABOR	1	GABOR	1	GABOR	1
hypodermic syringe	Charles Gabriel Pravaz	WOOD	4					ALEXANDER WOOD	5
integrated circuit	Jack S. Kilby	ROBERT	1	JACK	1	KILBY	1	NOYCE	1
iron, electric	Henry W. Seely	SEELEY	3			SEELEY	2		
JELL-O(gelatin dessert)	Pearle B. Wait	COOPER	1	PETER	5	COOPER	5	PETER COOPER	8
Kevlar	Stephanie Kwolek	STEPHANIE	1	STEPHANIE	1	KWOLEK	1	KWOLEK	1
Kool-Aid(fruit drink mix)	Edwin E. Perkins	PERKINS	3	EDWIN PERKINS	4	EDWIN PERKINS	5	JAKE ROSS	4
Lego	Ole Kirk Christiansen	KIRK	6	CHRISTIANSEN	3	KIRK CHRISTIANSEN	6		
light bulb, incandescent	Thomas Alva Edison	EDISON	1	EDISON	1	EDISON	1	EDISON	1
light-emitting diode	Nick Holonyak, Jr.	HOLONYAK	2	HOLONYAK	4	HOLONYAK	6	HOLONYAK	2
linoleum	Frederick Walton	WALTON	1	FREDERICK	3	FREDERICK WALTON	5	WALTON	2
liquid crystal display	George Heilmeier	GEORGE	7					GEORGE HEILMEIER	7
locomotive	George Stephenson	STEPHENSON	2	TREVITHICK	1	TREVITHICK	2	RICHARD TREVITHICK	1
margarine	Hippolyte Mege-Mouries	HIPPOLYTE	5	HIPPOLYTE	1	HIPPOLYTE	1		
matches, friction	John Walker	WALKER	1	JOHN WALKER	1	WALKER	2	JOHN WALKER	1
microphone	David E. Hughes	HUGHES	17						
microscope, electron	Ernst Ruska	ERNST	3	ERNST	1	RUSKA	3	RUSKA	1
microwave oven	Percy L. Spencer	SPENCER	3	SPENCER	1	PERCY	2	SPENCER	1
Monopoly (board game)	Charles B. Darrow	DARROW	11	CHARLES	2	DARROW	2	DARROW	3
Morse code	Samuel F.B. Morse	SAMUEL	2	SAMUEL	1	SAMUEL	1	ALFRED VAIL	1
motor, electric	Thomas Davenport	DAVENPORT	2					CHARLES KETTERING	1
motor, outboard	Ole Evinrude	EVINRUDE	1	EVINRUDE		EVINRUDE		EVINRUDE	1
mouse, computer	Douglas Engelbart	DOUGLAS	1	DOUGLAS	1	DOUGLAS	1	DOUGLAS	1
Muzak	George Owen Squier	SQUIER	3					SQUIER	1
neon lighting	Georges Claude	CLAUDE	3			CLAUDE	3	CLAUDE	1
nuclear reactor	Enrico Fermi	FERMI	1	FERMI	1			FERMI	1

Table A.1: Some results for the inventor-inventions set of questions (A-N).

		Baseline		GAQA		GA-ASSA		PreGA	
Invention	*Inventor*	Answer	R	Answer	R	Answer	R	Answer	R
oil lamp	Aime Argand	ARGAND	2	ARGAND	1	ARGAND	1	ARGAND	1
oil well	Edwin Laurentine Drake	DRAKE	6					DRAKE	2
pacemaker, cardiac	Paul M. Zoll	GREATBATCH	1			GREATBATCH	1	GREATBATCH	1
paper clip	Johan Vaaler	VAALER	1	JOHANN	3	JOHANN	5	VAALER	1
paper towel	Arthur Scott	SCOTT	3			SCOTT	2		
parachute, modern	Andre-Jacques Garnerin	GARNERIN	4	BALDWIN	1				
parking meter	Carl C. Magee	MAGEE	3	MAGEE	2	MAGEE	1	MAGEE	1
pasteurization	Louis Pasteur	PASTEUR	2	LOUIS PASTEUR	1	PASTEUR	2	PASTEUR	1
pen, ballpoint	Lazlo Biro	BIRO	1	LASZLO	5				
periodic table	Dmitry Ivanovich Mendeleyev	MENDELEEV	2	MENDELEEV	1	MENDELEEV	1	MENDELEEV	1
personal watercraft, motorized	Bombardier, Inc.	BOMBARDIER	5					BOMBARDIER	1
phonograph	Thomas Alva Edison	THOMAS	2	EDISON	1	EDISON	1	EDISON	1
photocopying (xerography)	Chester F. Carlson	CHESTER	1	CHESTER	1	CHESTER	2	CHESTER	1
photography, instant	Edwin Herbert Land	LAND	1	EDWIN HERBERT	1	LAND	1	EDWIN	1
Play-Doh	Noah W. and Joseph S. McVicker	MCVICKER	1	NOAH	4			JOSEPH	4
plow, steel	John Deere	DEERE	3	DEERE	1	DEERE	1	DEERE	1
pocket watch	Peter Henlein	PETER	5	PETER	2				
polygraph (lie detector)	John A. Larson	JOHN	1	JOHN	1	JOHN	1		
Post-it Notes	Arthur Fry (3M)	FRY	5					FRY	3
potato chips	George Crum	CRUM	1	GEORGE	1	CRUM	2	GEORGE CRUM	1
printing press, movable type	Johannes Gutenberg	GUTENBERG	1	GUTENBERG	1	GUTENBERG	1	GUTENBERG	1
radar	Christian Huelsmeyer	ROBERT	3	WATSON WATT	1	ROBERT	1	WATT	1
razor, electric	Jacob Schick	SCHICK	2	SCHICK	1	SCHICK	1	SCHICK	1
razor, safety	King Camp Gillette	GILLETTE	1	GILLETTE	1	GILLETTE	1	GILLETTE	1
reaper, mechanical	Cyrus Hall McCormick	CYRUS	1	MCCORMICK	1	MCCORMICK	1	MCCORMICK	1
refrigerator	John Gorrie	JOHN	4	STANDARD	1	STANDARD	1	STANDARD (John)	1
remote control, television	Robert Adler	POLLEY	8					POLLEY	1
respirator	Forrest M. Bird	BIRD	2			BIRD	1	FORREST	1
revolver	Samuel Colt	SAMUEL	2	COLT	1	COLT	1	COLT	1
roller coaster	LeMarcus A. Thompson	THOMPSON	3			THOMPSON	2		
rubber, vulcanized	Charles Goodyear	GOODYEAR	3	GOODYEAR	1	CHARLES	1	GOODYEAR	
rubber band	Stephen Perry	PERRY	2	PERRY	4			PERRY	1
safety pin	Walter Hunt	HUNT	3	WALTER	1	HUNT	1	HUNT	1
Scotch tape	Richard Drew (3M)	RICHARD	3	DREW	1	RICHARD	1	RICHARD	1
scuba gear	Jacques Cousteau, Emile Gagnan	COUSTEAU	4	COUSTEAU	2				
slot machine	Charles Fey	FEY	3	CHARLES FEY	1	FEY	1	FEY	1
snowmobile	Joseph-Armand Bombardier	BOMBARDIER	1	BOMBARDIER	1	BOMBARDIER	1	BOMBARDIER	1
soft drinks, carbonated	Joseph Priestley	TORBERN	14					TORBERN BERGMAN	1
sonar	Paul Langevin	LANGEVIN	10	LANGEVIN	2	LANGEVIN	5		
stamps, postage	Sir Rowland Hill	HILL	11						
steamboat, successful	Robert Fulton	FULTON	1	FULTON	1	FULTON	1	FULTON	1
steel, mass-production	Henry Bessemer	BESSEMER	4			BESSEMER	4	BESSEMER	1

Table A.2: Some results for the inventor-inventions set of questions (O-S).

