Keynounce
—
A Game for Pronunciation Generation through Crowdsourcing

Student Research Project at the Cognitive Systems Lab (CSL)
Prof. Dr.-Ing. Tanja Schultz
Department of Computer Science
Karlsruhe Institute of Technology (KIT)

from

Daniel Lemcke

Supervisors:

Prof. Dr.-Ing. Tanja Schultz
Dipl.-Inform. Tim Schlippe

Begin:	02. January 2013
End:	31. March 2013

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, den 8. February 2013

Abstract

With the online tool Keynounce we prove, that it is possible to generate pronunciation dictionaries with the help of an unknown and potentially unskilled crowd on the Internet. By providing a keyboard of sounds, as well as a synthesizer, the users actually have fun transcribing words into phoneme strings. This significantly reduces the cost of creating dictionaries compared to paying skilled linguists.

In addition to reducing costs, creating dictionaries for languages for which linguists are rare or not available, will be possible. Dictionary creation then simply depends on finding enough willing users with a sufficient knowledge (fluent speakers) of the targeted language. The languages in our testing scenario where English and German.

By providing a gaming experience, users voluntarily create pronunciation dictionaries. Dictionary creation is thus reduced to automatically filtering the provided content to get the best results.

For German pronunciations, all the collected data results in a phoneme error rate of 34.7% compared to GlobalPhone, which is based on validated pronunciations. After applying one simple filtering technique this number drops to 22.7%. When we discard the results from users who did not participate long enough, the phoneme error rate drops to 31% and 20.8% after post-processing. These results are based on a rather small crowd an with a bare minimum of post processing.

Zusammenfassung

Mit dem Online Tool Keynounce beweisen wir, dass es möglich ist, Aussprachewörterbücher mit Hilfe von unbekannten Internetnutzern zu erstellen. Keynounce stellt eine Tastatur mit verschiedenen Phonemen und eine synthetische Sprachausgabe zur Verfügung. Durch diese Kombination haben die Nutzer Spaß daran, Wörter in Aussprachen zu transkribieren. So erreichen wir eine wesentliche Reduzierung der Kosten zur Erstellung eines Aussprachewörterbuchs, vor allem im Vergleich mit professioneller Erstellung.

Zusätzlich zur Kostenreduzierung wird so auch die Erstellung von Aussprachewörterbüchern von Sprachen möglich, für die es wenige oder sogar gar keine Experten gibt. Es müssen nur genug Menschen gefunden werden, die diese Sprache fließend sprechen. Die von uns getesteten Sprachen sind Englisch und Deutsch.

Durch die spielerische Gestaltung von Keynounce sind unsere Nutzer mit Spaß an den Aufgaben und motivieren eventuell neue freiwillige Nutzer. Insgesamt reduzieren wir die Erstellung eines Aussprachewörterbuchs damit auf das Filtern von Ergebnissen.

Nimmt man alle deutschen Aussprachen zusammen, kommen wir auf eine Phonemfehlerrate von 34,7% gegenüber GlobalPhone, welches auf vorhandenen Aussprachen basiert. Nach der Anwendung einer einfachen automatischen Filterungstechnik erreichen wir 22,7% Phonemfehlerrate. Indem wir die Ergebnisse von Nutzern ignorieren, die nur wenige Wörter bearbeitet haben, können wir die Phonemfehlerrate auf 31% oder nach der Filterung auf 20,8% reduzieren.

Contents

1. Introduction

This work describes the complete process of building our web-based system Keynounce to gather phonetic notations. These notations can be used to create pronunciation dictionaries for Automatic Speech Recognition (ASR) systems at Cognitive Systems Lab (CSL) [1]. In the future the system can be an extension of the Rapid Language Adaptation Toolkit (RLAT) [2] to generate or enhance pronunciation dictionaries or let them be built by the Internet community.

With Keynounce we want to prove, that it is possible to generate high-quality pronunciation dictionaries with the help of an unknown and possibly unskilled crowd on the Internet. On the following pages we describe the complete process of creating the system, starting with the idea and the necessary terms and concepts, followed by the design and implementation choices that we have made. We explain the choices we have made and the solutions we have found, as well as the alternatives and why we have not chosen them.

Before designing a web-based system which encourages people to produce accurate pronunciations, a lot of decisions have to be made to make the system intuitive, robust and effective. The system should also engage the users' interest to prevent it from being a chore. It is our goal to discuss these decisions and review their impact on the system. Based on them, we used existing tools or built custom software to realize our system.

1.1 Goals of this Study

The main goal of this study is to prove, that anonymous Internet users can create or enlarge pronunciation dictionaries for ASR systems. The resulting dictionaries should have these qualities:

- Good quality compared to dictionaries created by linguists
- Low cost in creation
- Little to no postprocessing

To successfully accomplish this task, the main requirements are a stable Internet connection and an application which prompts the user and records his or her decisions.[1] There are other requirements for the system itself concerning usability, stability and performance. These constraints are secondary to the main requirements as the system could function without them. All requirements are discussed in detail in Section 4.1.

With this tool it will then be possible to harness the potential knowledge of Internet users all around the World Wide Web to create pronunciation dictionaries. This would reduce the costs of creating these dictionaries compared to paying skilled linguists. In addition to reducing costs, dictionaries for languages for which linguists are rare or not available at all will be possible. Dictionary creation will depend on finding enough willing users with a sufficient knowledge (fluent speakers) of the targeted language.

1.2 GlobalPhone and CMU Dict

GlobalPhone is an ongoing database collection that provides transcribed speech data for the development and evaluation of large speech processing systems in the most widespread languages of the world [3][4]. Pronunciation dictionaries in 20 languages have been established. As resources widely read national newspapers available on the World Wide Web were selected. Due to that, texts from national and international political and economic topics restrict the vocabulary.

As referencing material to compare our results with, we used English and German dictionaries in GlobalPhone style. The dictionary for the English language is the original GlobalPhone dictionary and contains 84,704 words or variations of a word. The German dictionary is based on the CMU dict[2] and contains 46,942 words or their variations. Both dictionaries feature one or more pronunciations for each word. We selected 100 random words from each dictionary. Pronunciation variations where not separately chosen. As we discuss in Section 5.3, we use these pronunciations as references for our results to compare with.

1.3 Crowdsourcing

The term crowdsourcing was coined by Jeffrey Howe[5] in 2006. Instead of the traditional outsourcing, where a company moves parts of its own structure or part of the workflow to a separate company, crowdsourcing gives a problem to an anonymous crowd. The incentive for the workforce is not simply money, but acknowledgment, fame, appreciation and joy are major factors to produce quality work.

After studying more than 40 definitions of crowdsourcing, Estellés and González (2012) propose a new integrating definition:

> Crowdsourcing is a type of participative online activity in which an individual, an institution, a non-profit organization, or company proposes to a group of individuals of varying knowledge, heterogeneity, and number, via a flexible open call, the voluntary undertaking of a task. The

[1]The willingness of the users is a prerequisite.

[2]https://cmusphinx.svn.sourceforge.net/svnroot/cmusphinx/trunk/cmudict/

undertaking of the task, of variable complexity and modularity, and in which the crowd should participate bringing their work, money, knowledge and/or experience, always entails mutual benefit. The user will receive the satisfaction of a given type of need, be it economic, social recognition, self-esteem, or the development of individual skills, while the crowdsourcer will obtain and utilize to their advantage that what the user has brought to the venture, whose form will depend on the type of activity undertaken [6].

1.4 Structure

In this chapter we have given a short introduction how we want to show, that pronunciation generation through Internet users is possible, as well as the reasons that motivated this study. We have also given a short overview about the Rapid Language Adaptation Toolkit and the GlobalPhone database and Crowdsourcing, which all feature in our study.

In chapter 2 we explain the basics, which are needed for this study. This includes the concept of Automatic Speech Recognition and more specifically the Pronunciation Dictionary, as well as Amazon's Mechanical Turk and the International Phonetic Alphabet. Chapter 3 gives an overview of related work. These works are Peekaboom, a game to have Internet users add meta information to pictures, a paper about the human factors when bootstrapping a pronunciation dictionary as well as the LexLearner which is also part of the Rapid Language Adaptation Toolkit.

In Chapter 4 we analyze the requirements for the system followed by establishing the targeted group of users, together with a consideration about motivation, usage, and availability. We also analyze the actual task that we could give the users in respect to difficulty, workability and incentive.

Chapter 5 shows our progress from the general interface design ideas to the finished interface. We also explain, why certain changes or decisions were necessary or useful. Which specific implementation is used in the backend is also discussed.

The results of three different sets of experiments are provided and explained in Chapter 7. We furthermore analyze the results as far as possible.

In Chapter 9 we conclude our work and point out, what changes or additions could be made. In addition we formulate some use-cases and possible extensions of this work and the impact this could have on ASR.

2. Basics

2.1 Automatic Speech Recognition

The fundamental problem of speech recognition is to find the most likely word sequence given audio file with speech. Factors like speaker variability, noisy environment and different properties of the recording equipment have negative influence on the recognition performance. The following formula (2.2), based on the Bayes' rule (2.1) summarizes the computational model used for large vocabulary continuous speech recognition (LVCSR):

$$\Pr(\mathbf{W}|\mathbf{X}) = \frac{\Pr(\mathbf{W})\Pr(\mathbf{X}|\mathbf{W})}{\Pr(\mathbf{X})} \tag{2.1}$$

$$\hat{\mathbf{W}} = \arg\max_{W} \Pr(\mathbf{W}|\mathbf{X}) = \arg\max_{W} \Pr(\mathbf{W})\Pr(\mathbf{X}|\mathbf{W}) \tag{2.2}$$

As a result of the digital signal processing the acoustic signal is represented as a sequence of acoustic vectors $\mathbf{X} = X_1X_2...X_n$ that captures the short time spectrum of the speech. The goal is to estimate the most probable word sequence $\hat{\mathbf{W}} = \hat{W}_1\hat{W}_2..\hat{W}_m$ depending on the linguistic knowledge we have for the language $\Pr(\mathbf{W})$ and the extracted acoustic rules. The probability of observing signal $\mathbf{X}$ given the fact, that word sequence $\mathbf{W}$ is spoken, forms the acoustic model $\Pr(\mathbf{X}|\mathbf{W})$. When computing the most probable word sequence, the denominator from the Bayes' rule $\Pr(\mathbf{X})$ is not considered, since it does not play a role in the maximization of the function. Finally, to find the word sequence with the highest probability, a search strategy must be applied. The most used search algorithms in speech recognition are A* and Viterbi search.

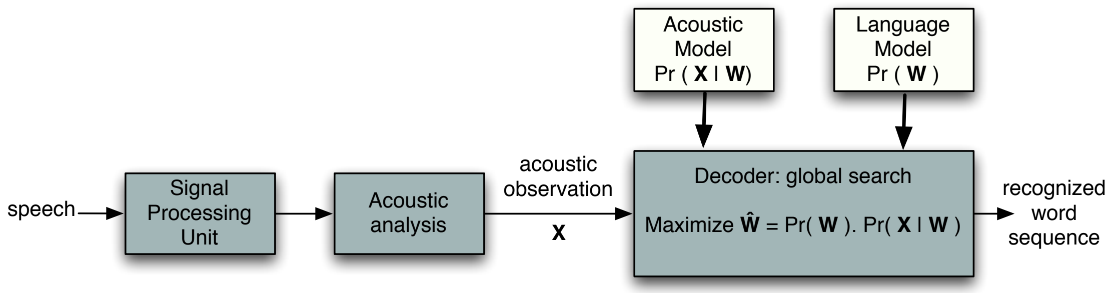


Figure 2.1: ASR

2.2 Pronunciation Dictionary

The acoustic model of the LVCSR is a mapping between transcriptions and pronunciations. The term "dictionary" is sometimes substituted with the term "lexicon". There is no standard for how to format a dictionary. Depending on the system's structure, the format can vary. Usually it is a map-like structure with words as keys and pronunciation strings as values.

Since a word can have multiple pronunciations, the dictionary structure must allow adding pronunciation variants and a way of marking them as pronunciation variants of the same word.

The keys (words) of the dictionary are language dependent and usually use the alphabet of the language to be recognized. The pronunciations of the words are given as phonetic transcription. There are standard phonetic transcriptions systems like the International Phonetic Alphabet (IPA), which is discussed in Section 2.4. There are many others developed for general or special purposes.

Example dictionary entries of English words and transcription in GlobalPhone are:

```
abbreviate  {{AX WB} B 9r IY V IY EY {T WB}}
abbreviated  {{AX WB} B 9r IY V IY EY DX AX {DH WB}}
abbreviated(2)  {{AX WB} B 9r IY V IY EY DX IX {DH WB}}
```

The pronunciation dictionary is the connection between words and phones. Depending on the application, the size of the dictionary can be from few words to millions of words. Speech recognizers for LVCSR use normally thousand words and above.

2.3 Amazon Mechanical Turk

The Amazon Mechanical Turk (MTurk) is a crowd sourcing Internet platform that establishes a marketplace for micropayment jobs [1]. MTurk is part of the Amazon Web Services. At this platform *Requesters* can offer tasks known as HITs (Human Intelligence Tasks) to *Turkers* and grant a monetary reward (plus 10% Amazon

[1]https://www.mturk.com/

fee) for its completion. Mostly these tasks consist of problems which computers are unable to do or are faster and more accurately done by humans. Widespread tasks are choosing the best photo that represents a given word or location, writing product descriptions, tagging content as safe for an immature audience, etc. Turkers can browse all tasks and can choose to complete them for the set payment.

The reward is payed after the Requester accepts the work, or automatically after a period of time, set by the Requester beforehand. In addition to the reward, the Turker also gets an update to his or her Approval Ratio (percentage of approved assignments), which effectively measures the quality of the work done by any Turker. Rewards are normally at around a couple of cents per assignment. So the Approval Ratio is a very important asset for the Turkers, since it may qualify them for higher paid tasks. It is also possible to assign a bonus to a Turker.

To assure a higher quality of results on his or her assignments, a Requester has some options of limiting access to certain groups of Turkers. Upon creating the HIT a number of predefined or self-created qualification can be chosen (Approval Rating, Number of HITs finished, Location, Adult Content). Qualifications created by the Requester have to be awarded to Turkers manually (e.g. Turkers who often worked for the same Requester) or in some cases automatically after completing a test. The HITs can also have a general entry test.

