IceQA: Developing a question answering system for Icelandic[*]

Ólafur Páll Geirsson

Abstract

In this report, we summarize the first attempt made to develop an open source question answering (QA) system for the Icelandic language. We present IceQA, a QA system built on top of the QANUS framework.

A couple of textprocessing modules are missing for the Icelandic language, and thus, IceQA is limited to answering questions in three categories: persons, locations and years. From an evaluation run over a set of 100 questions in these given categories, it achieves a strict accuracy score of %41 and relaxed accuracy score of %51. We look at the development process of IceQA and future prospects in question answering for the Icelandic language.

1 Introduction

With the gradual increase in amount of information being stored in natural language text, the need for an automated system to extract that information becomes more apparent. Such a system would allow the user to ask a question using natural language and receive an accurate answer both quickly and reliably. Current search engines such as Google and Bing may be successful at directing users towards answers given a query, but they do not put together a complete answer for the user. Instead, they return a ranked list of documents. Question answering (QA) systems in contrast deliver exact *answers*.

Inspiration for this research project came from the fact that much research has been put into QA over the last decade along with a trend towards an open advancement of question answering [3]. Being both an interesting interdisciplinary research area and having practical application, question answering has gained some public attention in the past years. The best known example of a QA system could be IBM Watson which won a Jeopardy! competition live on television. Other well known examples include Apple's Siri and Google Now. The introduction of newly available open source QA frameworks has facilitated for rapid development and evaluation of QA systems.

[*]This project was supported by grant no. 1320360091 from the Icelandic Student Innovation Fund.

We sought out to develop a prototype of an open-domain question answering system for the Icelandic language. Despite the amount of research put into QA in recent years, no such system has been developed for the Icelandic language. Following the naming conventions of existing Icelandic text processing tools, we named the system `IceQA`. To find a platform for developing IceQA we researched the current availability of open source QA frameworks. For working with Icelandic text we made use of IceNLP[7], an open source text-processing toolkit for the Icelandic language.

We hypothesised that it would be possible to develop a prototype of an open domain question answering system for the Icelandic language within the period of 3 months that would achieve some level of accuracy. We were eager to find out what accuracy was achievable, especially considering the progress [10] that has been made in language technology for the Icelandic language over the last 13 year. On the other hand, we were also willing to embrace the fact that there might be missing some essential modules to achieve an accuracy level comparable to state-of-the-art QA systems developed for other languages.

2 Background

The development of IceQA was split into four phases. The first phase for reading sources and preparation of information base, the second for evaluating existing open source QA frameworks, the third for integrating IceNLP components into such a framework, and the last phase for evaluating the outcome. In this section, we'll cover the terminology used in QA and the approach taken to develop IceQA.

2.1 Terminology

To understand QA it is necessary to look at the most common terminology used in this field.

Broad domain Broad domain refers to the breadth of topics for which a QA system can achieve acceptable level of accuracy. When designing a QA system it is necessary to consider in which domains of knowledge ranging from history and geography to science and to literature, the system should be able answer questions from.

Answer type taxonomy A set of types or classes which an answer to a given question can be classified into. The most widely used taxonomy in QA is a hierarchical two-layered taxonomy introduced by Li and Roth [5], including 6 coarse classes (abbreviation, entity, description, human, location and numeric value) and 50 fine classes.

Factoid question A question to which a correct answer is a single non-debatable fact or statement, in particular a named entity. An example of a factoid question would be 'Who is the president of Iceland?', where the correct answer would be the name Ólafur Ragnar Grímsson. An example of a non-factoid question is 'When do you get old?', to such a question a *correct* answer may not necessarily exist or vary depending on whom the question is directed to. Answers to factoid questions rarely exceed 5 words and are most commonly 1-2 words.

Named Entity Recognition (NER) The combined task of detecting and classifying a named entity in a given text. A named entity can be anything which is referred to with a proper name. The types of classes used in NER depends on a predefined taxonomy used for each context. Examples of classes used in traditional NER systems are persons, organizations and locations.

Question classification The task of assigning an answer type to a question. In QA, often the first step towards retrieving an answer to a given question is to perform question classification.

Query language complexity Query Language Complexity refers to the ambiguity and structural complexity of the questions in a QA problem. It relates to the difficulty of extracting the intended meaning of the question from its linguistic expression.

User interaction / Usability The degree of user interaction required to run the QA system, i.e. ask a question and receive an answer. For a QA system to be usable by the public it would be necessary to, for example, implement a web browser interface or providing a conversational agent[1]. Very likely however in the case of IceQA, we would only be limited to a command line interface.

