Building a Multilingual Speech Assistant

A Literature-Based Personal Digital Assistant Designed for Processing Multiple Languages and Code-Switching

Daniel Pietschke

A thesis presented for the degree of Bachelor of Science

Department of Cognitive Science
University Osnabrück
First Supervisor: Dr. Elia Bruni
Second Supervisor: Xenia Ohmer, M. Sc.
Hand in Date: 31st of August 2021

Abstract

Personal Digital Assistants (PDA) and Speech Assistants are becoming a bigger part of everyday life. Nowadays every smartphone is outfitted with at least one assistant of some sort. Additionally, migration and globalization are leading to an increasing number of multilingual people worldwide, and knowing more than one language is considered normal in many parts of the world. However, Personal Digital Assistants are not designed to work with multilingual input most of the time. They either support only one language from the start, which has to be determined when setting up the assistant, or they support multiple languages but only if the queries remain monolingual. Modern-day PDAs do not yet possess the property of processing code-switched input.

This thesis aims to design a PDA that is not only able to process multiple languages but also to do it without any pre-specification of languages. Furthermore, the PDA is designed to process code-switched input, which makes the system a true multilingual PDA. In order to do this, a literature review is conducted. The presented papers introduce the architecture of a PDA as well as several models and frameworks which are needed as background information. Two modules are chosen which have to be designed in order to make the PDA multilingual. The first is the speech recognition module which is based on four papers presenting different approaches to multilingual speech recognition. The second is a pre-processing module which is to be inserted before the standard Input Parsing module in order to simplify further processing of the input. This module has three steps: Named Entity Recognition, Language Detection, and Translation. Five papers serve as the basis for this module.

The resulting system combines different metrics which have proven successful in the presented papers, such as language margins, a strategy module, and the combination of language-dependent and language-independent language models and acoustic models for the speech recognition module. Furthermore, the speech recognition module is based on DNN frameworks that work in parallel and score the input on a query and token level. The strategy module makes the final decision of which audio transcription is the best one. The pre-processing module combines several Hidden Markov Models for Named Entity Recognition and uses a multilingual vector space in order to identify named entities. In order to detect language switches, several monolingual Hidden Markov Models are used. The Translation step is done by using the Mutual Information metric and the training corpus in order to identify suitable translations.

Since the combined systems performed very well in their respective papers, combining them is theorized to lead to good performance as well. Nevertheless, some limitations are identified. These limitations are *a)* a possible language bias as long as languages are not sufficiently represented in the training corpus, *b)* performance drops due to parts of the system not performing well and decreasing overall performance in the process, and *c)* the difficulty of collecting enough training data to enable the system to be used for many languages. The limitations are able to be solved by future research, however, and if the system is implemented and works well, it can provide multilingual services to people around the world, on their mobile

devices as well as tourist information counters in order to create a more globally connected world.

Contents

1 Introduction **4**

2 Speech Assistants **5**
- 2.1 Input Recognition 6
- 2.2 Input Parsing 6
- 2.3 State Update 7
- 2.4 Policy 8

3 Models **9**
- 3.1 General Framework Models 9
- 3.2 Acoustic Models 11
- 3.3 Multi-Pass and One-Pass 12
- 3.4 Input Recognition Models 12
- 3.5 Pre-Processing Models 22
  - 3.5.1 Named Entity Recognition 22
  - 3.5.2 Language Detection 26
  - 3.5.3 Translation 31