	Baseline		GAQA		GA-ASSA		PreGA	
Invention	*Inventor*	Answer	R	Answer	R	Answer	R	Answer	R
steel, stainless	Harry Brearley	BREARLY	13	HARRY	1			BREARLY	1
stereo, personal	Sony Corp.	SONY	2					SONY	2
stereophonic sound recording	Alan Dower Blumlein	BLUMLEIN	4	DOWER BLUMLEIN	1	BLUMLEIN	1	BLUMLEIN	1
stock ticker	Edward A. Calahan	CALAHAN	16						
stove, electric	William Hadaway	HADAWAY	4	WILLIAM	9				
straw, drinking	Marvin Stone	STONE	1	STONE	1	STONE	2	STONE	1
submarine	Cornelis Drebbel	DREBBEL	8			SIMON LAKE	3	LAKE	1
sunglasses	James Ayscough	AYSCOUGH	22						
tampon, cotton	Earle Cleveland Haas	HASS	11			HAAS	6	HASS	5
tea bag	Thomas Sullivan	SULLIVAN	2	THOMAS SULLIVAN	1	SULLIVAN	1	THOMAS	1
teddy bear	Morris Michtom	MICHTOM	15						
Teflon	Roy Plunkett	PLUNKETT	2	PLUNKETT	2	PLUNKETT	10		
telegraph	Samuel F.B. Morse	MORSE	1	MORSE	1	MORSE	1	MORSE	1
telephone, wired-line	Alexander Graham Bell	BELL	5			BELL	7	BELL	2
telephone, mobile	Bell Laboratories	BELL	2	BELL	3	BELL	2		
thermometer	Galileo	GALILEO	1	GALILEO	1	GALILEO	1	GALILEO	1
thermostat	Andrew Ure	JOHNSON	1	JOHNSON	1	JOHNSON	1	WARREN JOHNSON	1
threshing machine	Andrew Meikle	MEIKLE	3	ANDREW MEIKLE	1	MEIKLE	1	MEIKLE	1
tire, pneumatic	John Boyd Dunlop	DUNLOP	3	DUNLOP	1	DUNLOP	1	BOYD DUNLOP	3
toaster, electric	Crompton Co.	CROMPTON	5			CROMPTON	4		
toilet, flush	Sir John Harington	JOHN	6	JOHN	2	JOHN	2	HARRINGTON	10
tractor	John Froehlich	BENJAMIN	2	JOHN	1	HOLT	1	HOLT (Benjamin)	1
traffic lights, automatic	Garrett A. Morgan	GARRETT	3			MORGAN	2	MORGAN	2
typewriter	Christopher Latham Sholes	SHOLES	1	SHOLES	1	SHOLES	1		
vaccination	Edward Jenner	JENNER	14					JENNER	3
vacuum cleaner, electric	Herbert Cecil Booth	SPANGLER	1	SPANGLER	1	SPANGLER	1	SPANGLER	1
Velcro	George de Mestral	MESTRAL	1	GEORGE		GEORGE	1	MESTRAL	1
videocassette recorder	Sony Corp.	SONY	4					SONY	2
videotape	Charles Ginsburg	CHARLES	5	GINSBURG	3	CHARLES GINSBURG	4	CHARLES GINSBURG	7
washing machine, electric	Alva J. Fisher	FISHER	5						
World Wide Web	Tim Berners-Lee	TIM	1	LEE	1	BERNERS LEE	2	LEE	1
zipper	Whitcomb L. Judson	JUDSON	1	JUDSON	2	WHITCOMB	1	WHITCOMB	1

Table A.3: Some results for the inventor-inventions set of questions (S-Z).

		Baseline		GAQA		GA-ASSA	
CLEF-Question	*CLEF-Answer*	Answer	R	Answer	R	Answer	R
Who is Jean-Bertrand Aristide?	EXILED HAITIAN PRESIDENT	PRESIDENT	1	PRESIDENT	4	PRESIDENT	5
Who is the Japanese Emperor?	AKIHITO	AKIHITO	2	AKIHITO	1	AKIHITO	2
Who is Paul Simon?	SINGER-SONGWRITER	SONGWRITER	17	SONGWRITER	1	SONGWRITER	1
Who are the carabinieri?	STATE POLICE	POLICE	9			POLICE	2
Who freed the town of Sainte-Mere-Eglise on D-day?	U.S. PARATROOPERS	PARACHUTISTS	7	PARACHUTISTS	3		
Who plays the role of a prostitute in "Taxi Driver"?	JODIE FOSTER	FOSTER	3	JODIE	1	JODIE	1
Who directed the Spanish film, "The Worst Years of Our Lives"?	EMILIO MARTNEZ LZARO	EMILIO	30	EMILIO	9	EMILIO	3
Who won the IV International Chess Tournament "Ciudad de Pamplona"?	JORDI MAGEM	JORDI	25				
Who is Christo?	ARTIST OF HUNGARIAN ORIGIN	ARTIST	26				
Who was the president of the United States in 1994?	BILL CLINTON	CLINTON	3			CLINTON	2
Who was the president of North Korea before 1994?	KIM II SUNG, KIM IL SUNG	KIM	1	IL	6	KIM	2
Who is Andy Warhol?	ARTIST	ARTIST	5			ARTIST	6
Who is Keith Richards?	GUITARIST ROLLING STONES	GUITARIST	4				
Who discovered the AIDS virus?	LUC MONTAGNIER	MONTAGNIER	8			MONTAGNIER	7
Who is the German Minister for the Environment?	ANGELA MERKEL	JuRGEN	1	TRITTIN	1	TRITTIN	2
Who is the president of South-Yemen?	ALI SALEM	ALI	1			ALI	4
What organization fights for the liberation of Tamil Eelam?	LTTE	LTTE	2	LTTE	2		
What is the name of the national Belgian airline?	SABENA	SABENA	4	SABENA	1	SABENA	1
Who manufactures Invirase?	HOFFMANN-LA ROCHE	ROCHE	6	ROCHE	1	ROCHE	1
Who is the Secretary General of the German section of Amnesty International?	VOLKMAR DEILE	IRENE	10				
Who is Peeter Tulviste?	VICE CHANCELLOR OF TARTU UNIV.	PROFESSOR	2	PROFESSOR	1		
Who is Nodar Djavakhishvili?	HEAD OF THE GEORGIAN C. BANK	CHAIRMAN	1				
What is the name of the European standard for digital mobile communications?	GLOBAL SYSTEM FOR MOBILE COMMUNICATION	GMS	1	GMS	1	GMS	2
Who committed the terrorist attack in the Tokyo underground?	AUM SHINRIKYO SHOKO ASAHARA	AUM	10	AUM	8	SHINRIKYO	8
What is the name of the Queen of the Netherlands?	BEATRICE, BEATRIX, BEATRIZ	JULIANA	1	JULIANA	1	JULIANA	1
Who is Montxo Armendariz?	SPANISH FILMMAKER	DIRECTOR	3	DIRECTOR	1	DIRECTOR	1
Who won the Cannes Film Festival in 1995?	EMIR KUSTURICA	KAURISMAKI	10				
Who is Paul Feyerabend?	EPISTEMOLOGIST SCIENCE PHILOSOPHER	EPISTEMOLOGIST	22				
Who was the last king of Romania?	MICHAEL OF ROMANIA	MICHAEL	1			MIHAI	
Who wrote "The Little Prince"?	ANTOINE DE SAINT-EXUPERY	SAINT EXUPERY	2	EXUPERY	1	SAINT EXUPERY	1
What is the name of the Queen of Denmark?	MARGRETHE II	MARGRETHE	1	MARGRETHE	1	MARGRETHE	1
Who is Wim Duisenberg?	PRESIDENT OF THE DUTCH CENTRAL BANK	PRESIDENT	5				
Who is Valentina Tereshkova?	FIRST WOMAN COSMONAUT	COSMONAUT	3	COSMONAUT	8	COSMONAUT	4
Who is Jorge Amado?	BRAZILIAN NOVELIST/WRITER	NOVELIST	4			NOVELIST	4

Table A.4: Some results for CLEF question set (Baseline,GA-QA,GA-ASSA).

