Using IceNLP in Computer-assisted language learning applications

Hlynur Sigurþórsson
Pétur Sigurðsson

Útdráttur

Í þessari skýrslu skoðum við möguleikann á því að nota íslenskt málvinnslutól - *IceNLP* - í kennsluhugbúnaði. Þetta verður gert með því að útfæra heimasíðu sem inniheldur safn af gagnvirkum málfræðiæfingum og mæla viðbrögð notenda og nákvæmni *IceNLP* við greiningu setninga sem þeir nota. Við munum einnig lýsa útfærslu á málfræðispili, sem við köllum *Málflækjan* en það notast við *IceNLP* málvinnslutólin við greiningu á aðgerðum spilenda í leiknum.

Abstract

In this report we will discuss the possibility of using the *IceNLP (Icelandic natural language processing)* toolkit in *ICALL (Intelligent computer-assisted language learning)* applications. This will be done by implementing a website that contains a family of dynamic grammar exercises and measure users feedback and accuracy of *IceNLP*. Furthermore, we will introduce our grammar boardgame *Málflækjan* which uses *IceNLP* to analyse users actions in the game.

1 Introduction

CALL (Computer assisted language learning) is a form of a learning process where a learner uses a computer to improve, help, or accelerate the process of learning a new foreign language or to improve grammar in his native language. The idea of *CALL* is not new. According to (Delcloque, 2000) the origin and development of *CALL* can be traced back to the 1960's where experiments with tutoring and linguistic applications where made on large mainframes. One example of such pioneering research was the *PLATO* project which was a generalized computer-assisted instruction system built by the University of Illinois in collaboration with *CDC (Control Data Corporation)*. With this system teachers were able to distribute programmable learning material to students over a network and instruct them through *PLATO* terminals. The *PLATO* project established some of the on-line concepts that are well known on the Internet today, such as forums, message boards, online exams, emails, chat rooms, picture languages, instant messaging, remote screen sharing and multi-player games. For a more detailed description of the *PLATO* project, see (Stifle, 1971).

How is the learning process carried out in *CALL* and how is the computer used in the process? Despite old history and many papers related to the topic, it is still difficult to answer that question. In the book (Beatty, 2003) the author defines *CALL* in the following way:

> "Definition of *CALL* that accommodates the changing nature is any process that in which a learner uses a computer and, as a result, improves his or her language."

This definition of *CALL* and of its usage is very abstract, where it does not state anything about how the learner uses the computer or what pedagogical theories are used in the teaching process. *CALL* applications are therefore any software application that can replace or be used with other learning material in language learning or as communications device for teachers to instruct students.

NLP (Natural Language Processing) is a field of *Computer Science*, that overlaps with the field of *Computational Linguistics* and *Artificial Intelligence*, where the main goal is to create algorithms that can process, understand and work with natural languages and to augment the communication between human users and computers.

Integration of *NLP* into *CALL* applications have been done with great success for languages such as English, German, French and Japanese and many such applications are in a daily use, both in classrooms as teaching materials and over the Internet for distance learning. By integrating *NLP* into *CALL* applications we can improve the feedback to users, create more dynamic and interesting applications where assignments can be created without predefined solutions, or do automatic student modeling. Such extensions are usually called *ICALL (Intelligent computer-assisted language learning)* or *NLP-CALL*.[1].

Above mentioned languages all have one thing in common – they all possess successful *NLP* tools with high accuracy and large corpora that describes the language – which is a precondition for the integration.

Natural languages are hard to work with due to grammar complexity and ambiguity. Therefore, a perfect accuracy in *NLP* tools are rarely seen. Then there is always the question of how high accuracy *NLP* tools must achieve for industrial exploitation? We consider that 95% accuracy is the minimal accuracy for *NLP* tools to be integrated into *CALL* applications.

Hrafn Loftsson, one of the supervisors in this project, and Eiríkur Rögnvaldsson, have created an *NLP* toolkit for Icelandic called *IceNLP* (Loftsson & Rögnvaldsson, 2007b). This toolkit includes a tokenizer, a morphological analyser called *IceMorphy*, a POS *(part-of-speech)* tagger called *IceTagger* and a parser called *IceParser*.

In this report, we will do an evaluation on *IceNLP* and address the question "Is *IceNLP* accurate enough to be used in *ICALL* applications?" by creating a simple *ICALL* survey website and use it to collect more realistic data from possible users of such system and measure the accuracy of *IceNLP* with that data. We will also try to evaluate the trust that users have on such application and how visual errors in *ICALL* applications affects their trust.

Furthermore, we will describe the development of an *ICALL* multiplayer cardgame, which we call *Málflækjan*,

[1] Throughout this text we will use the term ICALL

that we created for this project and a configurable linguistic framework, which we call *ISLingustic*, that was used both in the website and in the game.

The rest of this report will be structured as follows: In section 2, we will discuss the research that has been done in the field of *NLP* with emphasis on Icelandic and introduce the *IceNLP* toolkit and its components and describe few existing *ICALL* applications. In section 3, we will describe our configurable framework and its components and our *ICALL* web page and our *ICALL* multiplayer game. In section 4, we will analysed the data that we got from the survey. We will then finish by discussing our results from the data analysis.

2 Related work

Before we began this project we did a research on existing Icelandic *CALL* applications and their usages of *NLP*, but as we expected none of them took any advantage of *NLP* technology at all. The reasons for that is that *Language Technology* is relatively young discipline in Iceland and was almost non-existing few years ago and high accurate Icelandic *NLP* tools did not exist until recently. (Rögnvaldsson, 2003).

During our research we found out that most Icelandic *CALL* application were "fill in the gaps" gramma exercises with predefined solutions. The problem with such applications are that they only give educational value at first, then over period of time the users starts remembering the solution to the exercises and the educational value changes from language learning to testing the users memory.