2.2 Defining IceQA

By understanding common question answering terminology, it is possible to define what type of a question answering system IceQA should be.

To begin with, the broad domain was determined. For testing IceQA a collection of all articles from the Icelandic Web of Science (is. Vísindavefur Háskóla Íslands) was used. This was considered as a good corpus for a few

[1]Ex. Apple's Siri

reasons. It's large enough for development purposes, it covers a broad range of topics while still being credible enough to produce reliable answers.

Next, an answer type taxonomy was defined. To do so we looked at existing tools for the Icelandic language to classify questions as well as the taxonomy used by existing QA system for other languages. Many successful QA system today use the above mentioned taxonomy defined by Li and Roth. However, this two-layered taxonomy has been designed to cover the types of answers from the TREC[2] track. The TREC tracks are used as a platform to evaluate separate QA systems which often have been under development for many years, in contrast with a 2 month prototype implementation of IceQA. To add on top of that, for the Icelandic language there does not exist a question classification module. This does dramatically limit our capabilities to produce a reliable answer type from questions presented in Icelandic. Keeping this in mind and our goal of developing a prototype we settled on a simpler answer ontology. The answer type taxonomy would only contain three answer types: person, location and year.

Finally, for user interaction we would settle with the interface provided by the framework on which we build IceQA. The only requirement set to IceQA's usability is a standard way of evaluating its accuracy.

2.3 Platform selection

Writing IceQA from scratch would be an overwhelming task, instead we decided to use an existing QA framework on top of which we would build IceQA. Some open source QA framworks exists, we had two particular systems in mind. Firstly, OpenEphyra [11] developed at Carnegie Mellon and secondly QANUS [9] developed at the National University of Singapore. We researched both systems, beginning with OpenEphyra.

OpenEpyra is a widely used open source QA framework. When IBM set out to develop their IBM Watson, they used a slight modification of OpenEphyra as a baseline comparison [2, p. 66]. Running a provided example of the system on our computer however did not work 'out of the box'. After some online researching and digging into the documentation and code-base for a couple of days no solution to the problem still appeared[3]. The decision was made to look at other alternatives instead.

QANUS is built with extensibility in mind and as such, it turns out to be an ideal framework for developing IceQA. The framework is well documented, written in Java like IceNLP and the provided reference implementation of a QA system worked 'out of the box'. The framework also does not put any limitation on what kind of text processing modules are used making it pos-

[2]The Text REtrieval Conference (TREC) is an on-going series of workshops focusing on a list of different information retrieval research areas.

[3]In hindsight, this may likely may have due to the limited experience of the researcher in Java development.

sible to integrate IceNLP into the system. Neither does the framework make any restriction on the input data format, making it possible to incorporate a different kind of corpus than AQUAINT-2, the one provided from the reference implementation. Further implementation details of QANUS will be explained in more detail in Section 3, IceQA Implementation.

Two other QA frameworks were also looked at, ARANEA and Qanda by MITRE, before a decision was made to settle on QANUS. ARANEA is a system which extracts answers from the Web using two different techniques: knowledge annotation and knowledge mining [6]. ARANEA is written in C making it harder to integrate with IceNLP modules. Also, ARANEA was rather developed to test a particular approach to QA instead of as an all-round QA platform. This made ARANEA a less suitable option for IceQA than QANUS. Qanda is a QA system developed by MITRE[4] which has had some success in the TREC tracks [1]. Currently however only one module of Qanda has been made available for download making it an unattractive platform for developing IceQA.

Having researched frameworks options for developing IceQA, the decision was eventually made to settle on QANUS. This would not necessarily mean that QANUS is the final choice for developing IceQA. The experience gained from developing with QANUS however will make us better able to make an informed decision later on to switch platforms, if necessary. Keeping this in mind the decision to resolve on a particular platform would not necessarily have to be a game changing one.

3 IceQA Implementation

QANUS adopts a pipelined approach to QA[8], dividing the QA task into four sub-tasks:

1. Information base preparation
2. question processing
3. answer retrieval
4. evaluation

The overview of the system can be seen in Figure 1 (Borrowed from[8]). We will look at all four stages individually and then explain the modifications made during development of IceQA.