		PreGA	
CLEF-Question	*CLEF-Answer*	Answer	Rank
Who is Jean-Bertrand Aristide?	EXILED HAITIAN PRESIDENT	PRESIDENT	3
Who is the Japanese Emperor?	AKIHITO		
Who is Paul Simon?	SINGER-SONGWRITER		
Who are the carabinieri?	STATE POLICE		
Who freed the town of Sainte–Mere–Eglise on D-day?	U.S. PARATROOPERS	PARACHUTISTS	3
Who plays the role of a prostitute in "Taxi Driver"?	JODIE FOSTER	FOSTER	2
Who directed the Spanish film, "The Worst Years of Our Lives"?	EMILIO MARTNEZ LZARO	EMILIO	2
Who won the IV International Chess Tournament "Ciudad de Pamplona"?	JORDI MAGEM	JORDI	25
Who is Christo?	ARTIST OF HUNGARIAN ORIGIN	ARTIST	26
Who was the president of the United States in 1994?	BILL CLINTON		
Who was the president of North Korea before 1994?	KIM II SUNG, KIM IL SUNG	IL	2
Who is Andy Warhol?	ARTIST		
Who is Keith Richards?	GUITARIST ROLLING STONES	GUITARIST	3
Who discovered the AIDS virus?	LUC MONTAGNIER	MONTAGNIER	4
Who is the German Minister for the Environment?	ANGELA MERKEL	TRITTIN	1
Who is the president of South-Yemen?	ALI SALEM	ALI	4
What organization fights for the liberation of Tamil Eelam?	LTTE	LTTE	1
What is the name of the national Belgian airline?	SABENA	SABENA	1
Who manufactures Invirase?	HOFFMANN-LA ROCHE	ROCHE	1
Who is the Secretary General of the German section of Amnesty International?	VOLKMAR DEILE	IRENE	10
Who is Peeter Tulviste?	VICE CHANCELLOR OF TARTU UNIVERSITY	A PSYCHOLOGY PROFESSOR	2
Who is Nodar Djavakhishvili?	HEAD OF THE GEORGIAN CENTRAL BANK	CHAIRMAN	1
What is the name of the European standard for digital mobile communications?	GLOBAL SYSTEM FOR MOBILE COMMUNICATION	GMS	1
Who committed the terrorist attack in the Tokyo underground?	AUM SHINRIKYO SHOKO ASAHARA	AUM SHINRIKYO	4
What is the name of the Queen of the Netherlands?	BEATRICE, BEATRIX, BEATRIZ	WILHELMINA	1
Who is Montxo Armendariz?	SPANISH FILMMAKER	DIRECTOR	1
Who won the Cannes Film Festival in 1995?	EMIR KUSTURICA	KAURISMAKI	5
Who is Paul Feyerabend?	EPISTEMOLOGIST SCIENCE PHILOSOPHER	MOST CONTROVERSIAL PHILOSOPHERS	4
Who was the last king of Romania?	MICHAEL OF ROMANIA		
Who wrote "The Little Prince"?	ANTOINE DE SAINT–EXUPERY	SAINT–EXUPERY	1
What is the name of the Queen of Denmark?	MARGRETHE II	MARGRETHE	1
Who is Wim Duisenberg?	PRESIDENT OF THE DUTCH CENTRAL BANK	FOUNDING CHAIRMAN OF THE EUROPEAN CENTRAL	5
Who is Valentina Tereshkova?	FIRST WOMAN COSMONAUT	COSMONAUT	4
Who is Jorge Amado?	BRAZILIAN NOVELIST/WRITER	GREATEST NOVELIST	6

Table A.5: Some results for CLEF question set (PreGA).

		Baseline		GAQA		GA-ASSA	
Symphony	*Composer*	Answer	R	Answer	R	Answer	R
No. 14 Ararat	Alan Hovhaness	HOVHANESS	5	ALAN	4	HOVHANESS	4
No. 15 Silver Pilgrimage	Alan Hovhaness	HOVHANESS	11			HOVHANESS	1
No. 16 Kayagum	Alan Hovhaness	HOVHANESS	5	ALAN	4	HOVHANESS	5
No. 17 Symphony for Metal Orchestra	Alan Hovhaness	HOVHANESS	11	HOVHANESS	1		
No. 18 Circe	Alan Hovhaness	HOVHANESS	12				
No. 19 Vishnu	Alan Hovhaness	HOVHANESS	1	HOVHANESS	1	ALAN	1
No. 20 Three Journeys to a Holy Mountain	Alan Hovhaness	HOVHANESS	1	HOVHANESS	1	HOVHANESS	1
No. 21 Symphony Etchmiadzin	Alan Hovhaness	HOVHANESS	1	HOVHANESS	1	HOVHANESS	1
No. 22 City of Light	Alan Hovhaness	HOVHANESS	1	ALAN HOVHANESS	1	HOVHANESS	1
No. 23 Ani	Alan Hovhaness	HOVHANESS	11			HOVHANESS	1
No. 24 Majnun	Alan Hovhaness	HOVHANESS	8			HOVHANESS	1
No. 25 Odysseus	Alan Hovhaness	HOVHANESS	12			HOVHANESS	10
No. 46 To the Green Mountains	Alan Hovhaness	HOVHANESS	10	ALAN HOVHANESS	1	ALAN HOVHANESS	1
No. 49 Christmas Symphony	Alan Hovhaness	HOVHANESS	12	ALAN HOVHANESS	1	ALAN HOVHANESS	3
No. 50 Mount St. Helens	Alan Hovhaness	ALAN	2	ALAN HOVHANESS	1	HOVHANESS	1
No. 52 Journey to Vega	Alan Hovhaness	HOVHANESS	13				
No. 53 Star Dawn	Alan Hovhaness	HOVHANESS	4			HOVHANESS	1
No. 57 Cold Mountain	Alan Hovhaness	HOVHANESS	12				
No. 66 Hymn to Glacier Peak	Alan Hovhaness	HOVHANESS	7	ALAN	1	HOVHANESS	1
No. 67 Hymn to the Mountains	Alan Hovhaness	HOVHANESS	10				
No. 3 The Camp Meeting	Charles Ives	CHARLES	1	IVES	1	IVES	1
No. 1 Der Titan The Titan	Gustav Mahler	MAHLER	1	GUSTAV	2	MAHLER	2
No. 2 Auferstehung Resurrection	Gustav Mahler	MAHLER	1	MAHLER	4	MAHLER	1
No. 6 Tragic	Gustav Mahler	MAHLER	2	GUSTAV	1	MAHLER	2
No. 2 Lobgesang Hymn of Praise	Felix Mendelssohn	MENDELSSOHN	3	FELIX	2	MENDELSSOHN	2
No. 3 Scottish	Felix Mendelssohn	MENDELSSOHN	1	FELIX	2	MENDELSSOHN	2
No. 4 Italian	Felix Mendelssohn	MENDELSSOHN	2	FELIX	1	MENDELSSOHN	1
No. 5 Reformation	Felix Mendelssohn	FELIX	2	FELIX MENDELSSOHN	1	MENDELSSOHN	1
No. 25 Little G minor	Wolfang Amadeus Mozart	MOZART	1	MOZART	1	MOZART	1
No. 31 Paris	Wolfang Amadeus Mozart	MOZART	2	WOLFGANG AMADEUS	2	WOLFGANG	5
No. 32 Overture in the Italian style	Wolfang Amadeus Mozart	MOZART	4	MOZART	4	MOZART	7
No. 35 Haffner	Wolfang Amadeus Mozart	AMADEUS	3	WOLFGANG AMADEUS MOZART	1	MOZART	2
No. 36 Linz	Wolfang Amadeus Mozart	WOLFGANG	6	MOZART	2	MOZART	2
No. 38 Prague	Wolfang Amadeus Mozart	WOLFGANG	7	MOZART	1	MOZART	1
No. 41 Jupiter	Wolfang Amadeus Mozart	MOZART	3	WOLFGANG AMADEUS	2	MOZART	1
No. 2 The Four Temperaments	Carl Nielsen	NIELSEN	2	CARL	1	NIELSEN	1
No. 3 Sinfonia Espansiva	Carl Nielsen	CARL	2	CARL	1	NIELSEN	1
No. 4 The Inextinguishable	Carl Nielsen	CARL	2	CARL	1	NIELSEN	1
No. 6 Sinfonia Semplice	Carl Nielsen	NIELSEN	3	CARL	1	NIELSEN	2
No. 2 Cambridge	Hubert Parry	CHARLES	14	SIR CHARLES HH PARRY	5	CHARLES HH	7
No. 5 Korean	Krzysztof Penderecki	PYOTR	13	PYOTR ILYICH	4		
No. 7 The Seven Gates of Jerusalem	Krzysztof Penderecki	PENDERECKI	3	KRZYSZTOF	3	PENDERECKI	2

Table A.6: Some results for a set of questions aiming at composers (H-P) of symphonies (`Baseline`,`GA-QA`,`GA-ASSA`).