The initial idea of this project was to create an *ICALL* website, that could be used in language learning in primary education, were *IceNLP* would be used to create grammar exercises on the fly from users input or from a randomly selected text from some source. Such application would have much more usability then application with pre-defined solutions since users will not remember the solutions and the educational value would not diminish.

We found a similar websites for English (see 2.4 - WERTI) that created grammar exercises from randomly selected news articles, but After 10 minutes of using it we both got bored and felt isolated while solving the exercises.

We believe that the most fundamental requirements when creating a teaching material that could be used in a classroom, or in individualized learning on a website, is that the material is correct so its purpose helps the learner in the learning process. It is also important that the learner trusts the materials to be correct. Lack of trust will decrease interest in studying what the material is representing and therefore the usage of the system. With higher trust rate it is more likely that the system will be used as a learning material in language learning. It is also fundamental that the material is engaging and enjoyable.

Therefore we changed the initial idea of the project into creating an *ICALL* multiplayer cardgame where users collaborated in solving grammar exercises in engaging social environment.

In this section, we will discuss *NLP* in general and the research that has been done in that field with emphasis on Icelandic. We will then describe several successful *ICALL* applications that we found during our research.

2.1 NLP

For creating an *ICALL* application we need tools that know how to analyze sentences and words in terms of morphological informations and grammatical structures. These tools exist and are part of *NLP*. We will now discuss the main sub-tasks in *NLP* which we used in the work.

2.1.1 Text segmentation / Tokenization

Text segmentation is the task of dividing text into a meaningful units that are usually called lexemes. Before any analysis can be done on a sentence, we need to find where each sentence boundaries are so we can break them up into correct lexemes. This task is usually done by combining together lookup tables (for abbreviations, names, etc.) and regular expressions. The unit that deals with this task is usually called a tokenizer.

2.1.2 POS tagging

POS *(Part-of-speech)* tagging is the task of annotating each word in a text with its correct word class and morphological information based on both its definition, as well as its context. The string used as a label is called a tag. This task is one of the most fundamental in *NLP* since other *NLP* tasks, such as parsing, use the results of the tagging to solve their problems.

Combining existing taggers, using simple voting to select the tag to be distributed, is a well known method to increase tagging accuracy (Loftsson, 2006). Running many taggers for the same text can be very expensive, and if using one POS tagger is sufficient enough, then this will bog down the execution speed without improving the accuracy.

There exists three types of POS taggers:

- **Base Taggers**: assign the most frequent tag to a word derived from a corpus.
- **Rule-based taggers**: uses linguistic rules to eliminate ambiguity or to change tags with respect to the context. Rules can both be handwritten or derived from a corpus.
- **Statistical taggers**: uses the likelihood of tagging sequences of n-grams, derived from a corpus, to decide which tags to distribute. These taggers usually use Hidden Markov model (HMM) or variation of the Viterbi algorithm.

2.1.3 Parsing

Parsing, or syntax analysis, is the task of analysing a sentence and determine the grammatical structure of it with respect to the grammar of the language. This is usually done with well known parsing methods such as top-down parsing or bottom-up parsing where the lexemes attribute comes from the result of POS tagging.

2.2 Icelandic Frequency dictionary

IFD (The Icelandic Frequency dictionary) was published by the Institute of Lexicography in 1991 and is currently the only existing POS tagged Icelandic corpus (Pind, Magnússon, & Briem, 1991). *IFD* is carefully balanced and includes just over half a million running words from several sources such as novels, science texts, news text and texts from children's books. All the text in *IFD* was first published in 1980 - 1989. *IFD* was originally tagged by a POS tagger written by Stefán Briem (Briem, 1989) and then hand-corrected afterwards. The tagset used in *IFD* is similar to traditional analysis of Icelandic grammar categories and word classes. The tag string consists of most six characters where each

character has a particular morpho-syntactic meaning. For illustration see table 1. The tag *fp1en* corresponds to the pronoun *Ég* and is decoded as: *f* = pronoun, *p*=personal pronoun, *1* = 1st person, *e*=singular number, *n* = nominative case.

The *IFD* tagset includes 700 tags. This high number of tags reflects the complexity of the language compared to a less complicated language such as Swedish which only has 156 tags in one of its tagset (Nivre & Grönqvist, 2001). Of the word forms in the *IFD*, 15.9% are ambiguous as to the tagset within the *IFD* (Helgadóttir, 2004).

2.3 IceNLP

In our *ICALL* applications we use the *IceNLP* toolkit which is an *NLP* toolkit for Icelandic in development by Hrafn Loftsson and Eiríkur Rögnvaldsson. Currently, *IceNLP* includes a tokenizer, a morphological analyser called *IceMorphy*, a POS tagger called *IceTagger* and a finite-state parser called *IceParser*. We will now describe the construction and functionality of these components and discuss the accuracy measurmets done by the authors.

2.3.1 IceTagger

IceTagger is a linguistic rule-based POS tagger that consists of a morphological analyser, *IceMorphy*, local rules for initial disambiguation and heuristics for further disambiguation (Loftsson, 2008). *IceTagger* works as follows: *IceMorphy* looks up each word in a lexicon (derived from *IFD*) and returnes a tag profile (all possible tags for that word) if the word is found in the lexicon. If the word is not found then *IceMorphy* tries to guess the correct tag profile for the word using morphological analysis and ending analysis. Then local rules are used to eliminate tags that are illegimate based on local context then heuristics are used to change tags that are not in nearest neighborhood. *IceTagger* uses the same tagset as in *IFD* and obtains 91.5% tagging accuracy, measuring the whole tag string, on IFD.