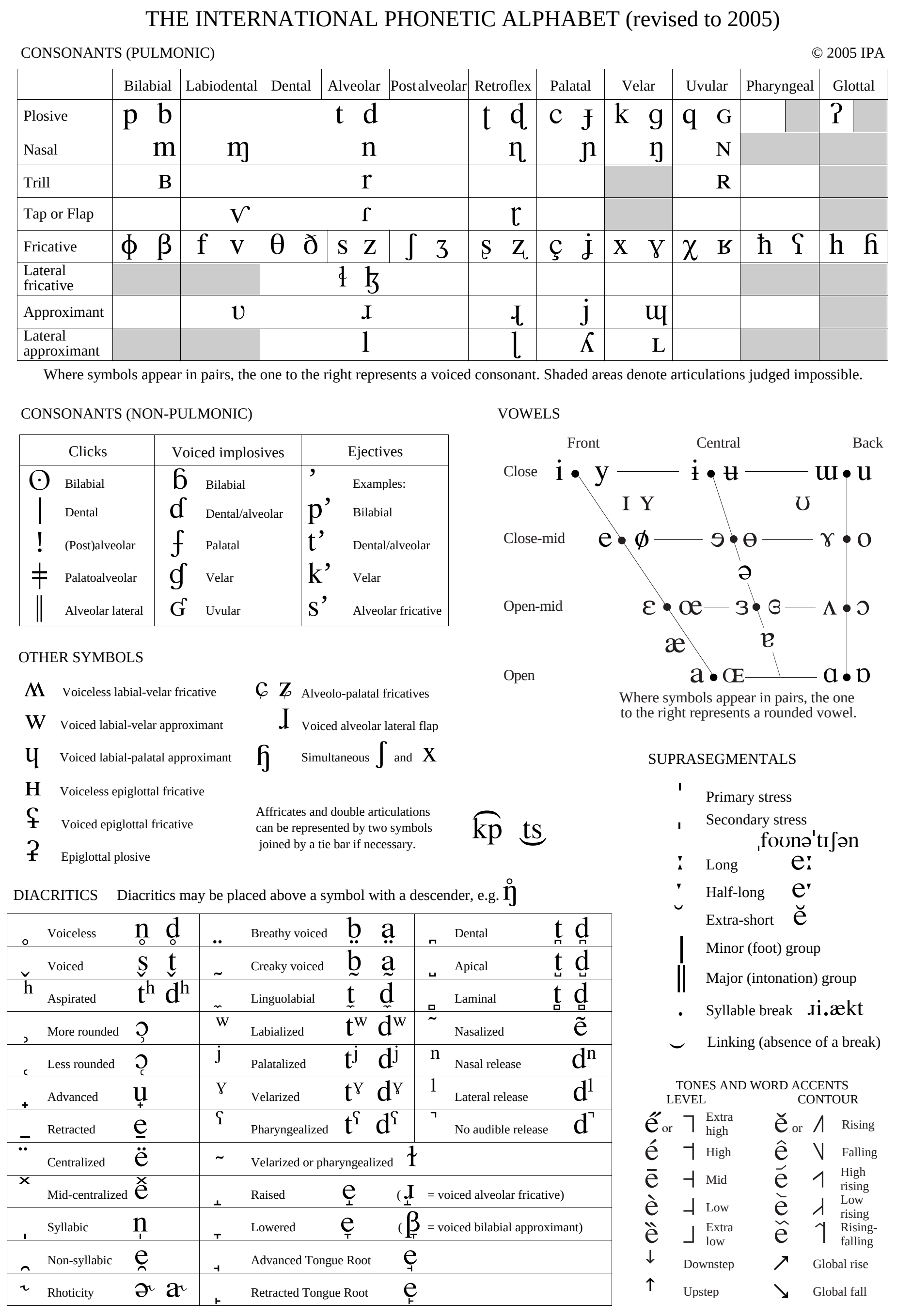
To place the HITs, the Requester can use an API or a more limited but better documented Requester site. The Requester can build single HITs. But most of the HITs consist of a batch of several assignments where one type of problem is given in variations. The Requester can also define, that a single assignment be completed by several unique Turkers. Assignments which were rejected by the Requester can automatically be replaced in the marketplace for other Turkers to work on.

At Cognitive Systems Lab Amazon's Mechanical Turk was used to perform text normalization tasks on large amounts of sentences in a tight time frame. This is possible, since a HIT with lots of assignments can be done by several Turkers simultaneously.

2.4 International Phonetic Alphabet

We have chosen the International Phonetic Alphabet (IPA) in order to get pronunciation information from the World Wide Web. The International Phonetic Association was formed by a group of French and British language teachers in the late 19th century. The first publication of the International Phonetic Alphabet was published by Paul Passy in 1888.

We selected this alphabet because of the following criteria: IPA is designed to represent one symbol for each verbalizable sound and vice versa in all of the languages in the world. This fact makes this alphabet really international as it can be used in every language and can be easily understood. Also, it can help people - whether they are learning or teaching a new language - by using IPA in a simple and intuitive way [7, 8]. Besides, IPA reduces ambiguity in denoting phonemes and is very useful as it is used in many dictionaries, e.g. dictionaries from UK publishers such as Cambridge, Collins, Longman, Oxford and also dictionaries from the German publisher Langenscheidt. Furthermore other phonetic alphabets are based on IPA, e.g. X-SAMPA.



THE INTERNATIONAL PHONETIC ALPHABET (revised to 2005)

CONSONANTS (PULMONIC) © 2005 IPA

	Bilabial	Labiodental	Dental	Alveolar	Postalveolar	Retroflex	Palatal	Velar	Uvular	Pharyngeal	Glottal
Plosive	p b			t d		ʈ ɖ	c ɟ	k ɡ	q ɢ	▨	ʔ ▨
Nasal	m	ɱ		n		ɳ	ɲ	ŋ	ɴ	▨	▨
Trill	ʙ			r				▨	ʀ		▨
Tap or Flap		ⱱ		ɾ		ɽ		▨			▨
Fricative	ɸ β	f v	θ ð	s z	ʃ ʒ	ʂ ʐ	ç ʝ	x ɣ	χ ʁ	ħ ʕ	h ɦ
Lateral fricative	▨	▨		ɬ ɮ						▨	▨
Approximant		ʋ		ɹ		ɻ	j	ɰ			▨
Lateral approximant	▨	▨		l		ɭ	ʎ	ʟ		▨	▨

Where symbols appear in pairs, the one to the right represents a voiced consonant. Shaded areas denote articulations judged impossible.

CONSONANTS (NON-PULMONIC)

Clicks		Voiced implosives		Ejectives	
				ʼ	Examples:
ʘ	Bilabial	ɓ	Bilabial	pʼ	Bilabial
ǀ	Dental	ɗ	Dental/alveolar	tʼ	Dental/alveolar
ǃ	(Post)alveolar	ʄ	Palatal	kʼ	Velar
ǂ	Palatoalveolar	ɠ	Velar	sʼ	Alveolar fricative
ǁ	Alveolar lateral	ʛ	Uvular		

VOWELS

Front Central Back

Close i • y —— ɨ • ʉ —— ɯ • u

ɪ ʏ ʊ

Close-mid e • ø —— ɘ • ɵ —— ɤ • o

ə

Open-mid ɛ • œ —— ɜ • ɞ —— ʌ • ɔ

æ ɐ

Open a • ɶ —— ɑ • ɒ

Where symbols appear in pairs, the one to the right represents a rounded vowel.

OTHER SYMBOLS

ʍ	Voiceless labial-velar fricative
w	Voiced labial-velar approximant
ɥ	Voiced labial-palatal approximant
ʜ	Voiceless epiglottal fricative
ʢ	Voiced epiglottal fricative
ʡ	Epiglottal plosive
ɕ ʑ	Alveolo-palatal fricatives
ɺ	Voiced alveolar lateral flap
ɧ	Simultaneous ʃ and x

Affricates and double articulations can be represented by two symbols joined by a tie bar if necessary. k͡p t͜s

SUPRASEGMENTALS

ˈ	Primary stress
ˌ	Secondary stress ˌfoʊnəˈtɪʃən
ː	Long eː
ˑ	Half-long eˑ
̆	Extra-short ĕ
|	Minor (foot) group
‖	Major (intonation) group
.	Syllable break ɹi.ækt
‿	Linking (absence of a break)

DIACRITICS Diacritics may be placed above a symbol with a descender, e.g. ŋ̊

̥	Voiceless	n̥ d̥	̤	Breathy voiced	b̤ a̤	̪	Dental	t̪ d̪
̬	Voiced	s̬ t̬	̰	Creaky voiced	b̰ a̰	̺	Apical	t̺ d̺
^{h}	Aspirated	t^h d^h	̼	Linguolabial	t̼ d̼	̻	Laminal	t̻ d̻
̹	More rounded	ɔ̹	^{w}	Labialized	t^w d^w	̃	Nasalized	ẽ
̜	Less rounded	ɔ̜	^{j}	Palatalized	t^j d^j	^{n}	Nasal release	d^n
̟	Advanced	u̟	^{ɣ}	Velarized	$t^ɣ$ $d^ɣ$	^{l}	Lateral release	d^l
̠	Retracted	e̠	^{ʕ}	Pharyngealized	$t^ʕ$ $d^ʕ$	̚	No audible release	d̚
̈	Centralized	ë	̴	Velarized or pharyngealized	ɫ			
̽	Mid-centralized	e̽	̝	Raised	e̝		(ɹ̝ = voiced alveolar fricative)	
̩	Syllabic	n̩	̞	Lowered	e̞		(β̞ = voiced bilabial approximant)	
̯	Non-syllabic	e̯	̘	Advanced Tongue Root	e̘			
˞	Rhoticity	ɚ a˞	̙	Retracted Tongue Root	e̙			

TONES AND WORD ACCENTS

LEVEL			CONTOUR		
e̋ or	˥	Extra high	ě or	˩˥	Rising
é	˦	High	ê	˥˩	Falling
ē	˧	Mid	e᷄	˦˥	High rising
è	˨	Low	e᷅	˩˨	Low rising
ȅ	˩	Extra low	e᷈	˧˦˧	Rising-falling
ꜜ		Downstep	↗		Global rise
ꜛ		Upstep	↘		Global fall

Figure 2.2: IPA chart revised 2005

The International Phonetic Alphabet mainly consists of letters from the Roman alphabet, only if not avoidable new symbols were established (see Figure 2.2). The IPA chart contains vowels, consonants, different intonations and diacritics. For more information about the IPA chart, please visit the official website of the International Phonetic Association [9].

3. Related Work

The ascension of the World Wide Web as a major factor in public life has made communal projects within the Internet crowd popular and successful. An example of a successful crowd project is the Internet encyclopedia Wikipedia [10], which is mainly maintained by unpaid and largely unknown individuals across the world.

Scientists have also tried to harness the power of the Internet community to help solve problems that computers are not able to handle. One large breakthrough was made, when a 15 year old biological puzzle on the way to battle AIDS was solved in just three weeks by amateur computer players [11]. This was done with the help of Foldit [12], a game built at the Center for Game Science [13] at the University of Washington, Seattle. Foldit attempts to predict the structure of a protein by taking advantage of humans' puzzle-solving intuitions and having people play competitively to fold the best proteins. The solutions can than be used for medical purposes.

There are some interesting projects about using the Internet for ASR. But as far as we could find, no one was researching in the area of dictionary creation through crowdsourcing. Three of them had an impact on what we do:

- Peekaboom
- Human Factors during Bootstrapping
- LexLearner

All three studies are further described in the following sections.

3.1 Peekaboom

At Carnegie Mellon University, they research how to harness the computational power of human brains by letting people play online games that actually accomplish a certain goal. They call them *Games with a Purpose* or GWAP [14]. Out of the different games on their website there was one, *Peekaboom* [15], that influenced this work. This game focuses on collecting meta information for images by trying to

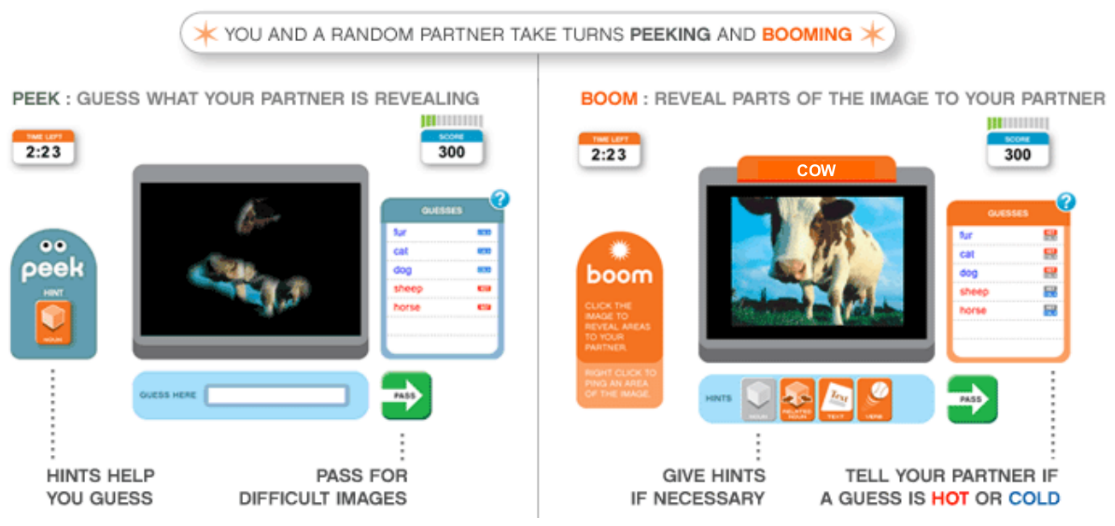

YOU AND A RANDOM PARTNER TAKE TURNS PEEKING AND BOOMING
PEEK : GUESS WHAT YOUR PARTNER IS REVEALING
2:23
300
peek
GUESSES
fur
cat
dog
sheep
horse
PASS
HINTS HELP YOU GUESS
PASS FOR DIFFICULT IMAGES
BOOM : REVEAL PARTS OF THE IMAGE TO YOUR PARTNER
2:23
COW
300
boom
HINTS
PASS
GIVE HINTS IF NECESSARY
TELL YOUR PARTNER IF A GUESS IS HOT OR COLD

Figure 3.1: Game PeekaBoom, where two players find meta information for pictures

determine, which area of the image a keyword refers to. They use images from the Internet and in another game associate keywords with them.

Two random players are paired up and given a word/picture pair. One player is in the role of *Peek*, while the other is *Boom*. Only *Peek* sees the image and the associated word and he or she reveals part of the image so that *Boom* can guess the correct word. This setup can be seen in Figure 3.1.

Points are awarded for the size of the uncovered area, where smaller is better. Also *Peek* can give several predefined hints, such as verb or noun, for which bonus points are awarded. That seems counterintuitive but the purpose of the game is to collect meta information. Knowing if the keyword is a noun or verb is useful for the purpose of the game. After the word is guessed, their roles are reversed and a new image/word pair is given. Players have four and a half minutes to complete as many pairs as possible. Both players get the same amount of points at the end of one session and then get paired with another player, if they want to continue.

Apart from the fun of playing the game, players get on high score lists and can improve their standing there, which is a further incentive to keep playing. That this was a success is documented in the amount of data that was collected. In about 11 months (August 2005 to June 2006) nearly 30.000 users generated over two million pieces of data. On average players spent about 72 minutes on the game in a single session.