		PreGA	
Symphony	*Composer*	Answer	Rank
No. 14 Ararat	Alan Hovhaness		
No. 15 Silver Pilgrimage	Alan Hovhaness	ALAN HOVHANESS	1
No. 16 Kayagum	Alan Hovhaness		
No. 17 Symphony for Metal Orchestra	Alan Hovhaness		
No. 18 Circe	Alan Hovhaness	HOVHANESS	3
No. 19 Vishnu	Alan Hovhaness	ALAN HOVHANESS	1
No. 20 Three Journeys to a Holy Mountain	Alan Hovhaness	HOVHANESS	1
No. 21 Symphony Etchmiadzin	Alan Hovhaness	HOVHANESS	1
No. 22 City of Light	Alan Hovhaness	ALAN HOVHANESS	1
No. 23 Ani	Alan Hovhaness	ALAN HOVHANESS	3
No. 24 Majnun	Alan Hovhaness		
No. 25 Odysseus	Alan Hovhaness		
No. 46 To the Green Mountains	Alan Hovhaness	ALAN HOVHANESS	1
No. 49 Christmas Symphony	Alan Hovhaness	ALAN HOVHANESS	1
No. 50 Mount St. Helens	Alan Hovhaness	ALAN HOVHANESS	1
No. 52 Journey to Vega	Alan Hovhaness		
No. 53 Star Dawn	Alan Hovhaness	ALAN HOVHANESS	1
No. 57 Cold Mountain	Alan Hovhaness		
No. 66 Hymn to Glacier Peak	Alan Hovhaness	ALAN HOVHANESS	1
No. 67 Hymn to the Mountains	Alan Hovhaness	HOVHANESS	1
No. 3 The Camp Meeting	Charles Ives	CHARLES	1
No. 1 Der Titan The Titan	Gustav Mahler	GUSTAV	1
No. 2 Auferstehung Resurrection	Gustav Mahler	1	
No. 6 Tragic	Gustav Mahler	MAHLER	1
No. 2 Lobgesang Hymn of Praise	Felix Mendelssohn	FELIX MENDELSSOHN	2
No. 3 Scottish	Felix Mendelssohn	FELIX	1
No. 4 Italian	Felix Mendelssohn	FELIX MENDELSSOHN	1
No. 5 Reformation	Felix Mendelssohn	MENDELSSOHN	1
No. 25 Little G minor	Wolfang Amadeus Mozart	MOZART	1
No. 31 Paris	Wolfang Amadeus Mozart	MOZART	2
No. 32 Overture in the Italian style	Wolfang Amadeus Mozart	MOZART	4
No. 35 Haffner	Wolfang Amadeus Mozart	WOLFGANG AMADEUS MOZART	1
No. 36 Linz	Wolfang Amadeus Mozart	MOZART	2
No. 38 Prague	Wolfang Amadeus Mozart	MOZART	1
No. 41 Jupiter	Wolfang Amadeus Mozart	MOZART	1
No. 2 The Four Temperaments	Carl Nielsen	CARL NIELSEN	1
No. 3 Sinfonia Espansiva	Carl Nielsen	CARL	1
No. 4 The Inextinguishable	Carl Nielsen	CARL	1
No. 6 Sinfonia Semplice	Carl Nielsen	CARL NIELSEN	1
No. 2 Cambridge	Hubert Parry		
No. 5 Korean	Krzysztof Penderecki	PYOTR	1
No. 7 The Seven Gates of Jerusalem	Krzysztof Penderecki		

Table A.7: Some results for a set of questions aiming at composers (H-P) of symphonies (PreGA).

		Baseline		GAQA		GA-ASSA		PreGA	
Symphony	*Composer*	Answer	R	Answer	R	Answer	R	Answer	R
No. 2 Iron and Steel, Symphony No. 2	Sergei Prokofiev	PROKOFIEV	1			PROKOFIEV	2		
No. 4 Arabescata	Einojuhani Rautavaara	EINOJUHANI	3	EINOJUHANI	1	RAUTAVAARA	1	EINOJUHANI	1
				RAUTAVAARA				RAUTAVAARA	
No. 7 Angel of Light	Einojuhani Rautavaara	RAUTAVAARA	2	EINOJUHANI	1	RAUTAVAARA	1	EINOJUHANI	1
				RAUTAVAARA				RAUTAVAARA	
No. 8 The Journey	Einojuhani Rautavaara	RAUTAVAARA	8			RAUTAVAARA	2	EINOJUHANI	1
								RAUTAVAARA	
No. 3 Symphony With Organ	Camille Saint-Saens	SAINT SAeNS	2	CAMILLE SAINT	2	CAMILLE	4	CAMILLE SAINT	1
No. 4 The Tragic	Franz Schubert	SCHUBERT	8	SCHUBERT	8			SCHUBERT	4
No. 8 Unfinished Symphony	Franz Schubert	SCHUBERT	1	SCHUBERT	1	SCHUBERT	2	SCHUBERT	1
No. 1 Frhlingssinfonie Spring	Robert Schaumann	SCHUMANN	2	ROBERT	2	ROBERT	4		
No. 3 Rheinische Rhenish	Robert Schaumann	SCHUMANN	4	SCHUMANN	3	SCHUMANN	6		
No. 2 To October	Dmitri Shostakovich	SHOSTAKOVICH	14					DMITRY	2
								SHOSTAKOVICH	
No. 3 The First of May	Dmitri Shostakovich	SHOSTAKOVICH	18						
No. 7 Leningrad	Dmitri Shostakovich	SHOSTAKOVICH	2	SHOSTAKOVICH	2	SHOSTAKOVICH	2	DMITRY	2
No. 11 The Year 1905	Dmitri Shostakovich	SHOSTAKOVICH	3	SHOSTAKOVICH	2	SHOSTAKOVICH	4	DMITRY	1
								SHOSTAKOVICH	
No. 12 The Year 1917 the Memory of Lenin	Dmitri Shostakovich	SHOSTAKOVICH	3	SHOSTAKOVICH	1	SHOSTAKOVICH	1	SHOSTAKOVICH	1
No. 13 Babi Yar	Dmitri Shostakovich	SHOSTAKOVICH	5	SHOSTAKOVICH	1	SHOSTAKOVICH	1	SHOSTAKOVICH	1
No. 1 in B-flat major Symphony No. 1	Charles Villiers Standford	VILLIERS	16					VILLIERS	3
No. 2 in d minor Elegiac	Charles Villiers Standford	VILLIERS	17	CHARLES	1	CHARLES	2		
						VILLIERS		VILLIERS	
no 6 in E flat major In Memoriam G F Watts	Charles Villiers Standford	CHARLES	13	CHARLES	6	CHARLES	9		
no 7 in D minor Symphony No.7 (Stanford)	Charles Villiers Standford	CHARLES	11	CHARLES	4	CHARLES	9		
No. 1 Winter Dreams	Peter Tchaikovsky	TCHAIKOVSKY	5	PYOTR	2	TCHAIKOVSKY	2	PYOTR	1
No. 2 Little Russian	Peter Tchaikovsky	TCHAIKOVSKY	1	PETER	1	TCHAIKOVSKY	4	TCHAIKOVSKY	1
No. 3 Polish	Peter Tchaikovsky	TCHAIKOVSKY	13						
No. 7 Sinfonia Antartica	Ralph Vaughan Williams	VAUGHAN	1	RALPH	1	VAUGHAN	1	WILLIAMS	1
No. 1 O Imprevisto The Unforseen	Heitor Villa-Lobos	VILLA LOBOS	15	VILLA LOBOS	9				
No. 2 Asceno The Ascension	Heitor Villa-Lobos	VILLA LOBOS	16						
No. 3 A Guerra The War	Heitor Villa-Lobos	VILLA LOBOS	12						
No. 4 A Vitria The Victory	Heitor Villa-Lobos	VILLA LOBOS	19						
for Organ No. 9 Gothique Gothic - for Organ	Charles-Marie Widor	WIDOR	12	WIDOR	3	CHARLES	2		

Table A.8: Some results for a set of questions aiming at composers (P-W) of symphonies.