In a recently released paper, from the authors of *IceNLP*, they describe new variations of *IceTagger* (Loftsson, 2009). In these new variations they have integrated an HMM pos tagger into a linguistic rule-based POS tagger for wordclass disambiguation. They call this method *HMM+Ice* (HMM tagger runs before IceTagger). They also describe two other variations: *Ice+HMM* (*IceTagger* runs before HMM tagger) and *HMM+Ice+HMM* (HMM tagger runs first, then *IceTagger* and then HMM tagger again). They state that *HMM+Ice+HMM* obtains 92.5% tagging accuracy, measuring the whole tag string, using a corrected version of *IFD*.

This is the highest tagging accuracy stated for an Icelandic POS tagger but is still lower then the minimum acceptance accuracy that we discussed in the introduction.

We believe that the data used in these accuracy measurements are much more detailed then we would ever see from a normal user of *ICALL* application and that the desired accuracy of 95% could be achieved by remeasuring the accuracy on data from normal users.

Moreover, in the evaluation they are measuring the whole tag string. With an *ICALL* application that would only need a part of the tag string that included the desired data, then we can reduce the tagset to that subpart and only measure the tagging accuracy on that specific part. Why include data in our measurements that are not needed and will never affect the application nor the user of it?

word	POS tag
Ég	fp1en
var	sfg1eþ
með	ao
pening	nkeo
inni	aa
á	aþ
bók	nveþ

Table 1: Example from *IFD*.

2.3.2 IceParser

IceParser is an incremental finite-state parser that takes text tagged with the *IFD* tagset as an input and returns the text annotated with a shallow scheme, that was created by the authors of *IceNLP*. This scheme includes the constituents structure of the sentence and its syntactic functions.

IceParser consists of two modules: a phrase structure model that indicates constituents structures and a syntactic function model that that add tags for grammatical functions.

An evaluation of *IceParser* with F-measure using a golden standard of 509 randomly selected sentences and achieve 96.7% accuracy for constituent functions and 84.3% for syntactic structures.

When *IceTagger* was used to tag the sentence before parsing the F-measurements dropped down from 96.7% to 91.9% for constituents structures, and 84.3% to 75.3% for syntactic functions.

2.4 Overview of foreign ICALL applications

- **Banzai** is a Japanese web-based language software package which enables learners to freely produce Japanese sentences and to provide detailed feedback concerning the specific nature of the learner's errors (Nagata, 2002). It accepts inputs in kana and kanji, and presents relevant photographic and graphical images of Japan and of everyday situations.

- **WERTI** is an *ICALL* system that can be viewed as an intelligent automatic workbook where students can practice their reading, listening, and writing skills in english (Amaral, Metcalf, & Meurers, 2006). WERTI uses state-of-the-art *NLP* technology to analyze students input, and detect spelling, morphological, syntactic and semantic errors and contains student models to keep track of stundents progress. Its *NLP* components allow the system to handle activities that go beyond the usual multiple choice or fill-in-the-blanks used by regular *CALL* systems.

- **E-tutor** is a comprehensive language learning environment for German. Users can practice their language skills, e.g. vocabulary, listening, and reading, comprehension and grammar at all levels of German, from beginner to advanced (Heift & Nicholson, 2001).

- **REAP** is search engine that offers individualized practice to students by presenting authentic and appropriate reading materials selected automatically from the web. REAP was extended to French in 2008 by Juan Pino (Heilman, Collins-Thompson, & Eskenazi, 2006).

- **GRIM** is a Swedish language learning environment with several tools for language exploration such as grammar checking, surface syntactic analysis, word inflection and a dictionary (HWestlund, 2004).

- **OAHPA** is a collection of different programs for learning the north Sámi language, and contains programs which use *NLP* (Antonsen, 2009).

2.5 Icelandic CALL applications

- **Icelandic Online**: is a web based distance-learning system for foreign students where all communication between students and teacher takes place through the website. This application is currently being used in a distance learning course in Icelandic at the University of Iceland (UI). The website is divided into three progressive courses. Each course takes the user through a series of predefined exercises. Icelandic online was developed in three stages from 2004 to 2006 by students and teachers at the Centre for Research in the Humanities at UI and received a grant from The Icelandic Centre for Research (RANNÍS) in 2006. The project's website is http://icelandic.hi.is/

- **Námsgagnastofnun, The National Centre for Educational Materials (NCEM), Website**: NCEM is a state-run publishing house that provides the compulsory schools (students aged 6-16) in Iceland with all kinds of educational materials. On their website (www.namsgagnastofnun.is) they have a family of *CALL* applications aimed at many different age groups.

3 System

In this section we will discuss the design and implementation of our *ICALL* website, our *ICALL* multiplayer boardgame *Málflækjan* and an *NLP* framework that we created for our applications.

3.1 ISLinguistic

ISLinguistic is a configurable *NLP* framework, written in Java, that we created as the main building block for our *ICALL* application. The motivation for creating this framework was twofold. First to group together all code that dealt with *NLP* and create an easy way to plug-in and configure third-party *NLP* tools through a configuration file. Secondly to create framework that could easly be used in other *ICALL* applications. Figure 1 shows a overview of our framework.

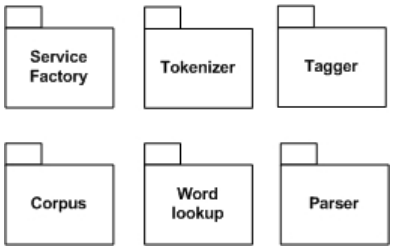

Figure 1: Package overview of ISLinguistic

We will now describe the inner structure of each of its parts in more depth to give a better understanding of their purpose and functionality before we discuss our website and our game.

3.1.1 Service Factory

The Service Factory is a class factory that instanciates other services in the framework. The service factory reads a configuration file to see what class to instanciate. If users want to implement their own services they can change the locations of that service to their implementations in the configuration file.