To prove that their collected data was accurate and to show what could be done with it, they extracted the area in each image, referenced by the keyword. After a couple of players have used the same image/word pair, it was possible to make a very accurate guess about the exact area the keyword was referring to. This area was calculated from the several areas revealed by different *Boom* players. They could then show, that the collected data from the game was as good or even better in pinpointing the desired object, than bounding boxes made by volunteers. For this a bounding box was calculated, which would encompass all of the aggregated data of the players. On average the two bounding boxes, made by the system and the volunteers, were very close together.

3.2 Human Factors during Bootstrapping

In [16] the effect of the human factor in bootstrapping a pronunciation dictionary was evaluated. It was evaluated if helpers with limited or no linguistic experience could develop a quality pronunciation model in less time than it would take a trained expert to do using conventional methods.

For this purpose a system was designed which presents the user with a written version of a word of the target language, a prediction for a pronunciation, made by the system, and the ability to sound out each phoneme of the prediction. When sounding out the phonemes a prerecorded sound for the selected phoneme is played back to the user. Then the user can decide, if the sequence of predicted phonemes is correct. By altering the sequence, the user also alters the prediction procedures of the system, making them more accurate.

This process speeds up the process of creating a substantial pronunciation dictionary considerably, compared to conventional methods. This is possible, because the adapting prediction process of the system cuts down significantly on the number of corrections per word a user has to make. Additionally it is possible for unskilled users to produce high quality dictionaries, rivaling those made by linguistic experts.

3.3 LexLearner

A very similar method to [16], is the Lexicon Learner (LexLearner). LexLearner is part of the Rapid Language Adaption Toolkit [2]. The LexLearner component is responsible for generating word pronunciations with the aid of a user. Users do not have to be linguistic experts but need to be proficient in the target language.

In an initialization stage, rules to predict pronunciations are seeded by asking the user to match phonemes with their most commonly associated grapheme. These rules are then used to present the user with a pronunciation prediction for a word.

As suggested by [16], these predictions are accompanied by a wavefile which is generated through a phoneme-concatenation synthesizer. The user corrects the pronunciation and the system updates the rules accordingly.

Further details are described in [17].

3.4 Summary

These three examples have shown us the following things:

- Internet users can generate useful information
- Internet users will participate, given the right incentive
- The aggregate information can be calculated to present a higher value, than the single pieces

- Human listening and comparing abilities enable them to correct phoneme sequences for written words
- Linguistically unskilled users can produce pronunciation dictionaries rivaling professional work

With that in mind, we would then start to build our own system, which would incorporate some of the concepts and ideas, reviewed in this chapter.

4. Analysis of Requirements

4.1 Requirements

When implementing a system one could define two distinct properties - desirable and obligatory. The obligatory properties are the basic requirements which we have to meet. The desirable properties are what makes the system user friendly and overall a satisfactory experience.

The essential requirements for our system are the following: Inexpensive, robust, simple.

inexpensive

To make the acquisition of pronunciations worthwhile, we need to have an **inexpensive** system. Otherwise there would be no advantage to building a new system compared to hiring professional linguists to do the work. We meet this requirement by using open source software wherever possible. Everything else we implement ourselves around the existing software.

robust

To make the system as **robust** as possible, we use established and well-proven components. The system is supposed to run as unimpeded as possible and we want to avoid any teething troubles within the components themselves. This leads us to the last requirement - simplicity.

simple

To design a robust system goes hand in hand with **simple**. A simple, straight forward approach leaves a lot less room for mistakes or misunderstandings with the user than a complex system. Thus a clearcut user interface and tasks make for a robust as well as a simple and user friendly system. Complex features like registration, identification and excessive guidance of the user are avoided as much as possible or done automatically.

We also need other qualities in our system. But these are secondary to the afore mentioned ones because they are not essential. We want to make use of the most common aspects of every person's knowledge of speech and make the users use his abilities to produce high quality pronunciation data for scientific use. We derive from this, that our system needs an intuitive user interface and work flow. This removes the necessity of training users beforehand and makes them more comfortable with the system.

4.2 User analysis

accessible

As this work should prove the viability of a gaming approach to gathering dictionary data for ASR systems with the help of online users, we need an easily **accessible** system. That means that we should aim on not excluding any participants but rather filter inappropriate data afterwards. Any kind of restriction on our side might bar users from the experiment. The only restrictions that we want to lay down are those, that come with the technology. So anyone with an Internet connection and a fairly modern browser should be able to participate.

For this work we wanted to establish four different categories which should encompass all of the anticipated users of our experiment:

- Onlooker
- Gamer
- high incentive Gamer (hiG)
- Troll

For classifying the behavior of our participants, we proceed on the assumption that the game will provide a few words for the users to translate before offering any kind of results, review or feedback. A **session** is a group of words, which need to be translated to phonemes. Each session ends with a review of the users results and a comparison with other results and the reference pronunciation. For users without any experience, the first one or two sessions might be practice, since they do not get feedback until they have finished the first session.

Onlooker

We label those users **Onlooker**s, that were asked to play the game, or have somehow heard of it and test it, out of mild curiosity or a feeling of obligation. This group does not try very hard and does not participate in the game for more than one or two sessions, sometimes even leaving in the middle of a session. We do not differentiate between those who find the exercise too demanding or those whose interest we could not engage. We expected this group to be the largest in the study but input from these users probably does not yield too much valuable information, since they never get past the learning stage.

Gamer

In our category of **Gamer** we classify those users who finished the learning stage of the game. They are open to a new experience and are actively trying to get the best results. This means getting to know the controls and looking at their own results to improve their understanding of the problems. To do that, the Gamer has to complete at least two cycles of the experiment and is shown the results from which to learn. If he or she continues to play more than two sessions, they are then moved up from Onlooker to Gamer. The results expected from this group are what we are aiming for. The participants are actively interested and try for the best results. We hoped that this is the largest group providing us with enough data to work with.

hiG

The **hiG** are the users who for some reason or other stay with the experiment for a long time. This reason could be a personal involvement with the experiment or a fascination with the game. We expected to see some people in this class. Since this experiment is not a full-blown game, the long term incentives are limited or nonexistent. We therefore expected this group to be rather small. Depending on their reason for participating excessively, their results may vary. The results might range from being of good or even outstanding quality from fascinated users, to being below average if the incentive is a heavy sense of obligation and no real enthusiasm for the game. It is also possible, that these users do not improve their performance after a certain point, but just finish faster. This is conceivable, since the task itself does not, at this early stage, offer a lot of variety.

The last category that we want to establish is the one of the **Troll**. The term is used on the Internet for a person making disruptive and unreasonable remarks in online discussions [18]. In our case this should encompass everyone whose intention is not in adhering to the rules and playing the game as it is meant to be played. Since our experiment is open to the Internet community without any required identification of the participants, we expected this group to be quit large. Input from this group yields no valuable information and can be treated as noise. As we conduct our experiments within a rather limited group of people, we periodically check the results. Results or usernames that are offensive or indecent are manually renamed or removed. This is just a precaution to not offend other participants, since they might expect this kind of behavior on the Internet in general but not in a university study. We do try not to alter the input in itself but just change the visible parts to something less harmful. Troll

While we expect to see all types of participants in our experiment, we hope and aim specifically for the *Gamer* and the fascinated *hiG*.

Another assumption on our part is, that our expected user has little or no prior knowledge of creating linguistic data. Furthermore pronunciation alphabets like the International Phonetic Alphabet (IPA) [9] can not be assumed to be known to the users. This leads to the conclusion, that our system should either train all users to a certain degree or better yet, be so **intuitive** that no prior knowledge is needed. Optimally we can design a system where the user creates the linguistic data without needing to understand the concept behind it. intuitive

To summarize our results from this phase, we established the following design rules:

- Accessibility - no technological or knowledge barriers
- Ability to cope with input ranging from well-intentioned to mischievous due to wide user range
- Intuitive user interface

4.3 Analyzing user task

As our goal was to create a game, which encourages users to produce pronunciation data, there are different aspects to be considered. First of all the game needs to appeal to a large group and over a period of time, so that it runs without being constantly advertised. Which means that it can neither be too **difficult** nor too easy, as extremes probably repel too many users. difficult

The task should also be moderately **taxing** so as to be achievable for many users. taxing It should not be necessary for the users to have any prior knowledge, or go through an extended phase of learning, to complete the task. This should then result in an incentive inherent **incentive**, since it is not too complicated but moderately difficult.

In summary we want to create a task with the following requirements:

- Moderately difficult
- Moderately taxing without prior knowledge
- Inherent incentive to keep trying

Taking inspiration from the work described in Section 3.2, we established the following task as the core to our system:

Users are presented with a word from the target language and a keyboard comprised of phonetic symbols. Using these symbols the user is supposed to build a pronunciation for a word. Since we do not expect the users to know the symbols, he or she can have the symbol string be read aloud by a speech synthesis software. Comparing his or her knowledge of how the word is supposed to sound with what is generated from the chosen keys, the user should then try and find the best representation of the word.

Listening and comparing is not too challenging. But finding the correct string of symbols should prove challenging enough. Depending on the kind of symbols on the keys, representing the phonemes, the user can be aided or hindered. As explained in Section 2.4 we decided on using the symbols of the International Phonetic Alphabet (IPA) [9]. IPA or variations of IPA are found in many dictionaries as pronunciations for each word. This means, that the users might remember seeing the symbols and at best, have always wanted to know what they mean. By building their own strings out of these symbols they can now find out, which will be an incentive to try our game. At worst, the similarity of the symbols to the letters of the Latin alphabet, makes it easier for the user to make a first guess. The alphabet for the keys is interchangeable if other representations are more appropriate for a certain language.

4.4 Summary

The most important conclusions of this chapter are the requirements for the system:

- Low cost
- Robust design
- Simplicity

As well as the secondary qualities:

- accessible

- Intuitive layout of interface and work flow
- able to handle wide variety of input
- No prior knowledge

After establishing these requirements, the next chapter focuses on the design decisions of the interface and the implementation.

5. Design

In Chapter 4 we defined the requirements for the system. We then established the targeted group of users, together with a consideration about motivation, usage, and availability. We also analyzed the actual task that we can give the users with respect to difficulty, workability, and incentive. With this information established, the design process was divided into three distinct phases. In the first phase a rough overview of how the system is going to look and feel was established. This was visualized to describe the main points of the systems design. Starting with the general interface design, decisions about the backend could then be made in phase two. And in the last phase the more specific design decisions are discussed.

5.1 General Interface Design

The general design idea as shown in Figure 5.1 was a phonetic keyboard with which the user should transform a given word into a string of phonetic symbols.

As we have discussed in Section 4.3, the expected user has little to no prior knowledge about phonetic symbols. To help the user building the correct string of symbols, we add a *PLAY* button to the interface. This button feeds the string of phonetic symbols, which are selected at that point, to a speech synthesis program. The resulting synthesized audio file is then played back to the user. The user must decide if the audio, generated from his or her selected symbols, matches what he or she knows to be a correct pronunciation of the given word.

If audio and pronunciation are not similar enough, the user should try to adjust the symbol string accordingly. To help adjusting the symbol string some buttons are given to navigate within the string. This is accomplished with the help of *FORWARD* and *BACKWARD* buttons with which to navigate a cursor within the string of phonetic symbols. To delete a single symbol to the left of the cursor the *DEL* button was added.

A user might also choose to *ACCEPT* or *SKIP* a given word, if he or she has either found an apparently correct representation or can not create a good audio feedback.

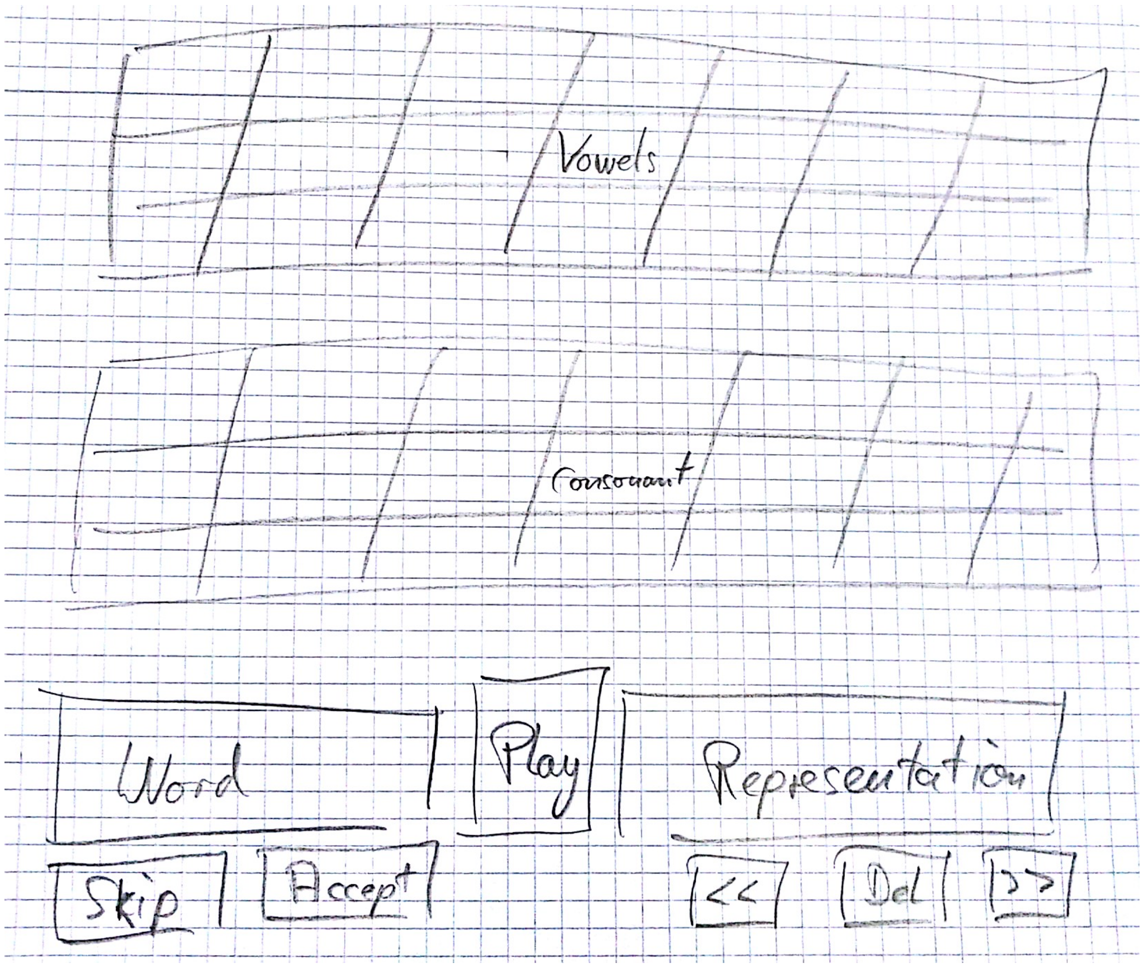


Figure 5.1: Sketch of Keynounce user interface

The keyboard for the symbols is shown in Figure 5.1 as being divided into two groups. As we expect users untrained in linguistics, we decided to have the symbols split into groups whose meaning is easily accessible to everyone.