		Baseline		GAQA		GA-ASSA		PreGA	
Country	*President*	Answer	R	Answer	R	Answer	R	Answer	R
Ecuador	ALFREDO PALACIO	PALACIO	6					PALACIO	9
Egypt	HOSNI MUBARAK	HOSNI	2	HOSNI	1	MUBARAK	1		
El Salvador	ANTONIO SACA	SACA	1	ANTONIO	2	ANTONIO	1	SACA	2
Equatorial Guinea	TEODORO OBIANG MBASOGO	OBIANG	1	OBIANG	1	OBIANG	1	MBASOGO	2
Eritrea	ISAIAS AFEWERKI	ISAIAS	1	ISAIAS	2	ISAIAS	1	AFEWERKI	2
Estonia	ARNOLD RUEUETEL	RUEUETEL	4	ARNOLD	3	ARNOLD	6		
Ethiopia	GIRMA WOLDE-GIORGIS	GIRMA	4	GIRMA WOLDE GIORGIS	1	GIRMA	6	GIRMA WOLDE	2
Fiji	RATU JOSEFA ILOILO	ILOILO	2	ILOILO	1	ILOILO	1	JOSEFA ILOILO	2
Finland	TARJA HALONEN	HALONEN	1	TARJA	2	HALONEN	1	TARJA HALONEN	2
France	JACQUES CHIRAC	CHIRAC	2			CHIRAC	2	CHIRAC	2
Gabon	OMAR BONGO	BONGO	1	BONGO	2	BONGO	1	OMAR	2
Gambia	YAHYA JAMMEH	JAMMEH	1	JAMMEH	1	JAMMEH	1		
Georgia	MIKHAIL SAAKASHVILI	SAAKASHVILI	1	SAAKASHVILI	1	SAAKASHVILI	1		
Germany	HORST KOEHLER	KOEHLER	12						
Ghana	JOHN KUFUOR	KUFUOR	1	KUFUOR	1	KUFUOR	1		
Greece	KAROLOS PAPOULIAS	PAPOULIAS	3	PAPOULIAS	6	PAPOULIAS	5		
Guatemala	OSCAR BERGER	OSCAR	1	OSCAR	1	OSCAR	1	BERGER	3
Guinea	LANSANA CONTE	CONTE	1			CONTE	2		
Guinea Bissau	JOAO BERNARDO VIEIRA	VIEIRA	3	VIEIRA	1	VIEIRA	1		
Guyana	BHARRAT JAGDEO	JAGDEO	3	BHARRAT	1	JAGDEO	1		
Haiti	RENE PREVAL	PREVAL	2	PREVAL	3	PREVAL	1	PREVAL	2
Honduras	MANUEL ZELAYA	ZELAYA	3	ZELAYA	9	MANUEL	9		
Hungary	LASZLO SOLYOM	SOLYOM	1	LASZLO	1	LASZLO	1		
Iceland	OLAFUR RAGNAR GRIMSSON	GRIMSSON	1	GRIMSSON	1	GRIMSSON	1	RAGNAR	2
India	ABDUL KALAM	KALAM	2	KALAM	1	KALAM	5	ABDUL KALAM	2
Iran	MAHMOUD AHMADINEJAD	AHMADINEJAD	3	MAHMOUD AHMADINEJAD	1	AHMADINEJAD	2	AHMADINEJAD	3
Iraq	JALAL TALABANI	TALABANI	4	TALABANI	1	TALABANI	2	JALAL TALABANI	4
Ireland	MARY MCALEESE	MARY	4	MCALEESE	1	MARY	1		
Israel	MOSHE KATSAV	KATSAV	12						
Italy	GIORGIO NAPOLITANO	NAPOLITANO	13	NAPOLITANO	10				
Kazakhstan	NURSULTAN NAZARBAYEV	NAZARBAYEV	2	NAZARBAYEV	1	NAZARBAYEV	1	NURSULTAN	2
Kenya	MWAI KIBAKI	KIBAKI	1	KIBAKI	7	KIBAKI	3	KIBAKI	4
Kiribati	ANOTE TONG	TONG	13						
Kyrgyzstan	KURMANBEK BAKIYEV	BAKIYEV	12						
Laos	KHAMTAI SIPHANDON	SIPHANDON	11						
Latvia	VAIRA VIKE-FREIBERGA	VIKE FREIBERGA	3	FREIBERGA	1	VAIRA	1	VIKE FREIBERGA	2
Lebanon	EMILE LAHOUD	LAHOUD	5	LAHOUD	4	LAHOUD	2	EMILE	3
Liberia	ELLEN JOHNSON-SIRLEAF	ELLEN	3	SIRLEAF	1	ELLEN	1	JOHNSON SIRLEAF	2
Lithuania	VALDAS ADAMKUS	VALDAS	2	VALDAS	1	VALDAS	1	ADAMKUS	2

Table A.9: Some results for a set of questions aiming at Presidents of countries (A-L).

	Baseline		GAQA		GA-ASSA		PreGA	
Country	*President*	Answer	R	Answer	R	Answer	R	Answer	R
Madagascar	RAVALOMANANA	RAVALOMANANA	15	RAVALOMANANA	1	RAVALOMANANA	1		
Malawi	BINGU WA MUTHARIKA	MUTHARIKA	2	BINGU	1	MUTHARIKA	1	BINGU WA	2
Malta	EDWARD FENECH ADAMI	ADAMI	15			ADAMI	3		
Marshall Islands	KESSAI NOTE	NOTE	2	NOTE	2	NOTE	2		
Mauritius	SIR ANEROOD JUGNAUTH	JUGNAUTH	6	JUGNAUTH	9	JUGNAUTH	6		
Mexico	VICENTE FOX	VICENTE	1	VICENTE	1	FOX	1		
Micronesia	JOSEPH URUSEMAL	JOSEPH	2	URUSEMAL	6	JOSEPH	4		
Moldova	VLADIMIR VORONIN	VORONIN	2	VORONIN	1	VORONIN	1	VLADIMIR VORONIN	2
Mongolia	NAMBARYN ENKHBAYAR	ENKHBAYAR	11	ENKHBAYAR	2	ENKHBAYAR	3		
Mozambique	ARMANDO GUEBUZA	GUEBUZA	1	ARMANDO	4	GUEBUZA	2	GUEBUZA	3
Namibia	HIFIKEPUNYE POHAMBA	POHAMBA	3	POHAMBA	2	POHAMBA	3		
Nauru	LUDWIG SCOTTY	LUDWIG	7			SCOTTY	3	LUDWIG SCOTTY	9
Nicaragua	ENRIQUE BOLANOS	ENRIQUE	2	ENRIQUE	1	ENRIQUE	1		
Niger	TANDJA MAMADOU	TANDJA	2	MAMADOU	5	TANDJA	1	TANDJA	5
Nigeria	OLUSEGUN OBASANJO	OBASANJO	1	OLUSEGUN	1	OBASANJO	1	OLUSEGUN OBASANJO	2
Pakistan	PERVEZ MUSHARRAF	MUSHARRAF	1	MUSHARRAF	1	MUSHARRAF	1	PERVEZ MUSHARRAF	2
Palau	TOMMY REMENGESAU	REMENGESAU	1	TOMMY	2	REMENGESAU	1		
Palestine	MAHMOUD ABBAS	ABBAS	3	ABBAS	7	ABBAS	2	ABBAS	2
Panama	MARTIN TORRIJOS	TORRIJOS	1	MARTIN TORRIJOS	8	TORRIJOS	2		
Paraguay	NICANOR DUARTE	DUARTE	20			NICANOR DUARTE	10		
Peru	ALEJANDRO TOLEDO	TOLEDO	1	ALEJANDRO	1	ALEJANDRO	1	ALEJANDRO TOLEDO	2
Poland	LECH KACZYNSKI	KACZYNSKI	6			LECH	3		
Portugal	ANIBAL CAVACO SILVA	CAVACO	11						
Romania	TRAIAN BASESCU	BASESCU	8	TRAIAN	1	TRAIAN	4		
Russia	VLADIMIR PUTIN	PUTIN	1	VLADIMIR PUTIN	1	PUTIN	1		
Rwanda	PAUL KAGAME	KAGAME	1	PAUL	1	KAGAME	1	KAGAME	2
Senegal	ABDOULAYE WADE	WADE	1	WADE	1	WADE	1	ABDOULAYE WADE	2
Seychelles	JAMES MICHEL	JAMES	6	JAMES	1	JAMES	1	JAMES MICHEL	3
Sierra Leone	AHMAD TEJAN KABBAH	KABBAH	3	TEJAN	1	KABBAH	1	KABBAH	2
Singapore	S.R. NATHAN	NATHAN	3	NATHAN	4	NATHAN	4		
Slovenia	JANEZ DRNOVEK	JANEZ	1			JANEZ	3	JANEZ	3
Somalia de facto	ABDULLAHI YUSUF AHMED	YUSUF	5						
South Africa	THABO MBEKI	MBEKI	1	THABO	1	MBEKI	1		
Sri Lanka	MAHINDA RAJAPAKSE	MAHINDA	2	MAHINDA	1	MAHINDA	1	RAJAPAKSE	4
Suriname	RONALD VENETIAAN	VENETIAAN	1	RONALD VENETIAAN	1	VENETIAAN	1	RONALD	2
Syria	BASHAR AL-ASSAD	BASHAR	1	ASSAD	1	AL	1	BASHAR AL ASSAD	4
Taiwan	CHEN SHUI-BIAN	CHEN	1	CHEN SHUI BIAN	1	SHUI BIAN	1	BIAN	2
Tajikistan	IMOMALI RAKHMONOV	RAKHMONOV	1	RAKHMONOV	3	RAKHMONOV	3		
Tanzania	JAKAYA KIKWETE	KIKWETE	11						