3.1.2 Tokenizer

Contains a tokenizer. The default implementation uses the tokenizer in *IceNLP*, but users can also implement their own tokenizer. That is done by implementing the *ITokenizerService* interface in the ISLinguistic.Tokenizer package and point to the new implementation in the configuration file. Figure 2 shows an overview of the Tokenizer package. Listing 2 shows how to request an instance of the Tokenizer

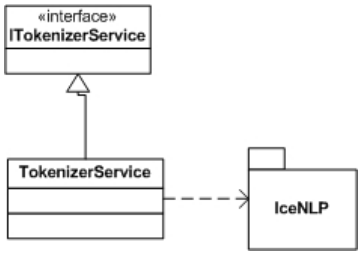

Figure 2: Overview of the Tokenizer package

service from the Service factory and how to tokenize a sentence.

```
ITokenizerService service =
ServiceFactory.getTokenizerService();
List<String> tokens =
     service.TokenizeSentence(text);
```

Listing 1: How to get Tokenizer-service from the Service-factory

3.1.3 Tagger

Contains a *POS* tagger. The default implementation uses *IceTagger*. Users can choose between different versions of *IceTagger* in the configuration file, or provide their own *POS* tagger by implementing *ITaggerService* interface in the ISLinguistic.Tagger.ITaggerService package and point to the new implementation in the configuration file. Figure 3 shows an overview of the Tagger package. It is also possible to add

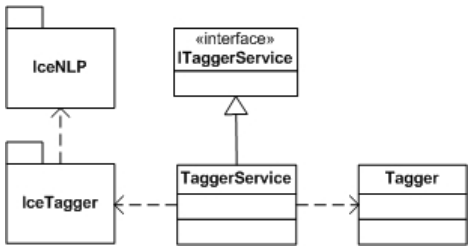

Figure 3: Overview of the Tagger package

more POS taggers inside the Taggers.xml file and enable a simple voting scheme in the configuration file. Listing 2 shows how to request an instance of the Tagger service from the service factory.

```
ITaggerService service =
    ServiceFactory.getTaggerService();
List<WordTag> result = service.tag(sentence);
```

Listing 2: How to get Tagger-service from the Service-factory

3.1.4 Parser

Contains a parser. The default implementation uses the *IceParser* but users can provide their own parser by implementing *IParserService* in the ISLinguistic.Parsing package and point to the new implementation in the configuration file. Figure 4 shows an overview of the Parser package and Listing 3 shows how to request an instance of the parsing service from the service factory.

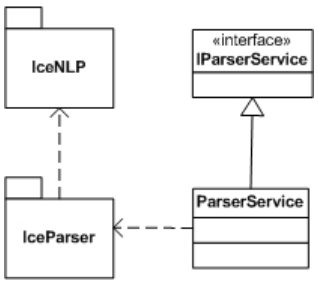

Figure 4: Overview of the Parser package

```
IParserService service =
    ServiceFactory.getParserService();
String result = service.parse(sentence);
```

Listing 3: How to get Parser-service from the Service-factory

3.1.5 Word lookup

Word lookup is a simple service that is used to check if a word is in a list of a collection of words. This service can be used as a simple speller. The default list that we provide is the Icelandic GNU Aspell dict list and is available at ftp.rhnet.is/pub/gnu/aspell but users can change the underlying list in the configuration file.

Listing 4 shows how to request an instance of the word lookup service from the service factory.

```
IWordLookupService service =
    ServiceFactory.getWordLookupService();
if(service.hasWord("punktur")){/* code section*/}
```

Listing 4: How to get Tokenizer-service from the Service-factory

3.1.6 Corpus

The Corpus package includes a single service for getting an instance of a corpus into memory. The underlying corpus can be changed in the configuration file.

```
IParserService service =
    ServiceFactory.getCorpusService();
List<String> corpus = service.getCorpus;
```

Listing 5: How to get Corpus-service from the Service-factory

3.2 ICALL web page

Our *ICALL* website is written in Java, using the MVC pattern from the Spring framework and *ISLingustic*. Figure 5 is a screen capture of the website.

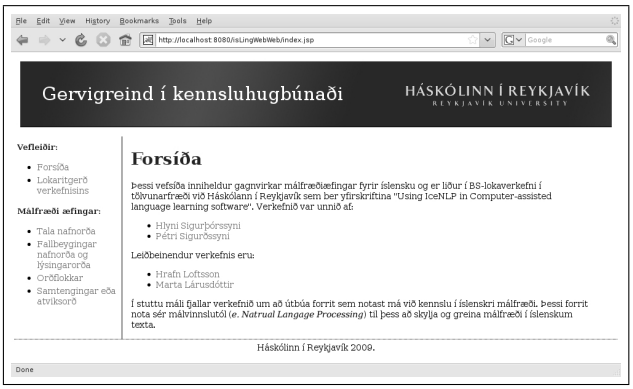

Figure 5: Screen capture of the ICALL website.

The website contains a series of grammar exercises that are generated from users input. These exercises are:

- **Noun number**: Users must determine the number of each noun that appear in their sentences. A drop down menu will appear beside each noun where they can select between singular or plural.
- **Noun case**: User must determine the cases of nouns that appear in their sentences. A drop down menu will appear beside each noun where users can select the cases.
- **Simple word classes**: Users must determine word classes for nouns, adjectives and verbs that appear in their sentences. A drop down menu will appear beside each noun, adjective and verb where users can choose between classes.
- **Conjunction or adverb**: Users determine whether a word is a conjunction or an adverb. A drop down will appear beside every conjunction and adverb that appears in their sentences that users can use to select between conjunction and adverb.

Figure 6 and figure 7 are screen captures of the word classes exercise on the website.