3.1 Information source preparation

Information base preparation (IBP) is the task of preparing a corpus to be used as an information base for the answer retrieval stage. This can include performing any necessary text-processing on the corpus such as part of speech tagging, named entity recognition and lemmatisation. The output of this stage in QANUS is a Lucene[5] index, ready for use in the answer retrieval

[4]A US federally funded R&D center

[5]An open source text search engine written in Java, https://lucene.apache.org/

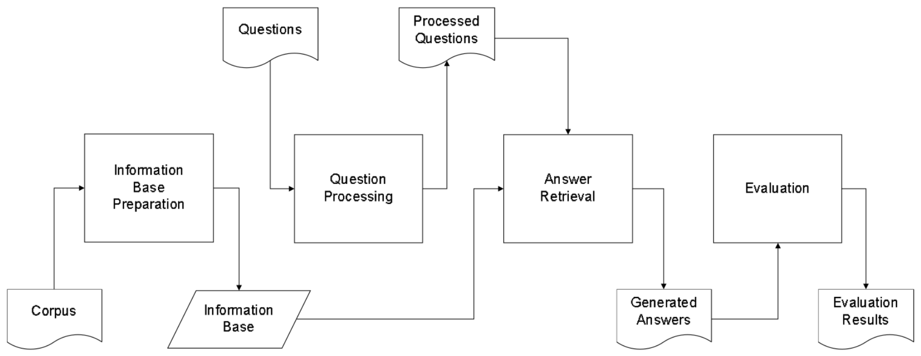

Figure 1: Overview of QANUS

stage. The Lucene index contains the corpus including annotations processed with the IceNLP toolkit.

In this stage, little modification was required for IceQA. Some Icelandic text-processing wrappers were developed in order to align the API[6] of IceNLP with the QANUS framework. The above mentioned text-processing tools are already available in IceNLP. No customizations were required.

3.2 Question processing

Question processing (QP) is, similarly as the IBP stage, the task of processing a question to be used later in the answer retrieval stage. This includes performing text-processing such as part of speech tagging and lemmatisation for the question but also question classification and possibly query expansion. The output of the question processing stage is an XML file including the question string and question annotations.

For IceQA, the problem with this stage is that no question classification tool for the Icelandic language exists. Much research has been put into the field of question classification and for many languages there are already existing question classification modules which achieve high accuracy. Thus, developing one from scratch for the Icelandic language, however, is a project on its own and outside the scope of this summer project. The solution instead was to rapidly develop a simple rule-based question classifier that would be able to classify the questions fitting the defined answer taxonomy of IceQA, as explained before in Section 2.2.

Another compromise that had to be made was on query expansion. In many QA systems, a question is translated into multiple queries including synonyms of words used in the original question. This can in some cases lead to a higher performing QA system [4]. To perform query expansion it

[6]Application programming interface.

is necessary to have thesaurus API which at the moment is either not freely available or easily accessible for the Icelandic language. Developing such a module would be possible but as with the question classification, still outside the scope of this summer project. The decision was made to skip performing query expansion. This would still suffice to run an evaluation of IceQA and see how accurately IceQA could perform in answer retrieval, as a baseline score.

3.3 Answer retrieval

Answer retrieval (AR) is the task of finding a correct answer to a given question. As one may suspect, implementing this stage is the most complicated part of developing a QA system. The input to this stage is twofold, a Lucene index including the processed corpus from the IBP stage and a list of annotated questions from the QP stage. In QA, two main classes of algorithms have been applied to the answer-extraction task[4], one based on answer-type pattern extraction and one based on N-gram tiling. In QANUS, the reference QA system implements the former one, answer-type pattern extraction. The output from the AR stage is a list of answers to the given questions, including source passages (sentences from where the answers were extracted) and a link to the document containing the answer text.

For IceQA, a couple of modification were made to the reference implementation that came shipped with QANUS. Mainly, the way how different types of questions were delegated to answering modules was made different. Originally, all answer extraction took place in a single class `FeatureScoringStrategy` with different methods handling different types of answer types. Instead, a module based approach was adopted. A `IAnswerer` interface was designed to deal with this problem. Also, there was no convenient interface for handling annotated text withing the AR stage. A `Passage` class was implemented and integrated into the QANUS framework to treat this nuisance. Details on how answer extraction is implemented in IceQA can be found in Section 3.5.

3.4 Evaluation

The evaluation stage manages comparison of obtained answers with expected answers, outputting a report with the accuracy of the QA system. Accuracy is defined as the ratio between the number of correctly generated results to the total number of factoid questions sent to the QA system. The results from the evaluation stage will be discussed in the Section 4.