Table A.10: Some results for a set of questions aiming at Presidents of countries (M-T).

		Baseline		GAQA		GA-ASSA		PreGA	
Country	*Prime Minister*	Answer	R	Answer	R	Answer	R	Answer	R
Egypt	AHMED NAZIF	NAZIF	2	NAZIF	1	NAZIF	2		
Estonia	ANDRUS ANSIP	ANSIP	4	ANDRUS	1	ANDRUS ANSIP	1		
Ethiopia	MELES ZENAWI	MELES	1	MELES	1	MELES	1		
Fiji	LAISENIA QARASE	QARASE	10	LAISENIA QARASE	1	LAISENIA QARASE	1		
Finland	MATTI VANHANEN	VANHANEN	1	MATTI	1	VANHANEN	1	VANHANEN	3
Gabon	JEAN EYEGHE NDONG	JEAN	4	JEAN	1	JEAN	1		
Georgia	ZURAB NOGAIDELI	ZURAB	3	ZURAB	1	ZURAB	1	NOGAIDELI	6
Greece	KOSTAS KARAMANLIS	KARAMANLIS	3	KARAMANLIS	3	KARAMANLIS	3	KOSTAS	4
Grenada	KEITH MITCHELL	MITCHELL	4	MITCHELL	1	MITCHELL	1	KEITH MITCHELL	2
Guinea	CELLOU DALEIN DIALLO	DALEIN	23						
Guinea Bissau	ARISTIDES GOMES	GOMES	4	ARISTIDES	1	ARISTIDES GOMES	1	GOMES	3
Guyana	SAM HINDS	HINDS	2	HINDS	1	HINDS	2		
Haiti	GERARD LATORTUE	LATORTUE	3	GERARD	3	GERARD	6		
Hungary	FERENC GYURCSANY	FERENC	2	FERENC	2	FERENC	2	FERENC	2
Iceland	HALLDOR ASGRIMSSON	ASGRIMSSON	3	HALLDOR ASGRIMSSON	1	ASGRIMSSON	1		
Iraq	JAWAD AL-MALIKI	AL	12	AL	2	AL	2		
Israel	EHUD OLMERT	EHUD	3	EHUD	1	EHUD	1	OLMERT	3
Italy	ROMANO PRODI	PRODI	1	ROMANO	4	PRODI	2	PRODI	4
Japan	JUNICHIRO KOIZUMI	KOIZUMI	1	JUNICHIRO	1	JUNICHIRO	1		
Jordan	MAROUF AL-BAKHIT	MAROUF	3	MAROUF AL BAKHIT	1	AL	1	AL	2
Kazakhstan	DANIYAL AKHMETOV	AKHMETOV	2	AKHMETOV	1	AKHMETOV	1	DANIYAL	3
Kyrgyzstan	FELIKS KULOV	KULOV	1	KULOV	1	KULOV	1		
Latvia	AIGARS KALVITIS	AIGARS	2	KALVITIS	4	AIGARS	2		
Lebanon	FOUAD SINIORA	SINIORA	15	SINIORA	3	SINIORA	10		
Lesotho	PAKALITHA MOSISILI	PAKALITHA	2	PAKALITHA MOSISILI	1	MOSISILI	1	MOSISILI	2
Lithuania	ALGIRDAS BRAZAUSKAS	BRAZAUSKAS	2	ALGIRDAS	1	ALGIRDAS	1		
Luxembourg	JEAN-CLAUDE JUNCKER	JUNCKER	1	JEAN	1	JUNCKER	1	CLAUDE JUNCKER	2
Macedonia	VLADO BUCKOVSKI	VLADO	3	VLADO	2	VLADO	2	VLADO	2
Madagascar	JACQUES SYLLA	SYLLA	4	JACQUES SYLLA	2	JACQUES SYLLA	4		
Mali	OUSMANE ISSOUFI MAIGA	OUSMANE	5	OUSMANE	3	OUSMANE	2	OUSMANE ISSOUFI	3
Malta	LAWRENCE GONZI	LAWRENCE	1	GONZI	1	LAWRENCE GONZI	1		
Mauritania	SIDI MOHAMED OULD BOUBACAR	OULD	2	OULD	1	OULD	1	SIDI MOHAMED	2
Mauritius	NAVIN RAMGOOLAM	RAMGOOLAM	2	RAMGOOLAM	1				
Moldova	VASILE TARLEV	TARLEV	6	VASILE	1	VASILE	1		

Table A.11: Some results for a set of questions aiming at Prime Ministers of countries (A-M).