We also need some kind of tutorial to give first time users a starting point. Giving no information at the start might discourage some users from trying. Returning users might also be able to use the tutorial as a point of reference to aspects of the game, which they do not remember correctly.

5.2 Backend Implementation

We have formulated in Section 4.2 that there should be little to no technological barriers imposed by the program. Our requirements are:

- accessible on the World Wide Web
- usable for a majority of World Wide Web users
- no additional software for users to install on their hardware
- small demands on the user's hardware
- graphical user interface (GUI)
- ability to play sounds
- ability to quickly change interface depending on user input
- open source or freeware components to reduce costs
- reduced loading time during gameplay
- database interaction

To meet the requirements of easy access for a majority of World Wide Web users without additional software installation and making small demands on the user's hardware, we looked at what was currently available on the web. What is mainly used today to create small online games and graphical user interfaces, as well as dynamic commercials on the Internet are Java [19] and Flash [20]. Both are widely used and most Internet users have everything that is required already installed on their machines.

We decided to use Flash and Actionscript, since it is an almost perfect fit to our needs. Flash was designed to bring moving pictures, sounds and interactivity to web pages. It supports bidirectional streaming of audio content and supports user input by mouse and keyboard. The corresponding Flash Player is already installed on many personal computers, since a lot of motion picture content on the Internet is provided with Flash.

The chosen synthesizer was eSpeak [21] because it provides fast and easily adapted text synthesis and moderate quality of speech. The slightly robotic quality of speech should help the user to accept slight imperfections resulting from a lack of prosody and missing accentuation. As long as a correct combination of sounds is there, a user should accept the word as fine. A high quality synthesis would suggest to the user that the lack of prosody is his fault and he might try to remedy that.

5.3 Specific Interface Design

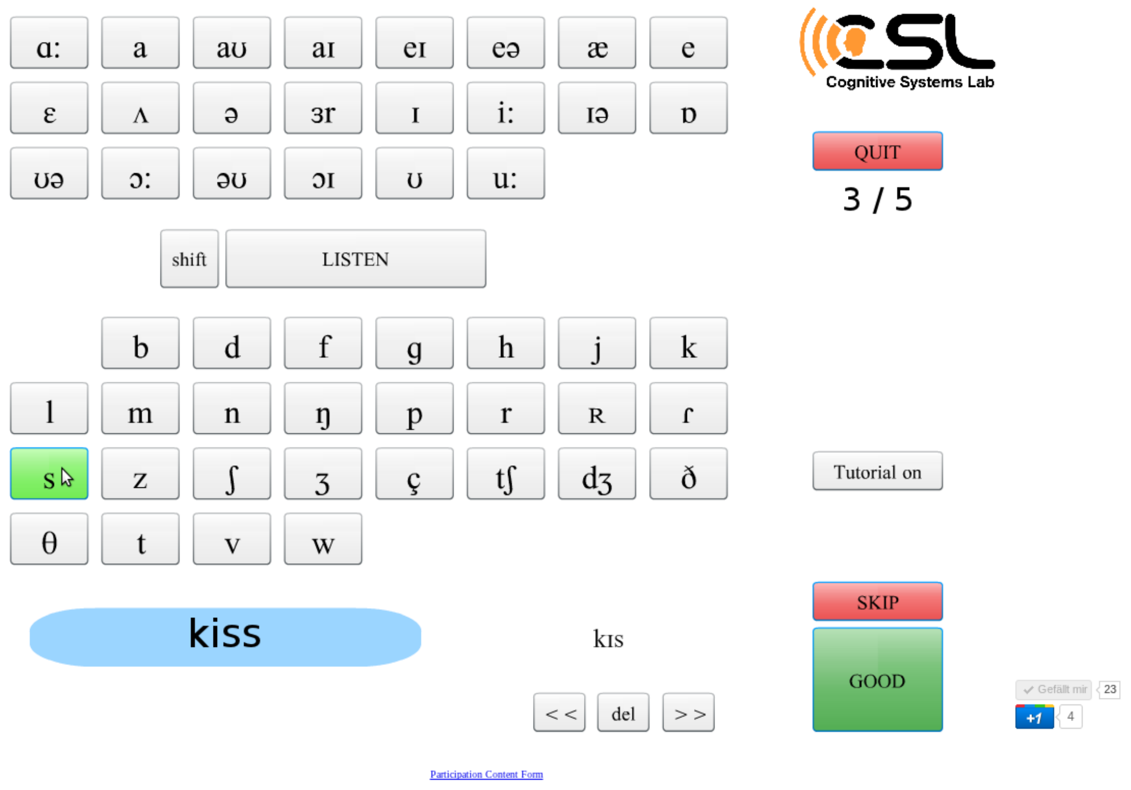


Figure 5.2: Keynounce user interface

Following the general design decisions of Section 5.1, we built a specific user interface which is shown in Figure 5.2. This specific design varies slightly from the design sketch in Figure 5.1 but the overall design was adopted. This design was established after some preliminary tests. In this chapter, we discuss the changes and additions that were made to the design sketch. We also explain why those decisions were made.

The most important part of the finished user interface is the symbol keyboard. As

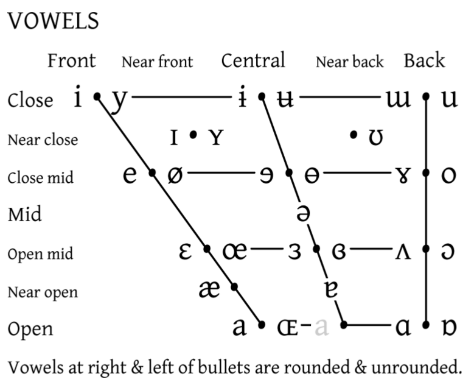
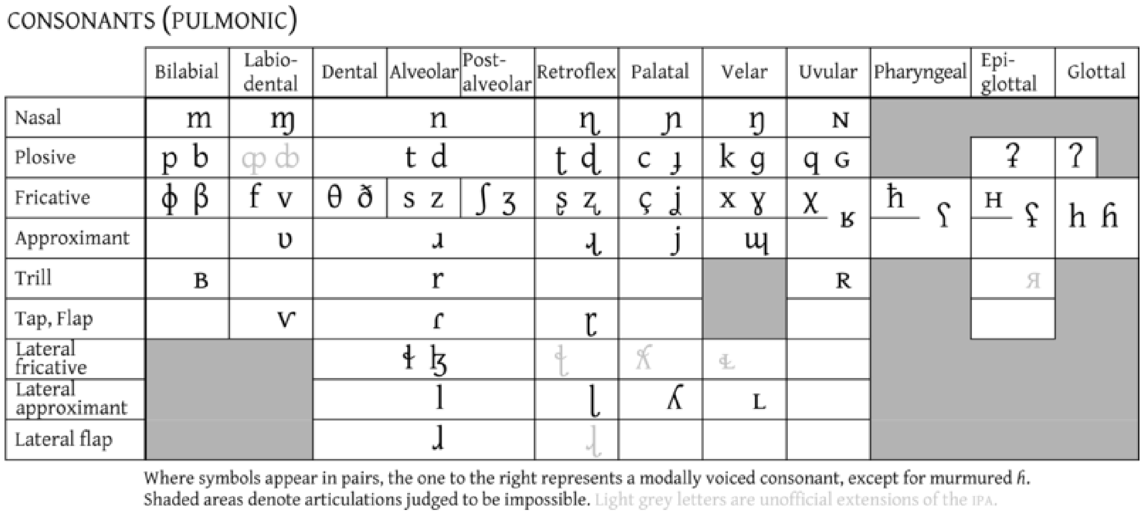


Figure 5.3: IPA vowel chart

discussed, it is divided into two groups. The division into vowels and consonants is chosen in light of our premise of untrained users. In the IPA tables the different sounds are also classified as consonants and vowels and are then mapped according


CONSONANTS (PULMONIC)

	Bilabial	Labio-dental	Dental	Alveolar	Post-alveolar	Retroflex	Palatal	Velar	Uvular	Pharyngeal	Epi-glottal	Glottal
Nasal	m	ɱ		n		ɳ	ɲ	ŋ	ɴ			
Plosive	p b	ȹ ȸ		t d		ʈ ɖ	c ɟ	k ɡ	q ɢ		ʡ	ʔ
Fricative	ɸ β	f v	θ ð	s z	ʃ ʒ	ʂ ʐ	ç ʝ	x ɣ	χ ʁ	ħ ʕ	ʜ ʢ	h ɦ
Approximant		ʋ		ɹ		ɻ	j	ɰ				
Trill	ʙ			r					ʀ		я	
Tap, Flap		ⱱ		ɾ		ɽ						
Lateral fricative				ɬ ɮ		ꞎ	𝼆	𝼄				
Lateral approximant				l		ɭ	ʎ	ʟ				
Lateral flap				ɺ		𝼈						

Where symbols appear in pairs, the one to the right represents a modally voiced consonant, except for murmured ɦ.
Shaded areas denote articulations judged to be impossible. Light grey letters are unofficial extensions of the IPA.

Figure 5.4: IPA consonant chart

to the part of the vocal tract that creates the sounds. This mapping is shown in Figure 5.3 and Figure 5.4.

Although this is a good classification for the knowledgeable user, it would need special explanations to convey the concept to an uninformed user. For this reason we have decided to arrange the symbols according to their corresponding letters in the target languages.

As a user without any background in linguistics, has no further clue to what the symbols stand for, the resemblance of the symbols to the letters of the Latin alphabet is the only recognizable thing. One of the prerequisites for our interface was that it needs to be intuitive, so we decided to use this resemblance as a clue to how the symbols are arranged.

For the vowels we chose to order them A-E-I-O-U and for the consonants we chose the normal alphabetical order of their associated letters. The buttons where the symbol itself matches the Latin letter and the expected sound in the target language thus became the focal points for our layout. The user probably tries out the most familiar symbols first. If the resulting audio does not meet the user's expectations, our explanation of how the symbols are arranged should lead him or her to try the symbols close to the first guess. The symbols in close proximity were arranged so, as to represent variations of the same sound. The further two symbols are from each other, the less similar their sounds. This order was established on a subjective level with a couple of lab workers. Therefore this should be as intuitive an arrangement of sounds as possible.

Changes that have been made to the first design sketch include the repositioning of the *SKIP* and *ACCEPT* buttons. They are now found in the lower right corner as that seemed to be the more intuitive place to search for finishing keys. The text *ACCEPT* has also been changed to *GOOD* to stress that the user marks the current string as a good representation.

Furthermore the *LISTEN* button was moved between the two symbol blocks to shorten movement of the mouse. As can be seen in Figure 5.5, the maximum distance

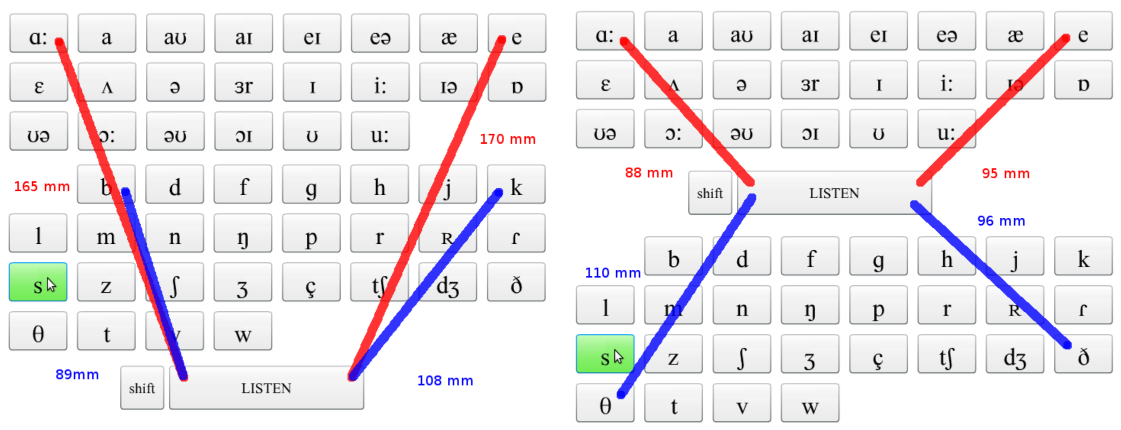


Figure 5.5: Longest distances at different positions of LISTEN button

between the *LISTEN* button and a consonant key is the same at 110 mm, whether the button is at the bottom or in the middle. When positioned at the lower part of the interface, the maximum distance between it and the vowel keys almost doubles from 95 mm to 170 mm. Although the actual distance obviously changes with the size and resolution of the users screen, the proportions are always the same. It was an obvious choice to move the button to a central position, since we expect the vowels to be more problematic to get right. This change does make the interface less strenuous to use in longer sessions.

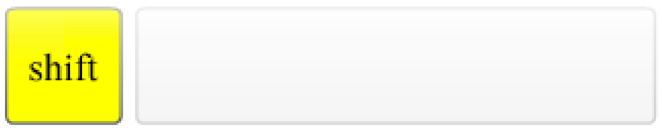


Figure 5.6: Highlighted SHIFT button

Next to the button for listening to the current string, another button was added. Preliminary testing showed, that fine tuning the symbol string resulted in a cumbersome repetition of add - listen - delete. This is due to the fact, that the users have found an approximation very quickly, but they were not satisfied with it. To get it just right, mostly one or two symbols had to be replaced. But to get the right nuances, a user sometimes has to listen to the sound of two or three versions alternatly and repeatedly. To alleviate this procedure the *SHIFT* button was installed. This button can be toggled on screen as seen in Figure 5.6, or held on the physical keyboard.

Clicking on a symbol while having activated *SHIFT* results in the instant addition of that symbol to the string and playing of the resulting audio. The symbol itself is temporarily added to the string at the left side of the current cursor position. The symbols are not added permanently. So different symbols can be tried with the current string in rapid succession. As a bonus the button also helps to get a feel for the sounds. The vowel sounds can be played by themselves. In case of the consonants a vowel has to be added first for the synthesizer to create a recognizable sound.

After our usability tests in the lab, we added a *QUIT* button to the interface. Some of the users wanted to end a session prematurely and still get the results for the finished words, so they repeatedly hit either *ACCEPT* or *SKIP*. Both cases can change the data, that we collect. By accepting unfinished words, the users drop false data into the database. To skip over words does not seem that problematic, but might pose problems in the future, when skipped words might be culled from the data as suggested in Chapter 9.

Below the option to quit the session, we have installed a countdown that shows the user how many words he or she has already finished and how many are still left in the session. This might incite a user to finish one or two more words, since they are the final ones.