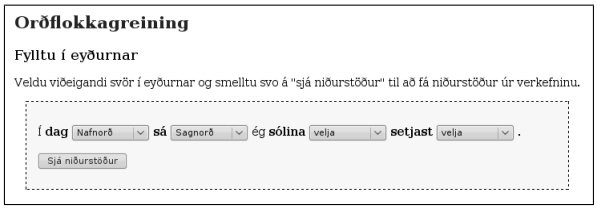

Figure 6: Screen capture of the word classes exercise where a user has created an exercise with the sentence *Í dag sá ég sólina setjast*

We created two versions of the website. One that is a standalone version where users can select exercises from a navigation list and another that was in a form of a survey. The survey version takes the user through all the exercises.

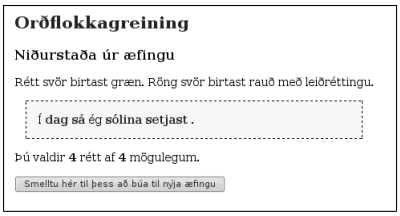

Figure 7: Screen capture of the results from the word classes exercise on the webpage

After each exercise the user is asked to fill out a survey that measures the trust from the users. Each sentence that the user typed in is then stored to a database.

3.3 Málflækjan

Málflækjan is a board game that puts the linguistic skills of a player to the test in a social enviroment. The game has four players and a deck of 73 cards. Each card has an attribute, a wordclass or a morphological feature, written on it and a number indicating its value. Table 2 contains a list of all the cards in the game. The deck is shuffled and four cards are distributed to each player. A player is then chosen randomly to be the *tangler*, this player gets to pick a sentence of three to nine words and the player on his/her right hand side then gets to play.

Each time a player has a turn, (s)he can choose to apply one or more of his/her cards to a single word on the table or draw two more cards. If the player chooses to apply cards to a word, (s)he puts them beside the appropriate word and announces his action. Then the tag of the word is checked against the attribute on the cards that the player applied. For every card that matched, the player gets points equal to the points on the card, but for every card that did not match the player gets a single penalty point regardless of the point value of the card. The cards that the player used are discarded and if at least one of the cards matched the tag for the word, the tag of the word is shown to all of the players and the word is discarded, or "grayed out". If there is only one word left on the table, the player who just played gets to be the *tangler*, but this time (s)he has to use the word that was left over when (s)he creates a new sentence. The player on his/her right hand side then gets a turn. If a player

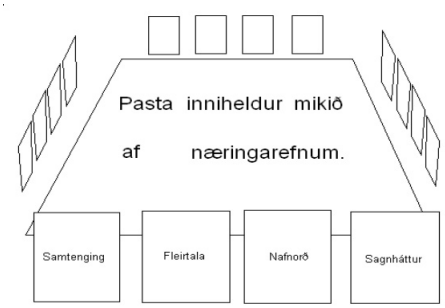

Figure 8: A depiction of a WordTangle game being played.

cannot use any of his/her cards and the deck is empty, (s)he has to forfeit his/her turn and the player on his/her right hand side gets to play. The game is over when the deck is empty and no one can apply more cards. Some of the cards are blank, these cards can be used as any tag that the player decides and has the point value of one. The player needs to announce what attribute (s)he wants that card to represent when (s)he applies it to a word. When a player gets a card that has more than one attribute on it, (s)he can decide which attribute (s)he wants to use when (s)he applies it to a word. When a card has more than one attribute, the attributes are put together in such a way that they could never both apply to the same word-class and thus never to the same word. The player should not have to decide which attribute (s)he uses since it should be obvious from the word-class. Figure 8 depicts the setup of the game.

We implemented *Málflækjan* as a multiplayer (client/server architecture) *ICALL* game where users can meet eachother on a server and play the game toghether over a network. As far as we know such application has never been created before in the field of *CALL*, namely the use of NLP technology in a competitive multiplayer board game. The emphasis of the game is to teach and train the linguistic skills of the user in an enjoyable but challenging environment. The user can choose who (s)he plays, play against random opponents or just stay in the chat room and mingle. This social aspect could increase the enjoyment of the game while the player is still constantly solving different grammar problems and thus training his/her grammar skills. Every user will have an entry in a scoreboard so that they can monitor their improvement over time, and compare their progress to others. This can be a great way of encouraging players to progress their learning and even look for alternative ways of improving their skills.

We will now describe the implementation of the client and the server.

3.3.1 Málflækja Client

The *Málflækja* client was written in Python. It uses the Panda3D graphics library for rendering. It does not contain any game logic, since that is implemented on the server side. When the user logs on, (s)he enters the lobby where (s)he can browse through existing games and chat with other players. From there the player can either create its own game or join another game. When all of the players in the same game are ready, the game begins according to the rules described earlier.

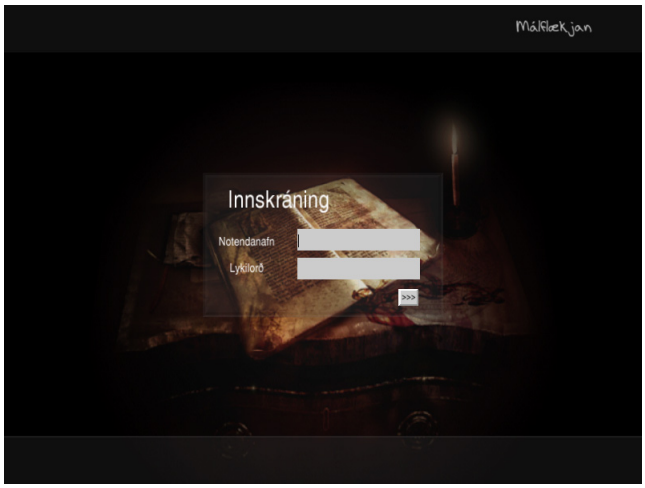

Figure 9: Málflækjan login screen.