	Baseline		GAQA		GA-ASSA		PreGA	
Country	*Prime Minister*	Answer	R	Answer	R	Answer	R	Answer	R
Morocco	DRISS JETTOU	JETTOU	1	DRISS	3	JETTOU	1	JETTOU	4
Myanmar	SOE WIN	WIN	5			SOE WIN	1	WIN	3
Netherlands	JAN PETER BALKENENDE	BALKENENDE	1	JAN PETER BALKENENDE	1	BALKENENDE	1	BALKENENDE	2
New Zealand	HELEN CLARK	HELEN	1	HELEN	1	CLARK	1	HELEN CLARK	2
Niger	HAMA AMADOU	AMADOU	2	HAMA	1	AMADOU	1	HAMA AMADOU	2
Norway	JENS STOLTENBERG	STOLTENBERG	1	JENS	1	STOLTENBERG	1		
Oman	QABOOS IBN SAID AL BUSAID	AL	1	AL	1	SAID	1	SAID	2
Pakistan	SHAUKAT AZIZ	SHAUKAT	1	SHAUKAT	1	SHAUKAT AZIZ	1		
Palestine	ISMAIL HANIYA	HANIYEH	5	ISMAIL	4	ISMAIL	6		
Papua	SIR MICHAEL SOMARE	SOMARE	1	SIR	4	MICHAEL	4	SOMARE	2
Peru	PEDRO PABLO KUCZYNSKI	KUCZYNSKI	10	PEDRO PABLO	5	KUCZYNSKI	2		
Poland	KAZIMIERZ MARCINKIEWICZ	MARCINKIEWICZ	1	MARCINKIEWICZ	2	MARCINKIEWICZ	1		
Portugal	JOSE SOCRATES	JOSE	1			JOSE	3		
Romania	CALIN POPESCU-TARICEANU	CALIN	6	TARICEANU	4	CALIN	4	TARICEANU	2
Russia	MIKHAIL FRADKOV	MIKHAIL	2	MIKHAIL	1	MIKHAIL	1		
Rwanda	BERNARD MAKUZA	MIKHAIL	14			MAKUZA	2		
Sao Tome and Principe	TOME VERA CRUZ	VERA	15						
Senegal	MACKY SALL	SALL	6	MACKY SALL	1	MACKY SALL	2		
Singapore	LEE HSIEN LOONG	LEE	1	LEE	1	LEE	1	HSIEN	2
Slovakia	MIKULA DZURINDA	DZURINDA	2	DZURINDA	2	DZURINDA	1		
Slovenia	JANEZ JANA	JANEZ	2	JANEZ	1	JANEZ	1		
Solomon Islands	MANASSEH SOGAVARE	SOGAVARE	3	MANASSEH SOGAVARE	1	SOGAVARE SOGAVARE	1	MANASSEH SOGAVARE	2
Somalia	ALI MUHAMMAD GHEDI	ALI	5	ALI	1	ALI	1	ALI MUHAMMAD	2
Swaziland	THEMBA DLAMINI	DLAMINI	2	THEMBA DLAMINI	1	DLAMINI	1		
Sweden	GORAN PERSSON	PERSSON	1	PERSSON	1	PERSSON	1		
Syria	MUHAMMAD NAJI AL-OTARI	NAJI	9	AL	3	AL	2	AL OTARI	2
Tajikistan	AKIL AKILOV	AKILOV	1	AKIL AKILOV	2	AKILOV	4		
Thailand	HAKSIN SHINAWATRA	SHINAWATRA	3	SHINAWATRA	3	SHINAWATRA	2		

Table A.12: Some results for a set of questions aiming at Prime Ministers of countries (M-T).

		Baseline		GAQA		GA-ASSA		PreGA	
Monuments/Cities	*Location*	Answer	R	Answer	R	Answer	R	Answer	R
Mount Rushmore	SOUTH DAKOTA–USA	DAKOTA	8					DAKOTAS BLACK HILLS	8
Mozart Statue	SAINT GILGEN–AUSTRIA	SALZBURG	1	SALZBURG	2	SALZBURG	1	AUSTRIA	1
National War Memorial	NEW ZEALAND	OTTAWA	6	IN OTTAWA	1	OTTAWA	2	OTTAWA	1
National World War II Memorial	WASHINGTON–USA	WASHINGTON	4	WASHINGTON	1	WASHINGTON	1	WASHINGTON	1
Orlandos Column	ORLANDO–FLORIDA	FLORIDA	3	ORLANDO	1	ORLANDO	1	TO ORLANDO	1
Parliament House Flag Pole	CANBERRA–AUSTRALIA	CANBERRA	1	CANBERRA	5	CANBERRA	3	CANBERRA	4
The Pyramids of Giza	EGYPT	EGYPT	3	EGYPT	8	EGYPT	7	EGYPT	3
Sam Houston Statue	HUNTVILLE–TEXAS–USA	HUNTSVILLE	3	HUNTSVILLE	4	HUNTSVILLE	3	HUNTSVILLE	5
Sawyer Point Flying Pigs	CINCINNATI–OHIO–USA	CINCINNATI	3	CINCINNATI	2			CINCINNATI	2
Soviet War Memorial	BERLIN-GERMANY	BERLIN	1	BERLIN	1	BERLIN	1	BERLIN	1
Stanley Park Totem Poles	VANCOUVER–CANADA	CANADA	4	VANCOUVER	1	VANCOUVER	1	VANCOUVER	1
Statue of Liberty	NEW YORK–USA	NEW,YORK	2	NEW YORK	2	NEW YORK	3	NEW YORK HARBOR	4
Sungnyemun Gate	SEOUL–KOREA	SEOUL	2	SEOUL	1	SEOUL	1	DOWNTOWN SEOUL	10
Texas Heroes Monument	GALVESTON–USA	GALVESTON	2						
Tupperware Tower	ORLANDO–FLORIDA–USA	ORLANDO	2					USA	9
Tyler Davidson Fountain	CINCINNATI–OHIO–USA	CINCINNATI	2			CINCINNATI	3	CINCINNATI	2
Uzupis Angel	VILNIUS,LITHUANIA	VILNIUS	1	VILNIUS	1	VILNIUS	1	IN VILNIUS	5
Victor Emmanuel Monument	ROME–ITALY	ROME	3	ROME	5	ROME	2	ROME	2
Wellington Arch	LONDON–WESTMINISTER	LONDON	3	LONDON	1	LONDON	1	LONDON	1
City of Limerick	IRELAND	IRELAND	6	IRELAND	1	IRELAND	1		
County Longford	IRELAND	IRELAND	1	IRELAND	1	IRELAND	1	IRELAND	1
County Louth	IRELAND	IRELAND	1	IRELAND	1	IRELAND	1	IN IRELAND	5
Drogheda	IRELAND	IRELAND	2	IRELAND	1	IRELAND	1	IRELAND	2
County Mayo	IRELAND	IRELAND	1	IRELAND	2	IRELAND	1	IRELAND	1
County Meath	IRELAND	IRELAND	1	IRELAND	1	IRELAND	1	OF IRELAND	1
County Monaghan	IRELAND	IRELAND	1	IRELAND	1	IRELAND	1		
County Offaly	IRELAND	IRELAND	1	IRELAND	1	IRELAND	1		
County Roscommon	IRELAND	IRELAND	3	IRELAND	1	IRELAND	1	IRELAND	1
County Sligo	IRELAND	IRELAND	1	IRELAND	1	IRELAND	1	IRELAND	1
Sligo	IRELAND	IRELAND	1	IRELAND	1	IRELAND	1	IRELAND	1
County Tipperary	IRELAND	IRELAND	1	IRELAND	1	IRELAND	1		
North Tipperary	IRELAND	IRELAND	9						
South Tipperary	IRELAND	IRELAND	8	IRELAND	4	IRELAND	7	IN IRELAND	3
Clonmel	IRELAND	IRELAND	4	IRELAND	4	IRELAND	1		
County Waterford	IRELAND	IRELAND	2	IRELAND	1	IRELAND	1	IRELAND	1
City of Waterford	IRELAND	IRELAND	5			IRELAND	5		
County Westmeath	IRELAND	IRELAND	1	IRELAND	1	IRELAND	1	IRELAND	1
County Wexford	IRELAND	IRELAND	1	IRELAND	1	IRELAND	1	IRELAND	1
Wexford	IRELAND	IRELAND	2	IRELAND	1	IRELAND	1	IRELAND	1
County Wicklow	IRELAND	IRELAND	1	IRELAND	1	IRELAND	1	IRELAND	1

Table A.13: Some results for a set of questions aiming at the LOCATION of monuments and cities.