4 Discussion **34**

5 Limitations of the Proposed System **43**

6 Conclusion **44**

1 Introduction

Technology is becoming more prevalent and involved in our lives. Especially smartphone use has increased over the last few years (O'Dea 2021) and the use of Personal Digital Assistants (PDA) as well (Vailshery 2021; Nair, Chintagunta, and Dubé 2004). Nowadays every smartphone is outfitted with at least one PDA of any kind, more people gain access to using this technology in their everyday lives. Additionally, due to globalization and migration, cultures and languages become intertwined and multilingualism is ever increasing (Bacon-Shone and Bolton 1998; Donakey 2007) which leads to more diverse communities around the world. This means that it will become increasingly common to speak more than one language which increases the demand for available applications to adapt to the multilingual users and become multilingual themselves. However, most PDAs, especially the ones used in smartphones and home applications, do not support more than one language at a time. Although supporting more than one language is possible, code-switching remains a problem. Code-switching describes a switch between at least two languages which can happen several times over the course of a conversation and even within a sentence (Riehl 2019). Very often code-switching is motivated by wanting to express certain feelings for a specific sentence or to adapt to the current communicative situation. However, code-switching can also happen when encountering specific "trigger words" (Riehl 2019, p. 2).

In this thesis, I aim to devise a theoretical framework for a multilingual PDA which is able to work with multiple languages and does not require a pre-specification for the languages it will have to process. The PDA is intended to also be able to handle code-switched input, making it universally applicable to anyone, given enough languages are supported. To do this, I will review literature, use the most promising ideas and combine them into an architecture that solves the challenges outlined above. Before starting the review of the models, I will introduce the structure of PDAs in Section 2 and explain each module and how it works. I will also explain the differences between the forms of PDAs and introduce the user profiles which are used in PDAs and will have a significant role in my own approach. Furthermore, I will select the modules I intend to focus on in this thesis and explain my decision. After that, I will introduce some general framework models which are often used in speech assistants in Section 3.1 and compare some of their advantages and disadvantages. Specific speech recognition architectures, which will be important for the final model, will be introduced and explained in Sections 3.2 and 3.3. After giving the necessary background information, I will review several approaches to multilingual speech recognition in Section 3.4 and introduce my own idea for a pre-processing module in Section 3.5. In that section, I will explain my focus on a pre-processing module and consider different approaches to the various necessary pre-processing steps. After each paper, I will discuss the most central implications of the results and which parts could be useful for my own approach. In Section 4, I will compare the ideas and approaches I have presented and create my own detailed PDA architecture. Furthermore, I will discuss some limitations of my idea which

I have identified in Section 5 before concluding and giving an outlook on further research in Section 6.

2 Speech Assistants

In order to build a Multilingual Speech Assistant or Personal Digital Assistant (PDA), it is necessary to understand is how a PDA works and how it is constructed. Sarikaya 2017 provides great insight into that topic. In his article, he divides PDAs into proactive and reactive PDAs. Proactive PDA means that the system will automatically manage calendar, messages, notes, and more, and suggest complementary ideas (for example after telling the PDA to book a seat at a movie theater it will ask you if it is supposed to book a taxi to take you there as well). Reactive PDAs, on the other hand, will only become active when they are activated by user input. Although these systems are distinct and are mostly constructed apart from each other, they can and often do have extensions which are more common in the respective other system (Sarikaya 2017). Since the pure reactive architecture is easier to work with and so far there is no true proactive PDA (Sarikaya 2017, p. 70), I will focus on the reactive system architecture in this thesis.

In general, a PDA is a very complex system that uses a lot of tools in order to offer the best possible user experience. Additionally to PDA-specific technologies, such as Machine Learning, Speech Recognition, Language Understanding (LU), Language Generation, and more (Sarikaya 2017, p.68), PDAs also use the tools and information provided by the device like gyro sensors, GPS, and motion sensors (Sarikaya 2017, p.69). This thesis will focus on the PDA-specific technologies since these are the ones directly affected and used by the PDA in a multilingual context.

Figure 1 shows the general framework of a reactive PDA. I will explain the workings of such a system by going through each step illustrated in the figure.