Card title	amount	Points
Nafnorð	3	1
Lýsingarorð	3	1
Fornafn	3	2
Sögn	3	1
Greinir	3	1
Atviksorð/Forsetning	3	2
Samtenging	3	2
Nafnháttarmerki	3	1
Töluorð	2	1
Karlkyn	3	1
Kvenkyn	3	1
Hvorugkyn	3	1
Eintala	3	1
Fleirtala	3	1
Nefnifall	3	2
Þolfall	3	2
Þágufall	3	2
Eignarfall	3	2
Fyrsta persóna	1	2
Önnur persóna	1	2
Þriðja persóna	1	2
Nútíð	1	1
Þátíð	1	1
Nafnháttur	2	1
Lýsingarháttur þátíðar / Frumstig	1	2
Boðháttur / Frumstig	1	2
Framsöguháttur / Miðstig	1	2
Viðtengingarháttur / Miðstig	1	2
Sagnbót / Efstastig	1	2
Lýsingarháttur nútíðar / Efstastig	1	2
Germynd / Atviksorð / Forsetning	1	2
Germynd / Frumstig	1	2
Miðmynd / Miðstig	1	2
Miðmynd / Efstastig	1	2
Blank	3	1

Table 2: Málflækja cards

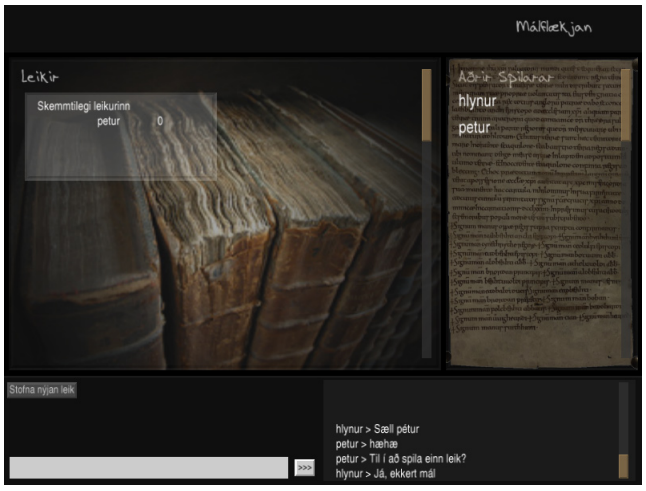

Figure 10: Málflækjan lobby, here players meet to coordinate games.

Figure 11: Málflækjan, midgame screenshot, here points are being awarded for the correct choice.

3.3.2 Málflækja Server

The *Málflækja* server was written in Java and uses our framework that was described in this section.

The server consists of

- *Socket handler* that handles incoming connections. This sockets spawns a new client thread that handles all socket communications between the client and the server.
- *Message handler* that handles all or distributes incoming messages from clients.
- *Game handler* that handles all games that are being played on the server. Each game has its own thread. Game specific messages get delivered from message handler to this handler.

The network protocol can be viewed in the final report for this project.

4 Evaluation

We sent out an e-mail request that included the survey, that we described in 3.2, to every student in the Computer Science department at Reykjavík University. 426 students received the e-mail, 75 people visited the website, and 33 people finished the survey. The average age of the participants was 27 years and the average study age was 2.4 years. For all of the students that finished the survey, we collected the sentences that they used in the exercises, total of 283 sentences or 3034 words, including punctuations. 24 words were misspelled and were removed.

We will now discuss the data analysis that we did on the data that we got from the survey. We will begin by discussing the accuracy measurements on the *IceNLP* toolkit with the sentences that we collected and then the analysis on the data that we got from the questionnaires.

4.1 IceNLP accuracy evaluation

For the accuracy evaluation we created two golden standards: one for measuring the tagging accuracy of the four

different variations of *IceTagger* (see section 2.3.1), and another for doing a f-measure on IceParser.

4.1.1 IceTagger tagging accuracy evaluation

Table 3 contains the results from tagging accuracy measurements for the four different types of IceTagger using our golden standard. In these measurements we were measuring the whole tag string.

When we were hand correcting our golden standard, we saw that many of the errors that was made by IceTagger were incorrect analysis of proper nouns and local name. So we measured the tagging accuracy while ignoring proper nouns and local names for nouns. Table 4 contains the results from these measurements.

IceTagger	95.0%
Ice+HMM	94.7%
HMM+Ice	94.3%
HMM+Ice+HMM	94.2%

Table 4: Tagging accuracy, ignoring proper nouns and local names, for the four different variations of IceTagger.

We also measured the tagging accuracy of the four variation if IceTagger, on tag strings that included only information needed in Málflækjan. These tag reductions are

- Strong or weak analysis for adjectives are ignored
- Analysis for pronoun and local names are ignored.
- Only the word-class analysis numerals are analysed.
- Adverbs and preposition are treaded as the same word class.

Table 5 includes the results from these measurements.

IceTagger	95.3%
Ice+HMM	95.1%
HMM+Ice	94.8%
HMM+Ice+HMM	94.7%

Table 5: Tagging accuracy for tag strings relevant to our ICALL game.

4.1.2 IceParser accuracy evaluation

We only measured noun phrase annotation in *IceParser* output. The reason for that is that noun phrase accuracy gives a good indication for the other phrase types since it appears most frequently in sentences (Loftsson & Rögnvaldsson, 2007) and doing a measure on each phrase type is a very time consuming task.

We randomly selected 50 sentences from the golden standard that we created for the accuracy measurements and used them to create a golden standard for noun phrase annotations. doing f-measure for each phrase type is a time consuming task.

Table 6 contains numbers the result from the comparison of the noun phrase output from *Iceparser* with our golden standard.