For IceQA, no modifications were required to run the evaluation stage.

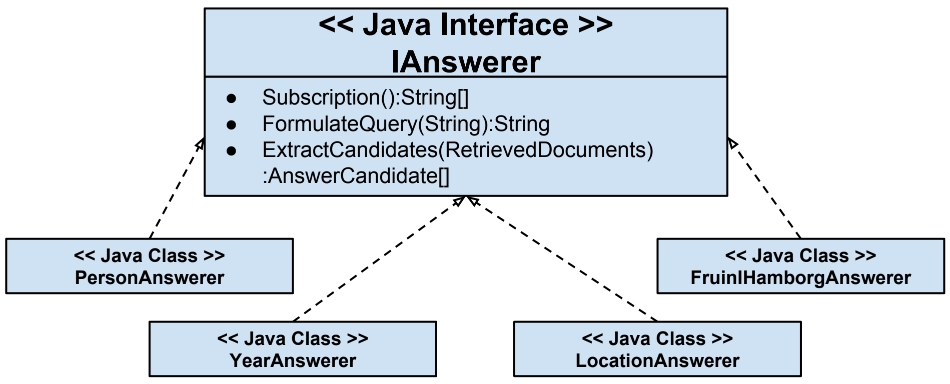

Figure 2: IceQA IAnswerer interface

3.5 Answer extraction

Getting the answering extraction algorithm right is critical in order to achieve a high level of accuracy. To offer a clean way to answer any question of arbitrary type an `IAnswerer` interface was developed and incorporated into IceQA.

Answer extraction in IceQA is split into a couple of steps including IAnswerer selection, query formulation, candidate extraction and candidate selection. The IAnswerer and candidate selection phases are performed by the `FeatureScoringStrategy` class while query formulation and candidate extraction take place inside the answering modules. We look at both the `IAnswerer` interface and then the `FeatureScoringStrategy` class to see in detail how answer extraction is implemented in IceQA.

3.6 IAnswerer interface

A Java class implementing the `IAnswerer` interface is called an answering module. Such a module is designed to answer a specific type of questions to which it ‘subscribes’ to. When IceQA is confronted with such a question, the answering module helps out with two parts of the answer extraction algorithm. Firstly, it is used to formulate an appropriate query into the information base. Secondly, after the formulated query has been used to query the information base the answering module is used to extract potential answer candidates. An overview of the `IAnswerer` interface can be seen in Figure 2. A dotted arrow line from a Java class pointing into the `IAnswerer` interface denotes that that class implements the `IAnswerer` interface. By implementing the `IAnswerer` interface, an answering module ‘subscribes’ to questions it intends to answer.

The `IAnswerer` interface is useful. For example, a person answering module (e.g. `PersonAnswerer`) would subscribe to questions which have the answer type PERSON while, a year answering module (e.g. YearAnswerer) would subscribe to questions having the answer type YEAR, and etc. Subscriptions are not only limited to answer types, but also include the question text. It is therefore also possible to develop an answering module that subscribes to questions containing a particular phrase or named entity. For example, it would be possible to make a Valur[7] answering module which could answer any questions related to the football club. Another example would be a 'Hvað keyptir þú fyrir peninginn sem frúin í Hamborg gaf þér?'[8] answering module for an interactive game with a conversational agent.

3.7 FeatureScoringStrategy class

The `FeatureScoringStrategy` class is the heart of answer extraction in IceQA. Inside the `getAnswerForQuestion` method the algorithm to extract an answer to a given question is performed. The algorithm used by IceQA for this part can be seen in Algorithm 1.

Algorithm 1: Answer extraction in `FeatureScoringStrategy`

```
Input: Question $q$, Information base $I$ ;
Output: Answer $a$ to question $q$ ;
begin
    Let $max$ be negative infinity ;
    Let $M$ be the set of answer modules having subscribed to $q$ ;
    for $m \in M$ do
        Let $q'$ be the formulated query by $m$ given $q$ ;
        Let $D$ be retrieved documents by $I$ given $q'$ ;
        for $d \in D$ do
            Let $P$ be the set of passages returned by querying $q'$ into $d$ ;
            Let $C_P$ be the set of candidates extracted by $m$ given $P$ ;
            /* A candidate is composed of an answer, source
               passage and document                          */
            for $c \in C_P$ do
                Let $s$ be the sum of heuristic scores given $c$ and $q'$ ;
                if $s > max$ then
                    Let $a$ be $c$
    return $a$
```

Currently, for IceQA the score calculated in the innermost loop is performed using five heuristics scores: paragraph coverage score, sentence coverage score, proximity score, document score and repeated term score. A description of each of them can be read as follows:

[7]A popular football club in Reykjavik.
[8]A traditional Icelandic question game.