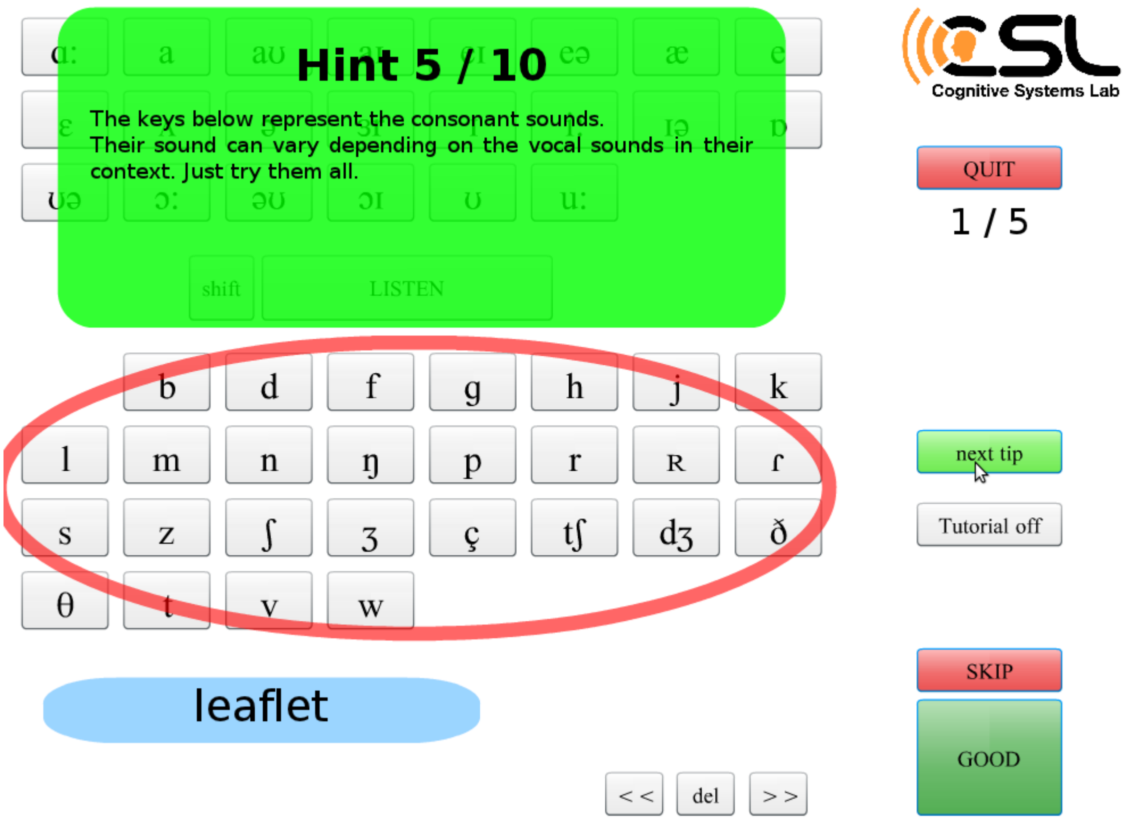
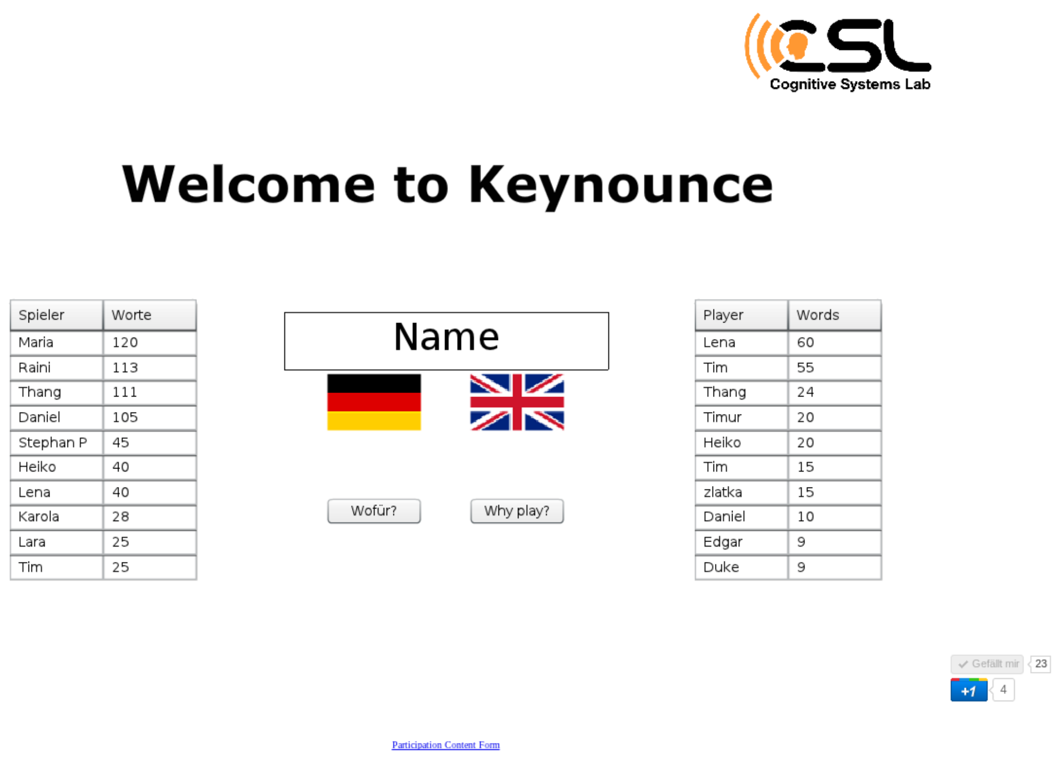


Figure 5.7: Tutorial in Keynounce

As stated in Section 5.1, a tutorial was implemented. It shows prominently above the standard interface at a user's first interaction with the system. It can be seen in Figure 5.7. For a returning user, the tutorial is automatically hidden, but he or she can choose to have a look at it. The tutorial gives hints on how to use the interface and the discussed part of the interface is highlighted with a red ring around it. To the right of the screen, the user can switch to the next hint or turn the tutorial off. Once the tutorial is turned off, everything is hidden and only the button to turn it back on stays on the interface. With that, the user can always go back to the tutorial and refresh his or her memory about certain aspects.

Since this experiment would be mainly based on the help of friends and aquaintances, we tried to connect with social networks to enlarge the potential userbase. For that we just added the Facebook LIKE button and Google's 1+ button. With these, our initial users have an easy way of propagating our experiment to their friends. This might result in more users.



Spieler	Worte
Maria	120
Raini	113
Thang	111
Daniel	105
Stephan P	45
Heiko	40
Lena	40
Karola	28
Lara	25
Tim	25

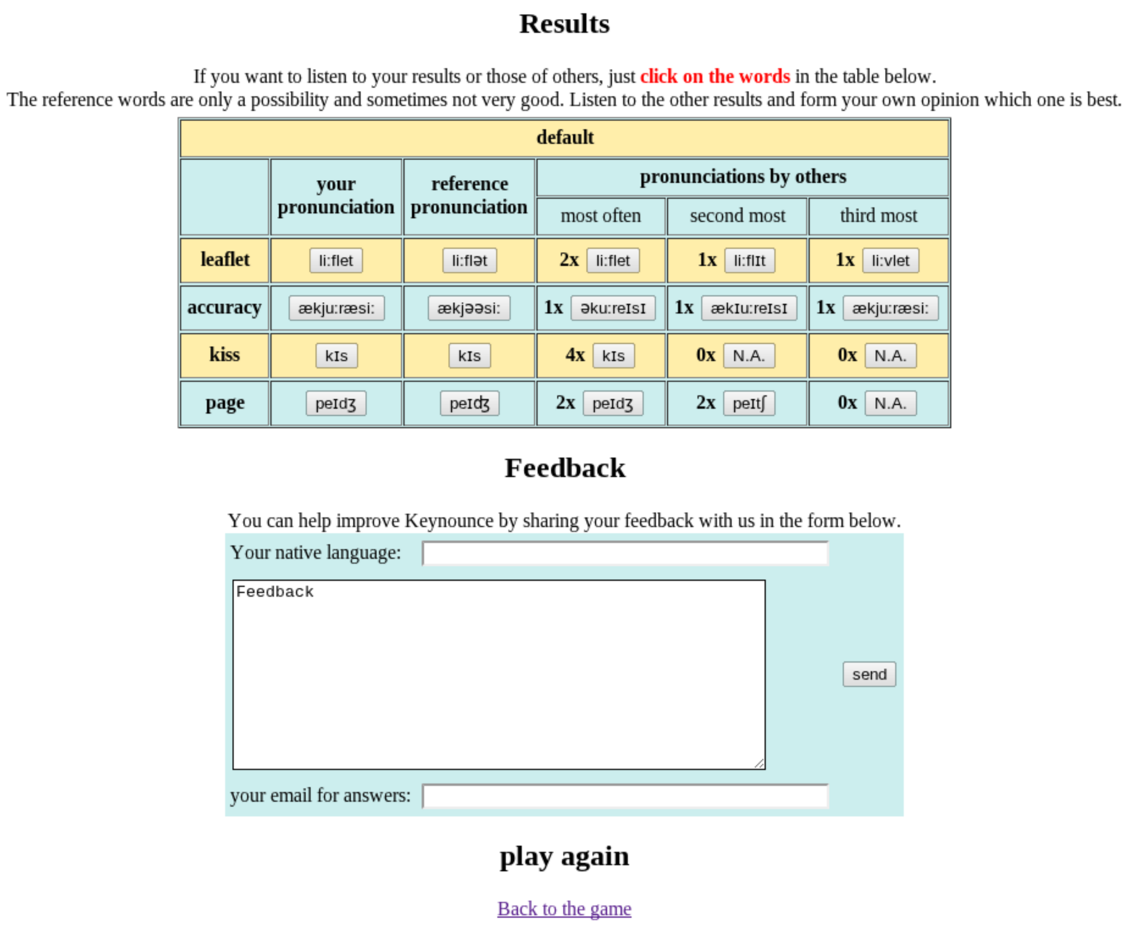
Player	Words
Lena	60
Tim	55
Thang	24
Timur	20
Heiko	20
Tim	15
zlatka	15
Daniel	10
Edgar	9
Duke	9

Figure 5.8: Welcome screen in Keynounce

Another adjustment to the interface was to create a more complete feeling with a start screen and a final screen. The screen at the beginning, as shown in Figure 5.8, allows a choice between the two available languages German and English and explains again why this game was created. It also features a ranking system that shows the names of participants and how many words they have submitted. This creates an incentive for the players since they either want to see their name to appear in that top 10 list or because they reminded of what effort other people put in.

The final screen (Figure 5.9) lists the submitted strings of the current player. It also shows the reference string as provided by the GlobalPhone dictionary and the top 3 submissions by other players. The top 3 are chosen by the simple way of counting the number of matching strings for one word. The string that was submitted most often by all players is the number 1 seat. All these strings can be listened to by the player and are not rated in any way by the program. This provides the player with feedback to learn from. He or she can decide if other versions submitted either by the reference or other players were superior to their own and why. If users deem all other versions to be inferior to their own, they might feel motivated to submit more words.

By not making any judgments ourselves, we keep the game clear of too much guiding influence. Both the reference string and the top 3 variants are stressed to be just variants that somebody else thought were ok. We leave it to the players to compare



Results

If you want to listen to your results or those of others, just **click on the words** in the table below.
The reference words are only a possibility and sometimes not very good. Listen to the other results and form your own opinion which one is best.

default									
	your pronunciation	**reference pronunciation**	**pronunciations by others**						
			most often		second most		third most		
leaflet	li:flet	li:flət	**2x**	li:flet	**1x**	li:flɪt	**1x**	li:vlet	
accuracy	ækju:ræsi:	ækjəəsi:	**1x**	əku:reɪsɪ	**1x**	ækɪu:reɪsɪ	**1x**	ækju:ræsi:	
kiss	kɪs	kɪs	**4x**	kɪs	**0x**	N.A.	**0x**	N.A.	
page	peɪdʒ	peɪʤ	**2x**	peɪdʒ	**2x**	peɪtʃ	**0x**	N.A.	

Feedback

You can help improve Keynounce by sharing your feedback with us in the form below.

Your native language:

Feedback

send

your email for answers:

play again

Back to the game

Figure 5.9: Final screen in Keynounce

and draw their own conclusions. Since the idea of the game was to gather information about unknown or little known languages, we might not even be able to judge new entries in the future except on how many other users submitted the exact same result. At the current stage we know exactly what the established answers are, for all words we have chosen. It would therefor be possible to give exact evaluations for a specific user's input. But this is only possible, because we use words from two well documented languages. Implementing some kind of scoring, accessible to the user during the experiment, based on established results would compromise our results for future work with undocumented languages.

5.4 Summary

In this chapter we have presented our preliminary ideas for the design of Keynounce and it's basic functionality. Afterwards, in compliance with the requirements postulated in Section 4.2 our implementation choice are explained. They are in short:

- Adobe Flash and Actionscript
- eSpeak as synthesizer
- Arrangement of IPA symbols (vowel and consonant placement)
- Userfriendly placement of buttons
- Tutorial placement
- Integration of facebook and google+
- Welcome screen with ranking
- Final screen with feedback

After making these implementation choices, we are explaining in the next chapter, what experiments we did and their success.

6. Experimental Setup

In this chapter we describe two experiments we set up to gather data with Keynounce.