We used (1) to calculate the precision of *IceParser*, which was 95.48%.

Number of noun phrases by IceParser	155
Number of corrected noun phrases by IceParser	148
Number of noun phrases in golden standard	158

Table 6: Results from noun phrase parsing of *IceParser* on text that was tagged by *IceTagger*

$$\text{Precision} = \frac{\text{Correct constituents in parsers output}}{\text{Number of constituents in parsers output}} \quad (1)$$

We used (2) to calculate the recall of *IceParser*, which was 93.67%.

$$\text{Recall} = \frac{\text{Correct constituents in parsers output}}{\text{Number of constituents in golden standard}} \quad (2)$$

We used (3) to calculate the harmonic value (f-measure), which was 94.54%.

$$\text{F-measure} = \frac{2 * \text{Precision} * \text{Reacll}}{\text{Precision} + \text{Recall}} \quad (3)$$

4.2 Survey data analysis

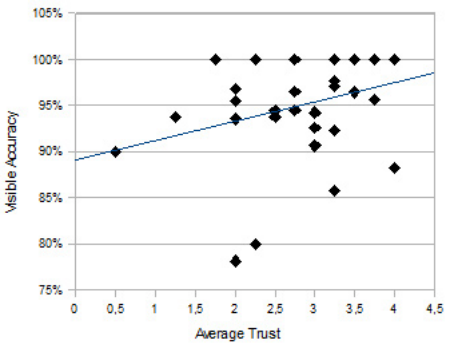

Figure 12: Correlation between visible accuracy and trust.

For our trust measurement, we asked the user, after every exercise, how confident (s)he was that the exercise that (s)he took was correct. The user was given 5 choices to choose between and we associated a number to every option. For every exercise, we recorded how many mistakes IceNLP made that would affect that particular exercise. We did that by reducing the tagset to only the things that was being tested in the exercise. For example, in the noun tense exercise, we only looked at the accuracy of IceNLP when determining nouns and the tense of nouns. We call these measurements visual accuracy. When we compare the trust value given by the user after the exercise and the visual accuracy, we see that with increasing accuracy the trust increses as well (see fig. 12). When we only look at the total visible accuracy of all of the exercises, we get the average of 95.08%.

After every exercise the user was asked if and how many errors (s)he noticed in the exercise. We compared this result to the given trust and found that the more errors the user noticed the less trust (s)he had toward the exercise (see fig. 13).

At the end of every exercise we presented an AttrakDiff 2 questionaire that has been translated in Icelandic by

	IceTagger	Ice+HMM	HMM+Ice	HMM+Ice+HMM
Number of known words:	2828	2828	2828	2828
Number of correctly tagged known words:	2698	2692	2681	2678
Known word tagging accuracy:	95.4%	95.2%	94.8%	94.7%
Number of unknown words:	206	206	206	206
Number of correctly tagged unknown words:	164	164	162	162
Unknown tagging accuracy:	79.6	79.6	78.6	78.6
Number of overall tagging errors:	172	178	191	194
Overall tagging accuracy:	94.3%	94.1%	93.7%	93.6%

Table 3: Tagging accuracy for the whole tag string for the four different variations of IceTagger.

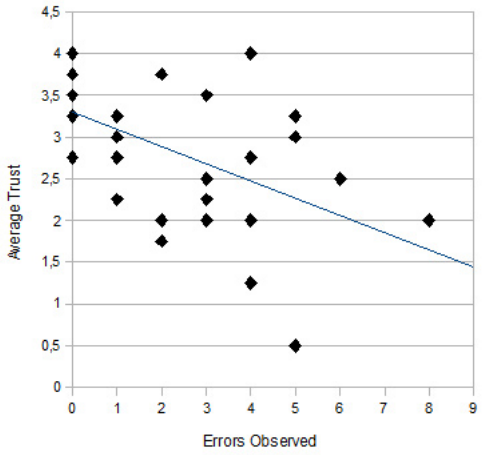

Figure 13: Correlation between trust and number of errors observed by the user.

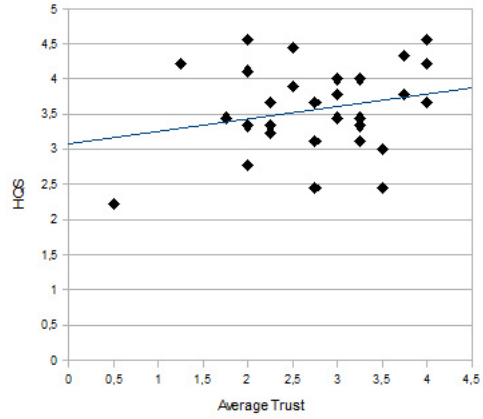

Figure 14: Trust compared to the HQS value.

Marta Lárusdóttir (Lárusdóttir & Isleifsdottir, 2008). This questionaire should be a good measurement of the user experience. We decided to put emphasis on the hedonic stimulation attribute of the questionaire, so we deducted many of the questions from the questionaire that related more to the user interface than the overall experience. The hedonic quality stimulus represents how much the system stimulates personal growth. When we compare this value to our trust values, we see that higher trust tends to go with a higher HQS value(see fig. 14).

5 Discussion

By remeasuring the accuracy of *IceTagger* on the sentences that we got from the survey we are seeing accuracy around 93% - 94% *(see table accuracyMeasurement)*. By excluding proper nouns and local names we are seeing tagging accuracy around 94% - 95%. Incorrectly analysed proper nouns or local names would not affect the applications that we created in this project, since we are never using them. But creating a better proper noun/local name analyser into *IceTagger* would most likely improve its accuracy.