		Baseline		GAQA		GA-ASSA		PreGA	
Person	*Birth date*	Answer	R	Answer	R	Answer	R	Answer	R
Lance Berkman	10-Feb-1976	1976	5	1976	1	1976	2	1976	4
Brandon Boyd	15-Feb-1976	1976	5	1976	6	1976	4		
Freddie Prinze, Jr.	08-Mar-1976	1976	9	1976	2	1976	1	1976	1
Reese Witherspoon	22-Mar-1976	1976	9	1976	2	1976	2		
Keri Russell	23-Mar-1976	1976	4	1976	3	1976	2	1976	2
Peyton Manning	24-Mar-1976	1976	3	1976	5	1976	2		
Amy Smart	26-Mar-1976	1976	9	1976	2	1976	7	1976	1
Candace Cameron	30-Mar-1976	1976	7	1976	4	1976	3		
Colin Farrell	18-Apr-1976	1976	6	1976	1	1976	2	1976	1
Ruud Van Nistelrooy	20-Apr-1976	1976	5	1976	2	1976	1	1976	2
Fred Savage	09-Jul-1976	1976	6	1976	1	1976	2	1976	2
J.C. Chasez	08-Ago-1976	1976	7	1976	1	1976	1	1976	2
Alicia Silverstone	04-Oct-1976	1976	10	1976	3	1976	1		
Rachel McAdams	07-Oct-1976	1976	6	1976	2	1976	1		
Bob Burnquist	10-Oct-1976	1976	3	1976	3				
Pat Tillman	06-Nov-1976	1976	5	1976	7	1976	1	1976	4
Donovan McNabb	25-Nov-1976	1976	1	1976	4	1976	4	1976	2
Jaleel White	27-Nov-1976	1976	4	1976	3	1976	1	1976	1
Joey Fatone, Jr.	28-Jan-1977	1977	1	1977	1	1977	2		
Shakira	02-Feb-1977	1977	8	1977	10				
David Phoenix Farrell	08-Feb-1977	1977	10	1977	5	1977	4		
Paul Cattermole	07-Mar-1977	1977	5	1977	1	1977	1	1977	1
James Van Der Beek	08-Mar-1977	1977	5	1977	1	1977	1	1977	1
Shannon Miller	10-Mar-1977	1977	5	1977	4			1977	3
Amanda Borden	10-May-1977	1977	4	1977	1	1977	1	1977	1
Samantha Morton	13-May-1977	1977	4	1977	1	1977	2	1977	1
Liv Tyler	01-Jul-1977	1977	10	1977	3	1977	2	1977	3
Edward Furlong	02-Ago-1977	1977	5	1977	3	1977	2	1977	1
Tom Brady	02-Ago-1977	1977	8	1977	2	1977	1		
Jon Heder	26-Oct-1977	1977	2	1977	2	1977	1	1977	2
Oksana Baiul	16-Nov-1977	1977	6					1977	6
Laura Wilkinson	17-Nov-1977	1977	4			1977	1	1977	2
Kerri Strug	19-Nov-1977	1977	7						
Brad Delson	01-Dec-1977	1977	1			1977	2	1977	2
Laila Ali	30-Dec-1977	1977	4						
A.J. McLean	09-Jan-1978	1978	7	1978	7	1978	4		
Ashton Kutcher	07-Feb-1978	1978	8	1978	3	1978	2	1978	1
Jen Frost	22-Feb-1978	1978	5						
Jensen Ackles	01-Mar-1978	1978	2			1978	6		
Kyle Howard	13-Apr-1978	1978	8	1978	1	1978	1		
Kenan Thompson	10-May-1978	1978	2	1978	1	1978	1		
Kanye West	08-Jun-1978	1978	2					1978	2
Shane West	10-Jun-1978	1978	3	1978	3	1978	1	1978	1
Joshua Jackson	11-Jun-1978	1978	5	1978	2	1978	2		
Zo Saldaa	19-Jun-1978	1978	5	1978	5	1978	9	1978	5
Tia and Tamera Mowry	06-Jul-1978	1978	4	1978	3	1978	6		
Louise Brown	25-Jul-1978	1978	5			1978	6		
Kobe Bryant	23-Ago-1978	1978	4	1978	3	1978	4	1978	2
Devon Sawa	07-Sep-1978	1978	7	1978	2	1978	2	1978	2
Benjamin McKenzie	12-Sep-1978	1978	7	1978	7	1978	8		
Ruben Studdard	12-Sep-1978	1978	5	1978	10	1978	8		
Sisqo	09-Nov-1978	1978	3	1978	5	1978	3		
Nelly Furtado	02-Dec-1978	1978	3	1978	3	1978	6	1978	3
Katie Holmes	18-Dec-1978	1978	2	1978	2	1978	3		

Table A.14: Some results for a set of questions aiming at a DATE as answer (1976-1978).

		Baseline		GAQA		GA-ASSA		PreGA	
Monuments/Cities	*Location*	Answer	R	Answer	R	Answer	R	Answer	R
Aaliyah	16-Jan-1979	1979	9	1979	2	1979	3	1979	2
Rob Bourdon	20-Jan-1979	1979	4			1979	11		
Tatyana Ali	24-Jan-1979	1979	2						
Andrew Keegan	29-Jan-1979	1979	8	1979	2	1979	1	1979	4
Josh Keaton	08-Feb-1979	1979	9			1979	8		
Ziyi Zhang	09-Feb-1979	1979	5	1979	4	1979	4	1979	2
Brandy	11-Feb-1979	1979	3			1979	2		
Jennifer Love Hewitt	21-Feb-1979	1979	7	1979	2	1979	6	1979	2
Benji and Joel Madden	11-Mar-1979	1979	7	1979	4	1979			
Adam Levine	18-Mar-1979	1979	1			1979	2	1979	8
Heath Ledger	04-Apr-1979	1979	4	1979	2	1979	1		
Keshia Knight Pulliam	09-Apr-1979	1979	3	1979	2	1979	2	1979	2
Claire Danes	12-Apr-1979	1979	8	1979	1	1979	3	1979	2
Kate Hudson	19-Apr-1979	1979	7	1979	2	1979	2		
Daniel Johns	22-Apr-1979	1979	3	1979	1	1979	3		
Lance Bass	04-May-1979	1979	6			1979	3		
Pierre Bouvier	09-May-1979	1979	1	1979	1	1979	1	1979	1
Jesse Bradford	28-May-1979	1979	2	1979	2	1979	3		
Shane Filan	05-Jul-1979	1979	1						
Pink	08-Sep-1979	1979	10	1979	1	1979	2		
Ariana Richards	11-Sep-1979	1979	3	1979	1	1979	2		
Erik-Michael Estrada	23-Sep-1979	1979	8			1979	10		
Mya	10-Oct-1979	1979	9			1979	2		
Stacy Keibler	14-Oct-1979	1979	4	14 1979	5	1979	9	1979	3
Ben Gillies	24-Oct-1979	1979	2	1979	1	1979	1	1979	1
Chris Joannou	10-Nov-1979	1979	3	1979	1	1979	1	1979	1
Rider Strong	11-Dec-1979	1979	3	1979	4	1979	7	1979	2
Michael Owen	14-Dec-1979	1979	3	1979	5	1979	2	IN 1979	8
Adam Brody	15-Dec-1979	1979	4	1979	10	1979	1	1979	4
Kristanna Loken	19-Dec-1979	1979	12	1979	1	1979	1	1979	1
Jenson Button	19-Jan-1980	1980	7	1980	2	1980	1	1980	1
Nick Carter	28-Jan-1980	1980	6						
Matt Lawrence	11-Feb-1980	1980	6			1980	3		
Chelsea Clinton	27-Feb-1980	1980	8	1980	2	1980	3		
Chingy	09-Mar-1980	1980	13	1980	1	1980	1		
Venus Williams	17-Jun-1980	1980	9	1980	1	1980	1	1980	1
Eric Stretch	22-Jun-1980	1980	25						
Michelle Kwan	07-Jul-1980	1980	9	1980	2	1980	3		
Dominique Swain	12-Ago-1980	1980	2	1980	2	1980	2		
Vanessa Carlton	16-Ago-1980	1980	3						
Macaulay Culkin	26-Ago-1980	1980	7	1980	2	1980	2		
Yao Ming	12-Sep-1980	1980	10	1980	3	1980	3		
Ben Savage	13-Sep-1980	1980	7			1980	4		
Martina Hingis	30-Sep-1980	1980	5	1980	4	1980	1	1980	2
Paul Thomas	05-Oct-1980	1980	20			1980	8		
Ashanti	13-Oct-1980	1980	9	1980	1	1980	2	IN 1980	2
Nick Cannon	17-Oct-1980	1980	3	1980	2	1980	1		
Christina Aguilera	18-Dec-1980	1980	11	1980	2	1980	2	1980	1
Jake Gyllenhaal	19-Dec-1980	1980	6	1980	1	1980	1	1980	2

Table A.15: Some results for a set of questions aiming at a DATE as answer (1979-1980).