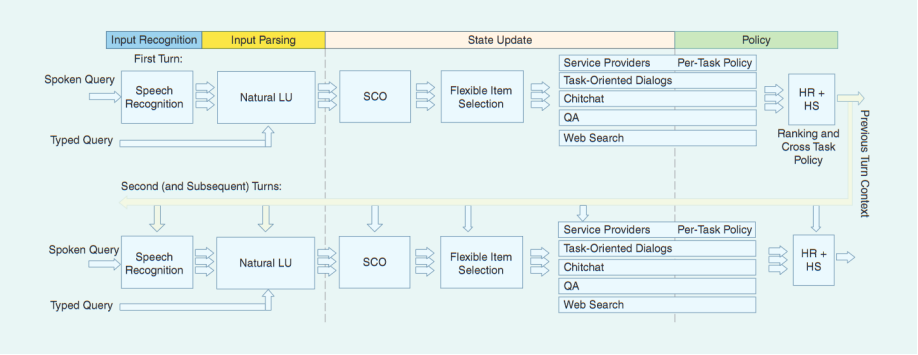

Figure 1: The architecture of a reactive PDA as given in Sarikaya 2017.

2.1 Input Recognition

When making a vocal query for a PDA, the first system part to be activated is the one labeled as "Input Recognition" in Figure 1. It contains the speech recognition system of the PDA. This part of the PDA, however, is only activated if a query is spoken. Typed queries go directly to "Input Parsing" (see Section 2.2).

Speech recognition starts whenever the system microphone picks up the correct voice activation (e.g. "Hey Google" or "Hi Bixby"). The input is then recognized by its acoustic signals, transcribed, and sent to "Input Parsing".

The detection of input and transcribing is done with the help of a phonetic lexicon or dictionary and the corresponding language models and learning algorithms (Knill et al. 2014, Bhuvanagirir and Kopparapu 2012, H. Lin et al. 2012, Gonzalez-Dominguez et al. 2014). The dictionary is a collection of every word known to the PDA with phonetic representations in order to be able to model voice input. It has to represent the input language almost perfectly. Furthermore, the language model must be able to recognize and label the phonetic signals correctly.

When building a PDA that is able to react to multilingual and code-switched input, the Speech Recognition module is the first part to take into account. The first decisions regarding the identification of languages for spoken queries are made here, therefore this system has to be able to handle complex input. The details of the multilingual Input Recognition module will be further discussed in Section 3.4.

2.2 Input Parsing

Input Parsing describes the part of a PDA which is responsible for managing the incoming queries. It works with either directly written queries or with transcribed vocal ones, as seen in Figure 1. At the base of Input Parsing is a Natural Language Understanding module (Natural LU in Figure 1) which possesses the required algorithms for converting the incoming queries into usable commands for the PDA.

The Natural LU system has to be able to not only parse the inputs semantically but, in some cases, also correct mistakes made in the Input Recognition step (Sarikaya 2017, p.73). To allow for high flexibility, the Natural LU system can use rule-based approaches or machine learning approaches (Sarikaya 2017, p.73).

When a query gets sent to Input Parsing, it is first sorted into the respective domain needed to complete the query (Sarikaya 2017, p.73, 74). These domains contain so-called "intents" which are domain-specific commands on a system level (e.g. the domain "food" and intent "get_restaurants" for the query "which restaurants are near me?") (Sarikaya 2017, p. 74). A "slot tagger" identifies important semantic information from the query which is used to complete the task, for example starting an app or getting information from the knowledge base (Sarikaya 2017). Slot information is also used for keeping the context over

queries in order to avoid the challenge of parsing ambiguous input (Sarikaya 2017). A representation of domain and slot is shown in Figure 2.

```
<Domain score="0.8956">reminder</Domain>
    <Intent score="0.7949">create_single_reminder</Intent>
  <Slots>
    <reminder_text ="call my mom" />
    <start_time <RawValue="9 am">
          <PropertyGroup Name="timex3">
            <Property Name="value" Value="am" />
            <Property Name="comment" Value="am" />
          </PropertyGroup>
    </start_time>
  </Slots>
```

Figure 2: Visualization of intents and slots - figure taken from Sarikaya 2017.