First of all we tried setting up a reduced version of Keynounce with Amazon's Mechanical Turk platform. It is a very quick way to reach a large userbase at very affordable prices. This approach can be reviewed in Section 6.1

After mTurk we used the full version of Keynounce and advertised the next experiment with friends and colleagues. While definitely cheaper than mTurk, this approach is comparatively slow. More detailed information can be found in Section 6.2

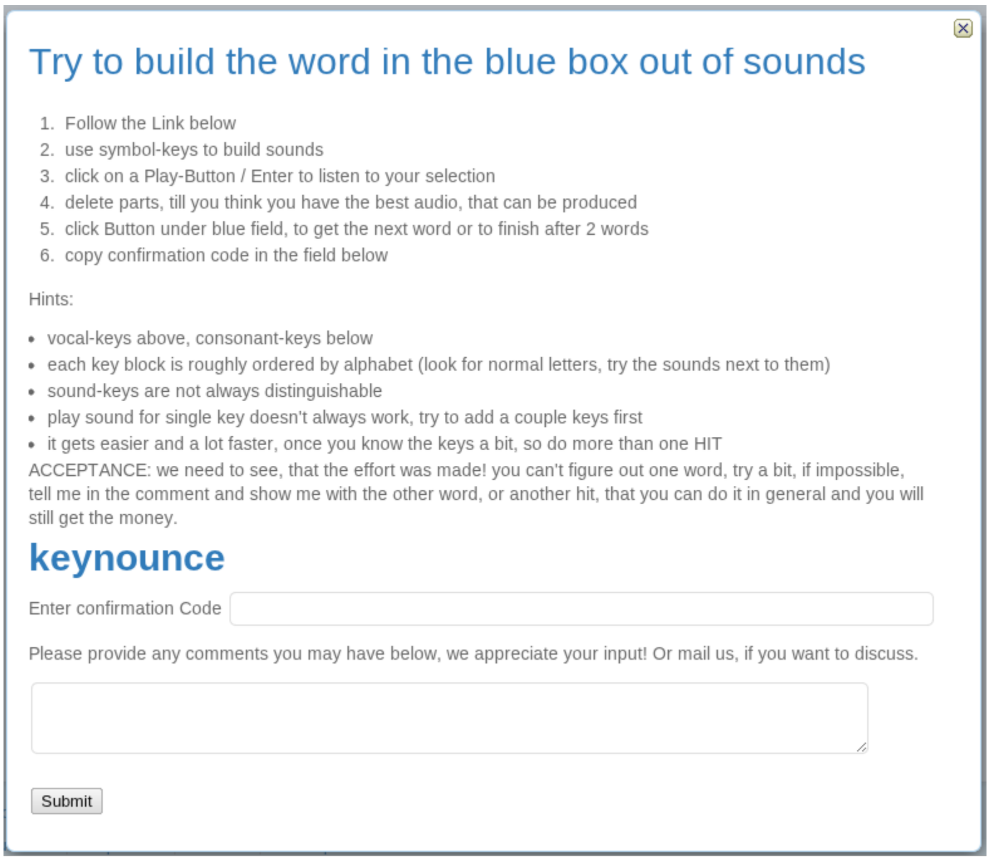
The conclusions for all experiments in this chapter are summarized in Section 6.3

6.1 Experiment on Amazon Mechanical Turk

The experiment on Amazon's Mechanical Turk was set up on 10 May 2011. We used the rather limited requester website provided by Amazon. This was due to the fact, that the more advanced forms to create HITs by API were barely documented, at that time. Through the requester website we created a short document (Figure 6.1) detailing to the turkers that we wanted them to transform two written words to their phonetic spelling. We also gave some hints on how to complete the task. They were then required to click on a link opening another tab with the Keynounce website. This was necessary because Amazon had made it impossible to use external web pages through the requesters website.

The Keynounce version used in this experiment lacked some of the *gaming qualities*, that we discussed in Section 5.3. More specifically the following parts had been removed from the interface:

- Start Screen
- Tutorial



Try to build the word in the blue box out of sounds

1. Follow the Link below
2. use symbol-keys to build sounds
3. click on a Play-Button / Enter to listen to your selection
4. delete parts, till you think you have the best audio, that can be produced
5. click Button under blue field, to get the next word or to finish after 2 words
6. copy confirmation code in the field below

Hints:

- vocal-keys above, consonant-keys below
- each key block is roughly ordered by alphabet (look for normal letters, try the sounds next to them)
- sound-keys are not always distinguishable
- play sound for single key doesn't always work, try to add a couple keys first
- it gets easier and a lot faster, once you know the keys a bit, so do more than one HIT

ACCEPTANCE: we need to see, that the effort was made! you can't figure out one word, try a bit, if impossible, tell me in the comment and show me with the other word, or another hit, that you can do it in general and you will still get the money.

keynounce

Enter confirmation Code

Please provide any comments you may have below, we appreciate your input! Or mail us, if you want to discuss.

Figure 6.1: HIT on Amazon Mechanical Turk

- Skip Button
- End Button
- Social Network Buttons

This was done, because the HITs in mTurk are supposed to be short, clear and to the point. We had given all the hints on the starting page of the HIT itself, which could be read by the turkers before starting the task. The buttons to prematurely end the session or skip a word were removed, because this was not an option, we wanted to give the turkers. The same was true for the social networks.

Additionally, the final screen was modified to show a short “Thank You” message and an authentication code for later verification with mTurk. The turkers were supposed to paste this code into an input field on the starting form of the HIT. With this authentication we were able to match the user in our database to the turker and could thus grant or reject his or her work. This was necessary to verify, that the turkers had actually done any work.

As explained in Section 1.2, we have picked 100 random words from the English CMU dict. Since the requester can chose in advance, we decided to get 10 individual answers for each word and thus 1000 phoneme strings in total. This number was picked, since it offered a good first impression on the work of the turkers without taxing our monetary resources overmuch.

We created 50 HITs with 2 words each. Amazon services also guaranteed, that a single user could not work on two assignments of an identical HIT. This means, that users could finish any number of words between 2 and 100 without being able to work on the same word twice.

Each assignment was awarded with $0.03 totaling to $15.00 and $2.50 in Amazon fees.

Since we did not use the more advanced API of the Mechanical Turk services, we had to approve assignments manually. We did not simply approve every finished HIT, because a large number of turkers refused to do any work and just gave back the authentication code. This meant we had to supervise the results more closely than expected.

As can be seen and is explained in the results (Section 7.1), this experiment did not produce good results.

6.2 Free Game Experiment

After the mTurk experiment did not meet our expectations about the quality of the results, we decided to engage volunteers in our next experiment. To engage the users interest, we used the full interface, as described in Section 5.3. We expected volunteers to have their focus on helping and maximizing their per minute output.

Differing from the mTurk experiment, we decided to split the experiment and provide users the choice between German and English words. This was motivated by the fact, that most of our users first language is German. Due to that we picked another 100 random words from the German GlobalPhone database. We also added a choice of language to the starting screen. The language would then be set for a batch of words and could be changed after each batch.

The batch size was increased from two words in the mTurk experiment to five for the volunteers. This way users would be implicitly tasked to finish at least five words. To alleviate this, we gave them the option to skip single words or abort a batch prematurely. This way we hoped to get more results from users, who would just "try it once".

We asked colleagues working at the lab, friends and acquaintances to help with this experiment. We also encouraged everyone to tell other people about Keynounce. This aspect was helped by the social network buttons we implemented. We had 21 Facebook "likes" and 5 "1+" from Google+. Overall this resulted in 224 participants.

As with the mTurk experiment we had no real safeguards against trolling or other abuse of our experiment. With the idea, that Keynounce would one day be an independent web application to generate data, we reasoned that there would be some noise anyway. As discussed in Section 4.1 the system is able to handle this.

We only interceded once, when someone tried to get "A.Hitler" on the leader board by just submitting one-phoneme words. We did this to prevent other volunteers from taking offense. Other than that we accepted all input and let the experiment run unsupervised.

6.3 Summary

In this chapter we described the two experimental setups we have built, as well as the different configurations we had to apply to the interface. The results of these experiments are presented and discussed in detail in the following Chapter 7.

7. Results

In this chapter we delineate the differences of the experiment with mTurk and the voluntary Keynounce experiment as described in the previous chapter. We also analyze in detail the results of the voluntary Keynounce experiment.

7.1 mTurk

One of the main differences between the mTurk and the volunteer experiment was that the mTurk experiment had a fixed amount of results we would accept from the mTurk workers. This was due to the fact, that we payed for the work and specified a limited number of results per word before the experiment. The volunteer game experiment had no fixed number of entries per word and was more of an open call. We did not know, how many participants we could expect. So we did not restrict their number in advance.

The mTurk experiment was also much faster in getting results. We initiated the experiment on May 10th and stopped on May 23rd with 387 approved and 531 rejected assignments, as can be seen in Figure 7.1. These assignments generated 1,902 words in total but 1,062 of these where spam. Most of the worthwhile input was generated on days 1-3, 7 and 8. We did not get any attention on days 4-6, probably because it was a weekend. The surge of spam answers on day 2 was probably a testing of our assignments by malicious users through dummy or robot accounts. After these were rejected, the spam did decrease somewhat.

In comparison the volunteer experiment which was started on 10/25/2011, generated 1,595 words until there was a snapshot taken on 04/17/2012. The participation is depicted in Figure 7.2. After a month of being online, 89% of the final number of words had been generated. Some people learned of the experiment later on, or returned to it, but the significant amount of work (1,417 words) was done within 30 days. This period of activity could probably be extended by promoting Keynounce over a longer period of time. Either that, or publishing Keynounce on a more frequented website would generate a lot more input.

There was also a difference in the average time spent on completing the pronunciation for a word, as can be seen in Figure 7.3. With an average of 74 seconds, the amount

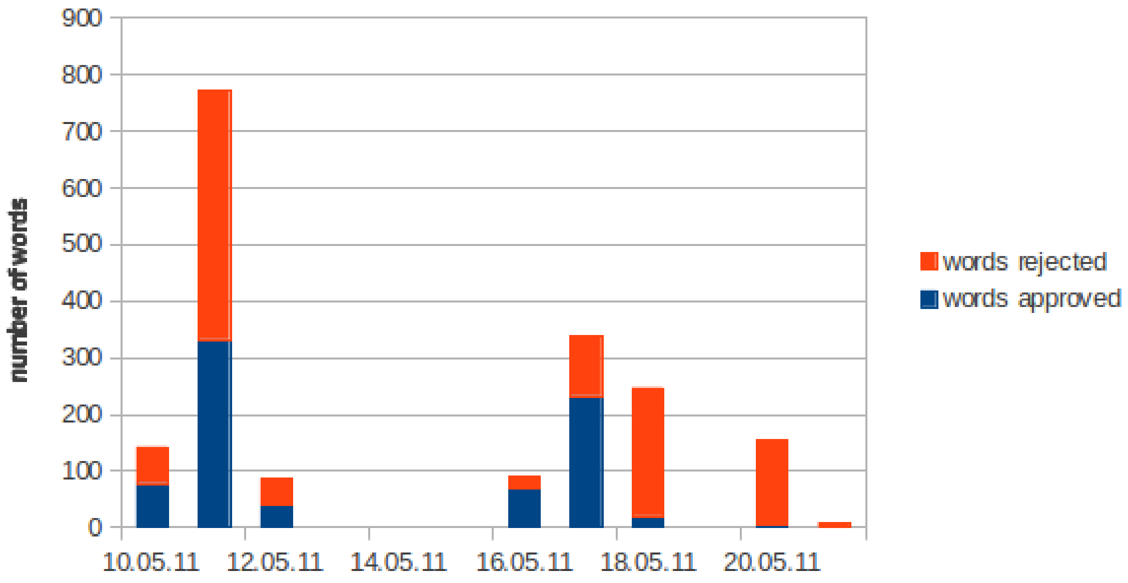


Figure 7.1: Number of words per day by Turkers

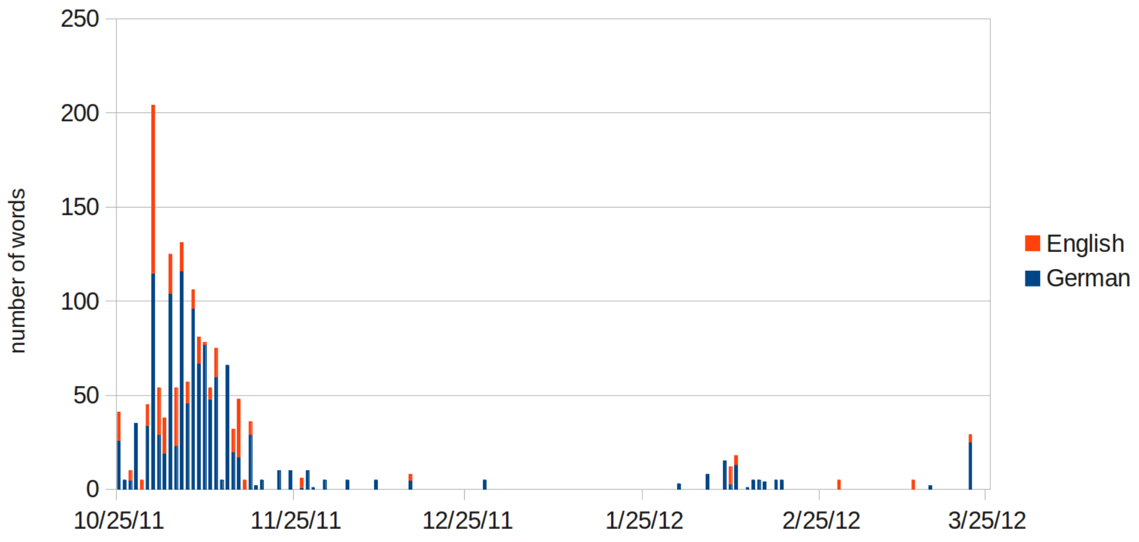


Figure 7.2: Number of German and English words per day by volunteers

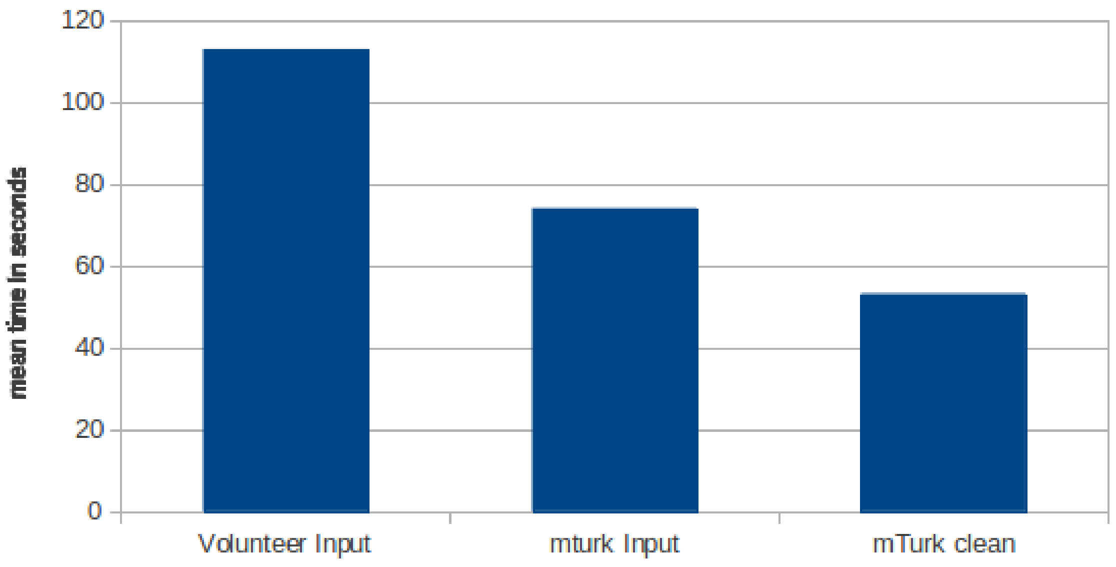


Figure 7.3: Comparison of amount of time needed per word

of time Turkers expended on a word was significantly smaller than the 113 seconds that volunteers needed. Even more so, because with the volunteer experiment, only the time to generate the word itself was recorded. The mTurk website did not only record the Keynounce process but the time it took to finish the HIT. The average time spent on the Keynounce website by the Turkers was 53 seconds, not even half as long as the volunteers.

Since neither the Turkers, nor the volunteers as a whole had any previous experience with the interface or pronunciation generation, the difference is not due to higher skills on the Turkers part. The main incentive for the Turkers was getting payed. With the micropayment system of mTurk, the main focus seems not to be on generating the best possible answer, but rather to get an adequate answer as fast as possible. This is a feasible approach, when dealing with simple answers like 'yes' or 'no'. But in our case, the first recognizable pronunciation seldom is the best one possible. The incentive of the volunteers is to help the project or its creators and maybe have some fun doing it. Their incentive is not losing value, the longer they experiment with a single pronunciation. And finding the best sounding solution might be rewarding on its own.