Paragraph coverage score The count of the occurrences of the search terms within a given passage divided over the number of words in the search string. For example if 3 out of 4 words in the search query are contained in the source passage paragraph, this score would be 0.75.

Sentence coverage score The same heuristic as coverage score except, the source passage is now limited to the sentence containing the candidate answer, not the whole paragraph. In the previous example, we can imagine that only 2 out of the 4 words in the query are contained in the string resulting in a sentence coverage score of 0.5. For good answer candidates, the sentence coverage score and paragraph coverage score would be the same.

Proximity score The distance between the occurrences of the answer candidate and query within the source passage. If the answer candidate can be found inside a passage containing the formulated query this score is high, if it is located far away in from the query term this score is low.

Sentence score The index of the source document returned by the text search engine. If the candidate answer is inside the highest ranked document this score is 1, if the candidate answer is inside the lowest ranked document this score is 0.

Repeated term score Penalise answer candidates containing repeated words from the formulated query. If the answer contains words from the formulated query it scores abnormally high in the other heuristics, this score is to weigh against that.

4 Evaluation

QANUS comes shipped with an evaluation module. In order to use it a set of gold-standard questions and answers needed to be collected.

4.1 Question restrictions

To evaluate the performance of IceQA a set of questions had to be collected. Given the broad domain of IceQA and defined answer type taxonomy some restrictions had to be made on what kind of questions were allowed to be used for evaluation. The restrictions to the questions were set as the four following rules:

1. The question must be in Icelandic.
2. The answer to the question has to be one of the types: person name, location or year.
3. The answer to the question must be found inside an article from the Icelandic Web of Science.
4. The answer to the question can not be a list of person names, locations or years.

A group of seven computer science students working at Reykjavik University over the summer volunteered to compose new questions admitting to these rules. Instructions to the participating volunteers were communicated through the web-page http://scriptogr.am/olafurpg/post/IceQA (page in Icelandic). Answers were submitted through a Google form with three text fields. The first field for the question, the second field for the correct answer and the third field for the URL to the source article. Please follow the provided link for more details on the instructions layout.

4.1.1 Questions collected

A total of 100 questions were collected, which was considered as sufficient to develop the prototype of IceQA and perform evaluation. A subset of the collected questions was used during the development of IceQA. This subset, often referred to as a ‘dev’ set, contained 21 questions: 7 with the answer type year, 7 with the answer type person and 7 with the answer type location.

4.2 Results

The evaluation metric used is an accuracy score calculated as the ratio of correctly answered question to the total number of questions, more formally

$$\text{Accuracy score} = \frac{\text{No. of correctly answered questions}}{\text{Total number of questions}}$$

		Strict accuracy		Relaxed accuracy	
Answer type	No. of questions	Correct no.	%	Correct no.	%
Location	30	6	20	14	46.7
Person name	27	8	29.6	16	59.3
Year	43	27	61.4	27	61.4
Total	100	41	41	57	57

Table 1: IceQA Accuracy

Two accuracy scores were calculated, one 'strict' and one 'relaxed'. The strict score was obtained directly by running the evaluation module while the relaxed score was manually calculated by comparing obtained answers with the expected answers. The relaxed score might be considered as a better representation of the actual performance since the strict evaluation only performs plain string comparison. This may lead to a correct answers being evaluated as wrong if for example the obtained answer is in a different case than the expected answer (e.g. obtained 'Íslandi' but expected 'Ísland') or some non-critical details are either missing or additionally present (e.g. obtained 'Reykholti' and 'Christian Goldbach' but expected 'Reykholti í Breiðafirði' and 'Goldbach').

The results from the evaluation stage can be seen in Table 1. As suspected, the relaxed accuracy is higher than the strict one. Not surprisingly either, the relaxed score is only higher for the location and person name answer types. It does not affect the year answer type category.