The results from the tagging accuracy also show that the results that we got were the exact opposite to what we expected. While *HMM+Ice+HMM* returned the least accuracy, the unmodified *IceTagger* returned the highest. It seems that when dealing with relatively simple sentences the basic implementation of *IceTagger* works best. This contradicts with the findings in the paper (Loftsson, 2009), but since the dataset that we were working on is much smaller than *IFD* it might not be statistically reliable. These results also show that the default version of *IceTagger* is a viable solution since the HMM integration is not affecting the accuracy much for such simple sentences, and it's more efficient to run only one tagger instead of two or three.

The results from our f-measurements on *IceParser* were slightly higher then the evaluation in (Loftsson & Rögnvaldsson, 2007). This accuracy increase is parallel to the increase in the tagging accuracy. We believe that this increase is due to the fact that the sentences are simpler then many of the sentences in *IFD*.

When we excluded everything from the tag string that was not used in Málflækjan we got 95.3% accuracy with the default version of *IceTagger*.

As we stated in the introduction section of this paper, we were aiming for 95% accuracy for our *ICALL* applications to be acceptable.

These results indicates that using *IceNLP* in *ICALL* applications is a viable solution.

The results from our survey and the accuracy of the NLP engine shows that there is a clear correlation between how well the NLP engine tags and how much the users trust

the system. Given that the trust is one of the main factors in creating successfull teaching material, we find that the accuracy of our NLP engine is of great importance.

When we look at the results from the AttrakDiff 2 questionaire, we see that the hedonic stimulation quality is affected by the trust of the user. The hedonic stimulation quality is a measurement of enjoyability when using the system, and how much the application encourages personal growth. People want to develop their skills and knowledge further and these are the attributes of the product that allow for that to happen. This suggests that when a user has more trust in the material presented to him, he is more encouraged to develop his skills further.

6 Conclusion

In this report we have described an evaluation of the accuracy of *IceNLP* that indicated that the toolkit is accurate enough to be used in similar applications to the ICALL applications that we have described in this report.

References

Amaral, L., Metcalf, V., & Meurers, D. (Eds.). (2006). *Pre-conference workshop on nlp in call – computational and linguistic challenges. calico 2006. may 17, 2006. university of hawaii.*

Antonsen, L. (2009). *Oahpa pedagogical programs based on linguistic resources.* (Is available at http://spraakbanken.gu.se/)

Beatty, K. (2003). *Teaching and researching computer-assisted language learning: computer-assisted language learning* (illustrated ed.). Pearson Education.

Briem, S. (1989). Automatisk morfologisk analyse af islandsk tekst. In *Papers from the seventh scandinavian conference of computational linguistics.* Reykjavik, Iceland.

Delcloque, P. E. J.-P. (2000). *History of call.* Snapshot of Philippe Delcloque's History of CALL website that has been taken offline. (Under revision Ana Gimeno, President of EUROCALL and is available on their website http://www.eurocall-languages.org/)

Heift, T., & Nicholson, D. (2001). Web delivery of adaptive and interactive language tutoring. *International Journal of artificial Intelligence in Education*, 12.

Heilman, M., Collins-Thompson, K., & Eskenazi, J. C. andMaxine (Eds.). (2006). *Classroom success of an intelligent tutoring system for lexical practice and reading comprehension.* (In proceedings of the Ninth International Conference on Spoken Language Processing.)

Helgadóttir, S. (2004). Testing data-driven learning algorihms for pos tagging of icelandic. *Nordisk Sprogteknologi. Årbog*, 257-265.

HWestlund, S. (2004). *Utformning och implementation av en interaktiv miljö för andraspråksinlärning.* Unpublished master's thesis, Department of Numerical Analysis and Computer Science, Royal Institute of Technology.

Loftsson, H. (2006). Tagging Icelandic text: an experiment with integrations and combinations of taggers. *Language Resources and Evaluation*, *40*(2), 175–181.

Loftsson, H. (2008). Tagging Icelandic text: A linguistic rule-based approach. *Nordic Journal of Linguistics*, *31*(1), 47–72.

Loftsson, H. (2009). Correcting a PoS-tagged corpus using three complementary methods. In *Proceedings of the 12^{th} conference of the european chapter of the acl (eacl 2009).* Athens, Greece.

Loftsson, H., & Rögnvaldsson, E. (2007b). IceNLP: A Natural Language Processing Toolkit for Icelandic. In *Proceedings of interspeech 2007, special session: "speech and language technology for less-resourced languages".* Antwerp, Belgium.

Loftsson, H., & Rögnvaldsson, E. (2007). Iceparser: An Incremental Finite-State Parser for Icelandic. In J. Nivre, K. M. Heiki-Jaan Kaalep, & M. Koit (Eds.), *Proceedings of the 16th nordic conference of computational linguistics (nodalida-2007).* Tartu, Estonia.

Lárusdóttir, M. K., & Isleifsdottir, J. (2008). *Measuring the user experience of a task oriented software.* (Where was this pulished?)

Nagata, N. (Ed.). (2002). *Banzai: Computer assisted sentence production practice with intelligent feedback.* (In proceedings of the Third International Conference On Computer Assisted Systems for Teaching and Learning/Japanese (CASTEL/J), 2002.)

Nivre, J., & Grönqvist, L. (2001). Tagging a corpus of Spoken Swedish. *International Journal of Corpus Linguistics*, *6*, 47–78.

Pind, J., Magnússon, F., & Briem, S. (1991). *The icelandic frequency dictionary.* The Institute of Lexicography, University of Iceland, Reykjavik, Iceland.

Rögnvaldsson, E. (2003). Tungumál, tölvur og tungutækni. *Íslenskt mál*, *23*, 71-93.

Stifle, J. (1971). *The plato iv architecture* (Tech. Rep.). Illinois Univ., Urbana. Computer-Based Education Research Lab.

Contents