This means, that the goal of the experiment and the incentive offered to Turkers exclude one another. Keynounce was created in way, that people could use a trial and error system to find a near perfect solution. But this takes time and effort. By paying minimal amounts of money for the tasks does get people involved in Keynounce. But the majority of Turkers has no interest to spend a lot of time on it. At least not for the amounts of money that we offered.

Although there are exceptions. One Turker did all HITs that were offered and he experimented a lot with the interface. The different pronunciations that our system logged from his account showed, that he even tried to form whole sentences, just to see how it worked. He commented most of his actions in the feedback panel.

This did not help with our experiments, but showed the allure of the interface to interested people. Some of his comments were:

> I like when a word is ok, but then gets better by adding another symbol to modify the vowel sound.

> I found the symbol that makes the 'th' sound faster than a couple months ago. I had forgotten that there was a symbol for it, but I remembered what a hard time I had trying to make the sound. It is that thing in between the 'd' and the 'f' (I think. It looks like a backwards 6 with a little mark on it.)

In summary the mTurk experiment yielded a lot of information, but a couple of points made this line of experiments unsuitable for our purposes. These points are:

- High amount of spam (55% unusable input)
- Fast but sloppy (1902 pronunciations within 5 days)
- No incentive to make an answer "the best answer"

These three points, but the last one in particular, made the mTurk experiments unsuitable. It would be possible to devise anti spam procedures to clean the results. But what we need from our test subjects is the willingness to search for the best solution, even if they have already found an understandable solution. This is at cross purposes with the needs of the Turkers, since they try to finish as many tasks as possible, in as little time as possible. This is the reason why we abandoned the mTurk approach and concentrated on voluntary participants to our experiments, even though that made it slower.

In the next section, we present the results of the volunteers and discuss them in detail. Where possible, we compare the volunteer data with what could be salvaged from the mTurk experiment.

7.2 Voluntary Keynounce

As mentioned in the chapter before, we collected pronunciations for 1.595 words in from 10/25/2011 to 04/17/2012. 89% of these pronunciation where collected within the first month.

Before we go into more detailed explanations of the collected data, we need to establish some terms:

Session In the voluntary Keynounce experiment a session is a sequence starting with the welcome screen, where a language can be chosen by the user. As background activity the interface records the chosen nickname of the user and creates a session ID. It then progresses through a maximum of five words of the chosen language, for which the user should build pronunciation strings.

The words are chosen at random, but words with fewer pronunciations get picked more often. Every time the user listens to a string or commits a string as a good word, the sequence, as well as the time and the session ID are stored in the database. Either after five words, or when the user uses the "end" button, a final screen is displayed, which shows the results as described in Section 5.3. This screen finalizes a session by terminating the ID.

Phoneme Error Rate The Phoneme Error Rate (PER) is the percentage of wrong phonemes in a given phoneme string. It is calculated by comparing the hypothesis strings with the reference string from either the English or German reference dictionary. This is done with SClite [22], which is part of the NIST Scoring Toolkit [23]. SClite aligns the hypothesized strings with the reference string and analyzes the differences.

It is normally used for whole texts, which are then compared on a word level. For our purposes we have split the strings into single phonemes by adding spaces after each phoneme. Thus SClite does not compare word errors but phoneme errors. So the PER is the percentage of wrong phonemes in the hypothesis string. A peculiarity of the PER is its ability to become larger than 100%. That is the case, when the hypothesis has more phonemes than the reference.

As we have expected from a voluntary userbase, we initially had quite a large number of volunteers who tried the game. This is depicted in Figures 7.4 and 7.5 as the orange lineplot. 122 different people for the German language and 45 for English took a look at it. The number of participants then plunged to 30 and 17, respectively, for the second session. Here our expected user groups (Section 4.2) of *Onlookers* and *Gamers* separate. While the *Onlookers* only tried one session, the *Gamers* tried a second or even more sessions. The numbers keep dwindling to just 4-8 German *HiG* and 3-4 English *HiG* who stay with the experiment for more than five sessions.

7.2.1 General results

As can be seen in the blue lineplot of Figure 7.4 of the German language results, the users learn very quickly. After the initial session the mean PER is 45.86%. For those users who look at the results screen and then start a second session, the mean PER drops to 35.33%. Until session 8 the PER has declined to 25.80%. At this point all *Gamers* have dropped out and only the *HiG* are left. The mean PER then becomes more erratic, since there are so few users left.

The green lineplot visualizes the median time it took all users to build a single word in the German language results. It starts out with 125.4 seconds per word which is almost 10.5 minutes to a session of 5 words. In the second session there already is a reduction of 30 seconds per word (94.4 s/word, 7.9 min/session). Those users who stay with the game after two sessions apparently become a lot more familiar with the keyboard and what sounds the symbols stand for. The time needed to produce the lowest PER of 25.80% is 41.8 seconds per word or just a little under 3.5 min per session. It also gets more erratic with only a few *HiGs* playing on, but in general keeps going down.

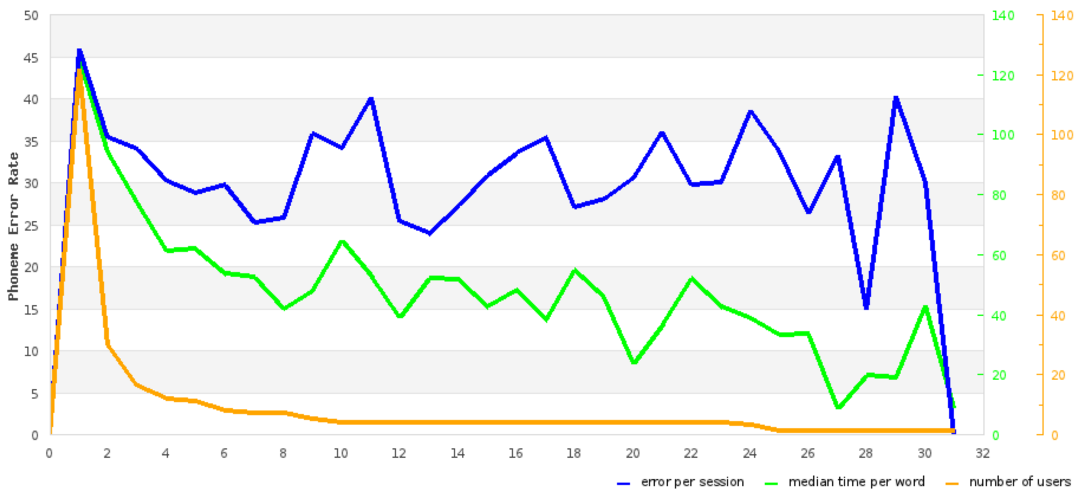


Figure 7.4: PER, time per Word and number of users per session with German words

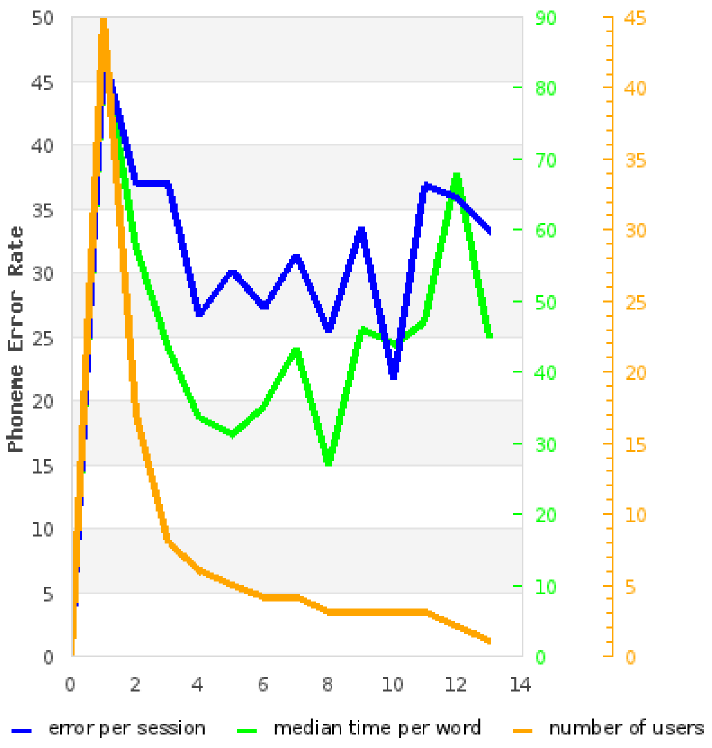


Figure 7.5: PERs per session with English words

Virtually the same can be seen in Figure 7.5 for the English users (blue lineplot). The mean PER starts out at 46.33% in the first session and 35.34% in the second session. Then it drops to about 27% in session 4 and 6 and becomes increasingly erratic afterwards.

The time per word measurements of the English users also closely resemble those of the German users. They start out with 84.4 seconds per word, which is 7 minutes per session. The median time goes down to 57.5 seconds per word or 4.8 minutes per 5 words in the second session. The lowest PER of 27% in sessions 4 and 6 are archived in a little under 3 minutes or 33.4 seconds per word. In session 5 the users finished 5 words in a median time of 2.6 minutes or 31 seconds per word, but the PER went a bit. With only the *HiGs*, it also turns erratic and even starts going up with just 4 players remaining.

We have used the median time in Figures 7.4 and 7.5 since it protects the plot from outliers. Some users probably left the game open for a while, and finished the session later. One user needed 3 hours to finish a session, another a little under 2 hours. These outliers do not have an impact on the median which results in a more reliable plot.

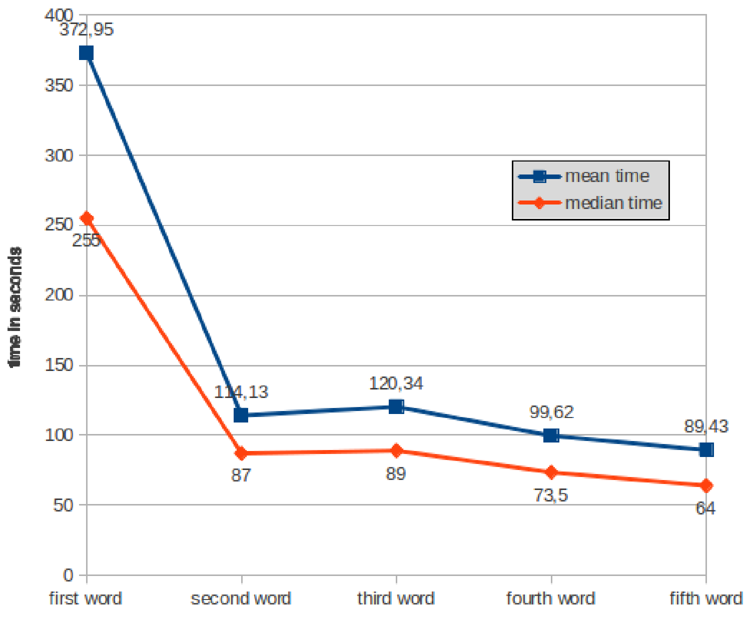


Figure 7.6: Arithmetic mean and median time needed for single words in first session

The reason for the initial rise and subsequent decline of the PER probably is a growing familiarity with the keyboard. Initially most users do not know what each symbol stands for. After some trial and error they start to understand what they are supposed to do and how to do that with the given keys. Better understanding leads to faster results. This can be seen in the decline of time used per session in general. We can even show it on a per word basis for the first session illustrated in Figure 7.6. The Figure shows the arithmetic mean time (blue lineplot) as well as the median time (red lineplot) needed for each word, in every user's first session. On average 373 seconds (over 6 minutes) are needed to find the first word. After that 114 and 120 seconds for the next two words. The last two words are then found in 99 and 89 seconds. The median is markedly lower. The median user needs 255 seconds or 4.25 minutes to complete the first word. The following two words are found in about 89 seconds. For the last two the median user only needs 73.5 and 64 seconds.

As before we have used the median and to compare the arithmetic mean of the timespan it took the users to complete pronunciations. The median is again lower than the arithmetic mean, since it ignores single datapoints where users needed unusually long. As before we assume that these outliers are created, when users get distracted and leave the game open in their browser for a while.

Unfortunately the data for the first session does not include all participants of the study. Some participants had their computers configured to delete cookies, which made it impossible to track, which words where generated in which session. The data of these participants has been excluded for this Figure.

The median time needed for the last words in the first session is below the median for the second sessions, as mentioned above. This can be explained as follows. After the first session, the users can evaluate their own solutions and compare them against the reference pronunciations and the three most frequent solutions of all other users for this word. Those users that continue with a second session afterwards are, on the one hand, more familiar with the keyboard. On the other hand they might have seen on the result screen, that better solutions are possible, which might lead to a little more time spent on the fine tuning. Another influence might be, that users are more inclined to spend time on each word, if he or she did not start the session by spending 6 minutes on the first word.

7.2.2 Results after postprocessing

In this section, we discuss more specific information about the benefits of Keynounce. So far we have shown, that there is improvement in the performance of users in general and in a specific group of users in particular. We only evaluate the German results, since we did not gather enough English data, to present valid information. We are now showing what can be gained from the crowd data, by using a simple tool.

nBest-lattice nBest-lattice is part of the SRI Language Modeling (SRILM) toolkit [24]. It constructs a word lattice from all hypotheses and extracts the path with the lowest expected word error. As with SCLite we use our phonemes as words, by adding spaces after each phoneme.

PER all	PER nBest	users, who processed	# of users
34.7 %	22.7 %	everyone	122

Table 7.1: PER over all words from German users in different word ranges

To begin with, we use SCLite to calculate the PER for all hypotheses that we gathered, matched against their reference pronunciation. The result is, as shown in Table 7.1, that 34.7% of the phonemes differed from their reference. Then we process the hypotheses for each word with nBest-lattice first, to get a better hypothesis. This results in a PER of only 22.7% which is a relative improvement of 34.6%.

An example of nBest-lattice' progress is shown in Table 7.2. The reference and hypotheses are depicted in the internal alphabet of espeak, which is based on Kirshenbaum's representation of IPA in ASCII characters [25].

reference: **d O n @ r s t A: k**			
user hypotheses	PER	nBest hypothesis	PER
d O n @ r s t a g	25 %		
d o: n @ r s t a g	37.5 %		
d O n E r s t A: g	25 %		
d O n @ r s t a g	25 %		
d o: n E: r s t A: g	37.5 %	d O n @ r s t A: g	12.5 %
d o: n n 3 * S t A: g	75 %		
d O n 3 s t A: g	25 %		
d O n @ s t A: k	12.5 %		
d O n 3 r s t A: g	25 %		

Table 7.2: Hypothesis calculated with nBest-lattice from user's hypotheses

It can be seen, that only one of the hypotheses has a PER of 12.5%. Every other hypothesis has 25% PER or more. A simple vote of the most common hypothesis would have chosen "d O n @ r s t a g", since this was the only hypothesis that has appeared twice. The nBest-lattice tool finds a hypothesis, which was not given in that form by the users. The calculated hypothesis has a PER of 12.5%, which is better than 89% of the other user proposed hypotheses. The only mistake, which remains in the calculated hypothesis is at the last phoneme position. Since only one of the user proposed hypotheses has the correct phoneme, the nBest-lattice tool had too little data to find the correct phoneme for that position.