At the beginning of this project, we hypothesised that it would be possible to develop an open domain question answering system within the period of three months that would achieve some level of accuracy. Looking at the results we may be safe to say that our hypothesis holds true. The top system in the TREC 2007 QA track LymbaPA07 and the tenth-placed system QUANTA achieved accuracy scores of 70.6% and 20.6% respectively [9]. The reference implementation of a QA system that comes shipped with QANUS achieves an accuracy of 11.9%. Comparing these results, one could come to the conclusion that IceQA performs exceptionally well. However, such a comparison is unfair against the other QA systems since they do not make the same restrictions on the types of allowed questions as IceQA does. Nonetheless, achieving over a 50% relaxed accuracy score is certainly a promising beginning.

5 Discussions

From a research perspective, the open-domain question answering problem is attractive as it is one of the most challenging problems in the field of computer science and artificial intelligence, requiring a combination of techniques from multiple interdisciplinary fields including information retrieval, knowledge representation and reasoning, natural language processing and machine learning. From a societal perspective, the open-domain is also attractive as it has the potential to revamp the way humans interact with computers. This motivated us to develop a question answering system, IceQA, for the Icelandic language. We hypothesised in the beginning that it would be possible for IceQA to achieve some level of accuracy on a run of open domain question answering.

We were able confirm our hypothesis by making IceQA achieve a very promising accuracy score over a limited domain. Starting out developing IceQA, we were confronted with the problem that a question classification tool for the Icelandic language does not exist. This forced us to make a compromise on the types of questions that would be presented to IceQA for evaluation. Only questions that have an answer type of name, location or year were tested. For these answer types, a simple rule-based question classification module had to be developed. This allowed us to run an evaluation of IceQA, obtaining a strict accuracy score of 41% and a relaxed accuracy score of 57%. In comparison with existing QA systems for other languages this score can be considered as relatively good. However, such QA system do not put the same restrictions on the allowed types of questions as IceQA does.

The source to IceQA is freely available for download from Bitbucket, https://bitbucket.org/olafurpg/iceqa. Also, a web interface to demonstrate the capabilities of IceQA is under development but, unfortunately, is not ready yet to be included in this report.

Question answering for the Icelandic language has a long way to go. Primarily, a question classification module needs to be developed before a truly open-domain question answering system becomes a reality. In addition, some work needs to be done in improving existing tools for question answering to attain a higher accuracy score. This in particular refers to the Icelandic named entity recognition module, IceNER. Moreover, further work can be put into the answer extraction algorithm currently used by IceQA. Nevertheless, it is safe to say that open-domain question answering is possible for the Icelandic language just as for any other language. Getting there, however, will require increased amounts of research.

References

[1] John D. Burger, Lisa Ferro, Warren Greiff, John Henderson, Marc Light, Scott Mardis, and Alex Morgan. MITRE's qanda at TREC-11. Technical report, DTIC Document, 2006.

[2] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A. Kalyanpur, Adam Lally, J. William Murdock, Eric

Nyberg, and John Prager. Building watson: An overview of the DeepQA project. *AI magazine*, 31(3):59–79, 2010.

[3] David Ferrucci, Eric Nyberg, James Allan, Ken Barker, Eric Brown, Jennifer Chu-Carroll, Arthur Ciccolo, Pablo Duboue, James Fan, and David Gondek. Towards the open advancement of question answering systems. Technical report, IBM Research Report RC24789, 2009.

[4] Dan Jurafsky, James H. Martin, Andrew Kehler, Keith Vander Linden, and Nigel Ward. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, volume 2. MIT Press, 2000.

[5] Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[6] Jimmy Lin and Boris Katz. Question answering from the web using knowledge annotation and knowledge mining techniques. In *Proceedings of the twelfth international conference on Information and knowledge management*, pages 116–123, 2003.

[7] Hrafn Loftsson and Eiríkur Rögnvaldsson. Icenlp: A natural language processing toolkit for icelandic. In *INTERSPEECH*, pages 1533–1536, 2007.

[8] Jun-Ping Ng. QANUS, a open-source question answering framework, January 2010.

[9] Jun-Ping Ng and Min-Yen Kan. *QANUS: An Open-source Question-Answering Platform.* 2010.

[10] Eiríkur Rögnvaldsson, Hrafn Loftsson, Kristín Bjarnadóttir, Sigrún Helgadóttir, Matthew Whelpton, Anna Björk Nikulásdóttir, and Anton Karl Ingason. Icelandic language resources and technology: Status and prospects. 2009.

[11] Nico Schlaefer, Petra Gieselmann, Thomas Schaaf, and Alex Waibel. A pattern learning approach to question answering within the ephyra framework. In *Text, Speech and Dialogue*, pages 687–694, 2006.