The parsed input is combined with knowledge from a user profile. User profiles are created by the system in order to obtain and keep important user information and to provide a personalized user experience (Sarikaya 2017). The user profile uses information which is given by the user such as name and home address. However, it can in principle also collect information about habits the user expresses, such as often used queries, domains, and also accepted query results, for example restaurants, contacts, and more. For my proposed PDA, the user profile takes on a significant role which will be discussed in Section 4.

2.3 State Update

Whenever a query is made to the PDA, the system tracks the state of the query using the slots mentioned above (Sarikaya 2017, p.74). The slots are important for handling multi-turn queries since very often the context of the first query is important for understanding a possible follow-up query. Sarikaya 2017 has a great example:

> "How is the weather in New York?"
> "What about the weekend?"
> Sarikaya 2017, p.74

Both of these queries have the same context: "weather". But this is not obvious from the second query alone. Therefore, a system needs a way to track the context of a query over the course of several inputs. This is done with the core part of the State Update module: The Slot Carryover (SCO) or Partial Context Manager.

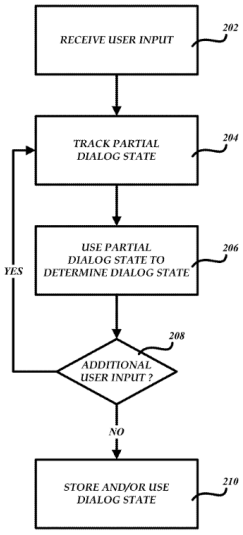

Figure 3: The SCO, picture taken from Boies et al. 2016.

Figure 3 illustrates how the SCO works. It takes the user input, in this case the parsed query and its slots, and uses this information to create a "partial dialog state" (Boies et al. 2016). This state can be updated with every turn and new incoming information, but most importantly it can be transferred between queries so that follow-up queries are not ambiguous to the system. The SCO system accomplishes this by using machine learning algorithms and rule-based decisions and is able to flexibly select the important context for each turn (Sarikaya 2017).

Based on the partial dialog state, a number of items can be selected to fulfill the query. The "Flexible Item Selection" is responsible for presenting the possible selected items to the user who is then able to select the appropriate one (Sarikaya 2017, p. 76). For example, if the system recognizes the domain "restaurant" and the intent "get_restaurant", then the system can ask for confirmation to look for restaurants that are open or nearby. Based on this information, the PDA then decides which task is appropriate for the query. In the mentioned example, the PDA is supposed to search for restaurants nearby. This means that the only task to be performed in this instance is "Web Search". If the follow-up query after presenting the results of the web search was "book a seat at restaurant X", the task would flexibly change to "Service Providers" and execute the required action based on specific policies which will be addressed in Section 2.4. As a result of the State Update module and the SCO, the PDA is able to first look for available restaurants and after a follow-up query book a seat at the user-selected restaurant from the list of presented places.

2.4 Policy

When a PDA is handling a query and the processing and partial dialog state creation is done, the system starts the required routines for the selected task. The results from all possible tasks are then collected and hypotheses are created. These hypotheses consist of possible answers the PDA can give, which are based on the selected task, the assumed dialog state and the selected "dialog acts" (Sarikaya 2017, p.76). These dialog acts are dialog options that will be presented with the requested information and represent different results or follow-up actions such as: show results, disambiguation, prompt for missing value, prompt for no results found, and more (Sarikaya 2017). The hypotheses are ranked and scored by the hypothesis ranking (HR) module, based on their slots, dialog acts, and partial dialog state (Sarikaya 2017). The most likely hypothesis is selected, not only by score but also depending on contextual information, such as the task, rank, and "business logic" (Sarikaya 2017, p. 76),

which uses application-specific logic rules to make its decisions. The selected hypothesis contains the dialog act that is being presented as the result of the query. For any additional query or dialog turns, the slots and partial dialog state are retained and updated to be used for contextual information, as seen in Figure 1.