We will call PERs that where calculated using the nBest-lattice tool **nBest PER**.

As outlined in Section 4.2, we expected users to need one or two sessions to get familiar with the game. This has been confirmed in Section 7.2.1, where we have shown, that it takes users a lot more time to complete the first session than any other session. We can further substantiate that by calculating the PER for all users who did finish just one or two sessions.

Table 7.3 shows, the difference between *Gamers* and the *Onlookers*. The fist line of the table gives the PER and the nBest PER for the 92 users who have submitted 5 words or less. This means, everyone is included, who stopped after the first word, up to those that finished the first session. This group archives a PER of 42.7% and an nBest PER of 37.6%. The second line shows the 45 users who have done 4 words or less. They have either abandoned the game, or skipped words. The achieved PER for this group is 54.5% and about the same when calculated with nBest-lattice. This already shows, that those users who do not have enough incentive to finish the first session properly, are in general not contributing worthwhile input.

PER all	PER nBest	Users, who processed	# of users
42.7 %	37.6 %	5 or less words	92
54.5 %	54.6 %	4 or less words	45
34.3 %	29.8 %	6 to 10 words	15
40.6 %	**29.4** %	10 or less words	107

Table 7.3: PER over all words from German users who worked on one or two sessions (5-10 words)

The PER significantly decreases for those users who have done a second session or at least 6 but not more than 10 words. These 15 users produced a PER of 34.3%. The result from nBest-lattice is 4.5% absolute lower. This means that even though there are fewer hypotheses to work with, they have more phonemes in common for nBest-lattice to find.

To conclude this, the last line in Table 7.4 shows the overall PER for users who did any amount of words within the first two session. The PER is at 40.6% and the lowest nBest PER is at 29.4%. But as we have seen in the preceding results, the input from users not finishing the first session is mostly of no interest. In general the results of users that try a second session are much more reliable, but still not as good as the PER over all user input.

PER all	PER nBest	Users, who processed	# of users
31.6%	21.6%	10 or more words	25
31.2%	20.9%	15 or more words	14
31.0%	**20.8%**	20 or more words	12
32.6%	25.4%	100 or more words	4

Table 7.4: PER German global

After finishing the first two sessions the *Gamers* slightly reduce the PER but show a significant decrease in the nBest PER, as can be seen in Table 7.4. Those that did more than two sessions have a PER of 31.6% which is an absolute 2.7% better than those users, who stopped after the second session. But the nBest PER drops to 21.6% which is 8.2% better. Users who did a fourth or fifth session still have a slight decrease in both PER and nBest PER to a minimum of 31% and 20.8%.

The 4 HiGs that did more than 100 words have a slight increase of 1.6% in their PER but a rather larger increase of 4.4% in the nBest PER. This is due to the fact of the significantly reduced number of hypotheses per word, generated by those four users. That the PER does not further drop with training might imply, that just showing the final screen with the results has its limits. Users who have looked often enough at that screen might not be as interested in the results any more. In future work it might be a goal to find some further means to let users improve themselves even after they have gotten into some kind of routine.

7.3 Summary

In this chapter, we have presented and discussed the results of our Keynounce experiments.

First we have discovered, that using Amazon's Mechanical Turk services does not really work with our game. Most mTurk users do not like to spend more time than absolutely necessary on any given task. Also there is the problem of high spam content, which in itself does not pose a problem. The unprofitable data could and in the future probably must be filtered, but coupled with pending payments the answers have to be either reliably good or the time saved to create linguistic data is spent on garbage control.

A second general fact is, that we were able to get a lot of input from voluntary users, but it took more time and effort, than just using mTurk services. On the upside, we did not have to guard against excessive garbage input, since we did not pay for anything.In future experiments Keynounce would have to be posted on a more highly frequented website. This would generate more input without being dependent on having a lot of friends.

More importantly for our work, we proved, that Keynounce works. It can create pronunciations for words which are quite good. Without much postprocessing the results can be enhanced even further. It was also quite clear, that Keynounce was able to train the users to get to the right spelling. This feature might be a possible enhancement in the future.

8. Conclusion

We started the Keynounce project with the goal of proving, that unskilled and anonymous people can help build a high quality pronunciation dictionary at low cost.

Our experiments show, that this is the case. Given the right kind of incentive in providing a gamelike experience, anonymous users will help for free. The experiments uses the human ability to compare synthesized speech with what the user knows to be a right articulation. As the human brain is able to decide relatively accurate, if slight differences are based on the synthesizer or the chosen pronunciation, most users can create good results.

The results of each individual user are not by any means perfect. This is due to the fact that we have no way to decide if the person is linguistically trained in any way, has hearing or speaking disabilities or is simply not willing enough to build the best result he or she could archive. Since our goal also was, to establish a way to create dictionaries for little or unknown languages, in the future we might not even be able to weed out imperfect results apart from obvious garbage, even if we wanted to. But since our experiment makes use of "the crowd", willing users on the Internet, one can expect a large group of users for almost every language. Even if the potential userbase on the Internet for a particular language is not high enough, the Keynounce project is simple enough to carry it on portable devices to speakers of every language.

Using the power of crowdsourcing for our benefit, we were able to calculate pronunciations out of the sum of user inputs, that were as good or even better than every single user input by itself. This is due to the fact that crowd answers tend to agree an a large percentage of the answer, so the difficult parts of the pronunciation are easily identified. If enough user input is available, the contested parts of the pronunciation can be calculated by what a majority agreed upon. If there is enough input, it might even be possible to calculate two or more pronunciation variations due to dialect or regional differences.

Our project also showed, that it might be easier to get people to help, when contacted via a micropayment system like Amayons Mechanical Turk. The disadvantage of having to spend money on the answers is alleviated by the speed with which the results are gathered. But it showed in our experiments that micropayment workers in general are not overly fond of investing a lot of time in fine tuning answers. This was at odds with what we needed users to do. This problem might be worked out by training and chosing the workers better and by investing more money. But the simple approach of just posting the problem and getting the answers was by far more effective without any monetary incentive. Money seemed to put the users on a tight timetable, which was counterproductive to our goals.

In summary our project works. Keynounce is able to invite people to help and offers them enough incentive to produce a reasonable amount of results. The interface is intuitive and simple so that there is next to no work involved in training new users. And finally the quality of the results is high enough to calculate a good quality of pronunciations, provided there is enough input.

9. Future Work

For future works with Keynounce some ideas and different approaches have presented themselves during the course of this work. Although they would be too much to implement in this work, they are definitely worthwhile pursuing in future Keynounce projects.

This work has shown, that a crowd of users without prior linguistic knowledge can build pronunciations for words known to them. The sum of the given pronunciations can be processed without much effort to yield a very accurate pronunciation. In the future these filtering methods should be explored in more detail. It would be useful to find a threshold for the amount of user input that is needed, to build high quality results with n-best lattice.

To make more stable pronunciations and to make Keynounce more versatile, some form of culling of bad words should be found. This could easily be archived by giving users the option of tagging a word as 'problematic'. Words with high amounts of these tags could then be filtered out. Adding this feature to Keynounce would make it possible to feed words into Keynounce unsupervised. It would then be possible to either crawl the Web for words or have other programs provide them for Keynounce. Once users have worked on these words and provided a sufficient amount of pronunciations, the words and the resulting pronunciation could be given back to the provider of the input words.

It must also be examined if there is an advantage of providing help for the users. For example: Keynounce could provide information about what other users propose as a solution for the current word. Especially parts of the pronunciation that a large amount of other users have agreed upon could be provided. Users could then accept this and use their time to find the best solution for the disputed parts of the pronunciation.

As a further incentive to play, Keynounce could provide users with a sandbox area. Some users already ignored the input word that they were supposed to pronounce and played around with Keynounce's possibilities. They even build pronunciations for whole sentences. To keep this kind of exploration out of the main pronunciation data, a sandbox area would work great. Maybe a function to build custom MP3 files from the user generated input would appeal to the users. To still gather information from this random play area, Keynounce could ask users to translate their pronunciation into plain text.

In Section 7.2.1 we showed, that users need some time to adapt to the Keynounce interface. This process could be shortened and thus the quality of the resulting data improved, by enlarging the tutorial. It must be determined if Keynounce offers enough incentive to play, if discovering the interface is taken out of the experience. Maybe it would be enough to guide users through their first batch of words, by showing them the results of other users after each word and not just after all words have been completed.

So far we have given the users a keyboard with the appropriate set of IPA symbols for the chosen language. To open Keynounce for less known languages, it could be useful to explore the possibility of a selflearning Keyboard. For that simply all IPA symbols could be displayed. By letting the users use every symbol there is, they would generate data on which keys are more likely for the given language. The Keyboard could then dynamically adjust itself to the appropriate keyset for this language by continually shrinking the unused keys and enlarging the frequently used ones.

It would also be interesting to be find out, if pronunciation variations for a word could be provided by a large number of users. Keynounce would have to decide not only on which is the best answer that can be filtered out of the sum of inputs. But it would also have to find the second and third best answers and if they are common enough, they might be a pronunciation variation.

Also with the Rapid Language Adaptation Toolkit (RLAT) [2] there already exists a powerful tool which aims at reducing the human effort in building speech processing systems for new languages and domains. Its innovative tools enable novice and expert users to develop speech processing models, such as acoustic models, pronunciation dictionaries, and languages models, to collect appropriate speech and text data for building these models, and to evaluate the results. A possible way to integrate Keynounce into this existing system would be, to automatically pipe all words, which are gathered through the webcrawling system of RLAT, into Keynounce. The words are thus presented to the community for dictionary creation. Thus an ever evolving dictionary for all possible languages could be build.

Bibliography

[1] "Cognitive Systems Lab." http://csl.anthropomatik.kit.edu//.

[2] "Rapid Language Adaptation Toolkit." http://csl.ira.uka.de/rlat.

[3] T. Schultz, "GlobalPhone: A Multilingual Speech and Text Database Developed at Karlsruhe University," in *Proceedings of the ICSLP*, pp. 345–348, 2002.

[4] T. Schultz and A. Waibel, eds., *Das Projekt GlobalPhone: Multilinguale Spracherkennung*, vol. Computers, Linguistics, and Phonetics between Language and Speech. Proceedings of the 4th Conference on NLP, Fakultät für Informatik Institut für Algorithmen und Kognitive Systeme (IAKS) Institut für Theoretische Informatik (ITI), 1998.

[5] J. Howe, "The Rise of Crowdsourcing," *Wired Magazine*, vol. 14, 06 2006.

[6] E. E. Arolas and F. G.-L. de Guevara, "Towards an integrated crowdsourcing definition," *J. Information Science*, vol. 38, no. 2, pp. 189–200, 2012.

[7] W. Tay, "Linguistic Wonders Series: The International Phonetic Alphabet."

[8] D. V. Compernolle, "Recognizing speech of goats, wolves, sheep and ... non-natives," *Speech Communication*, vol. 35, no. 1-2, pp. 71–79, 2001.

[9] I. P. Association, *Handbook of the International Phonetic Association.* Cambridge: Cambridge University Press, 1999.

[10] "Wikipedia, the free encyclopedia." http://www.wikipedia.org.

[11] P. Spotts, "Crowdsourcing science: how gamers are changing scientific discovery." http://www.csmonitor.com/Science/2011/1005/Crowdsourcing-science-how-gamers-are-changing-scientific-discovery, 2011.

[12] Z. Popovic, "Foldit." http://www.centerforgamescience.org/site/games/foldit.

[13] Z. Popovic, "Center of game science." http://www.centerforgamescience.org/site/.

[14] L. von Ahn, "Games with a purpose," *IEEE Computer*, vol. 39, pp. 92–94, 2006.

[15] L. von Ahn, R. Liu, and M. Blum, "Peekaboom: A game for locating objects in images," in *In ACM CHI*, pp. 55–64, ACM Press, 2006.

[16] M. Davel and E. Barnard, "The efficient generation of pronunciation dictionaries: Human factors during bootstrapping," in *8th International Conference on Spoken Language Processing*, 2004.

[17] J. Kominek and A. W. Black, "Learning pronunciation dictionaries: Language complexity and word selection strategies," in *Proceedings of the Human Language Technology Conference of the NAACL*, pp. 232–239, 2006.

[18] "What is a troll." http://kb.iu.edu/data/afhc.html.

[19] Oracle, "Java website." http://www.java.com/de/.

[20] Adobe, "Adobe Flash." http://www.adobe.com/de/products/flash.html, January 2011.

[21] jonsd, "espeak text to speech synthesizer." http://espeak.sourceforge.net/, January 2012.

[22] "SClite." http://www1.icsi.berkeley.edu/Speech/docs/sctk-1.2/sclite.htm.

[23] "NIST Scoring Toolkit." http://www1.icsi.berkeley.edu/Speech/docs/sctk-1.2/sctk.htm.

[24] A. Stolcke, "SRILM – An Extensible Language Modeling Toolkit," in *Proc. Int. Conf. Spoken Language Processing (ICSLP 2002)*, 2002.

[25] E. Kirshenbaum, "Kirshenbaum ASCII IPA." http://www.kirshenbaum.net/IPA/.

[26] Adobe, "Adobe Actionscript 3." http://www.adobe.com/devnet/actionscript.html, June 2011.

[27] Canonical, "Ubuntu Server 10.04 Long Term Support." http://www.ubuntu.com/, January 2011.

[28] H. A. Engelbrecht and T. Schultz, "Rapid Development of an Afrikaans-English Speech-to-Speech Translator," in *Proceedings of International Workshop of Spoken Language Translation*, 2005.

[29] P. Ipeirotis, "Mechanical Turk: The Demographics." http://www.behind-the-enemy-lines.com/2008/03/mechanical-turk-demographics.html, March 2008.

[30] Oracle, "PHP." http://www.php.net/.

[31] T. Schultz, A. W. Black, S. Badaskar, M. Hornyak, and J. Kominek, "Spice: Web-based tools for rapid language adaptation in speech processing systems," in *Interspeech*, 2007.

[32] S. Young, "Large vocabulary continuous speech recognition a review," *IEEE Signal Processing Magazine*, vol. 13, no. 5, pp. 1–4, 1996.

[33] "History of IPA." http://en.wikipedia.org/wiki/History_of_the_IPA.

[34] "Php 5." http://www.php.net/.

[35] "Micropayment." http://en.wikipedia.org/wiki/Micropayment, February 2012.

[36] "Amazon Mechanical Turk." http://aws.amazon.com/mturk/, December 2010.

Index