In the next chapter, I will introduce Gaussian Mixture Models, Hidden Markov Models, and Deep Neural Networks as well as central concepts for speech recognition systems, such as acoustic models and multi- and one-pass approaches. Furthermore, I will present several models for Input Recognition and the pre-processing module.

3 Models

In the previous section, I presented the workings of a PDA. I highlighted the steps a query takes and the modules that make a PDA work. Now I will focus on possible models that can be used in creating a multilingual PDA. As these models mostly concern the recognition and processing of language input, the modules I will focus on are Input Recognition and Input Parsing. The other parts of a PDA are not affected by the multilingual nature of the system, therefore I will not address them here.

First, I will introduce and explain the differences between Gaussian Mixture Models, Hidden Markov Models, and Deep Neural Networks as well as introduce acoustic models and multi- and one-pass approaches. After that, I will present several models for each of the two modules.

3.1 General Framework Models

Two of the most widely used general framework models in speech recognition and language identification are Gaussian Mixture Models (GMMs), which are also very often combined with Hidden Markov Models (HMMs), and Deep Neural Networks (DNNs).

A Gaussian Mixture Model is a "parametric probability density function" (Reynolds 2009, p. 1) which uses a combination of several Gaussians to model "continuous measurements" (Reynolds 2009, p. 1). The density function of each Gaussian in principle combines a number of single Gaussians with different parameters in order to achieve a probability distribution for linear data. This distribution is used to make predictions about possible inputs, which is the perceived learning outcome. By using the distribution, GMMs are even able to clear noise from recorded input (W. Liu et al. 2017, p. 16). Gaussian Mixture Models are often used in speaker recognition tasks due to their ability to smoothly approximate "arbitrarily shaped" input (Reynolds 2009, p. 2).

Hidden Markov Models are a stochastical sequence labeling algorithm. They consist of a finite number of states which have a probability for a certain output. After a state has generated an output, a state transition takes place, where

according to certain probabilities the active state is switched or not switched for the next step of labeling (Rabiner and Juang 1986).

In a combination GMM-HMM model, the GMM part will create a distribution which can be used for prediction and classification due to the HMM's hidden states representing the stochastic modeling of the utterance (W. Liu et al. 2017, p.15, 16). However, this type of framework suffers from some drawbacks as well. Due to their nature, GMMs are unable to model non-linear data, which decreases the overall ability to deal with nonlinear input for the entire GMM-HMM significantly (Reynolds 2009, p. 3; W. Liu et al. 2017, p. 16). Moreover, when dealing with very broadly varying inputs such as different languages and dialects, HMMs need to have sufficient representations of the variances in their training data in order to perform adequately (W. Liu et al. 2017, p.16). Apart from this, a big advantage of using HMMs is that once the necessary training data is acquired, training one is very easy (W. Liu et al. 2017, p.16). Additionally, they perform very quickly (as seen in Saito and Nagata 2003) and due to their sequence labeling nature (Rijhwani et al. 2017) they are a very good choice for working with languages, since language is essentially a sequence of words.

Artificial Neural Networks (ANN) are a different approach to modeling data. ANNs are designed to mimic the workings of natural neural networks (W. Liu et al. 2017, p. 2), which means that a single neuron needs a specific situation in which it is activated and the combination of different neurons leads to different outcomes. In a Machine Learning situation, the activation situation for a single neuron is a value threshold that has to be surpassed by the incoming values either provided by the input or by other activated neurons. An ANN always has an input layer which receives the input, a hidden layer which consists of a number of neurons, and an output layer which generates the output classification of the ANN. The hidden layer is called "hidden" because the values are unlabeled and their output is not interpretable by humans. This is what the output layer is for. Depending on the neurons from the hidden layer, the output layer generates a "decision" of the ANN which is understandable for humans. Deep Neural Networks are variations of ANNs that have more than one hidden layer in order to model more complex input (Bengio 2009, p. 4). DNNs are additionally able to work with large amounts and small amounts of data without overfitting the training data (W. Liu et al. 2017, p. 2; Gonzalez-Dominguez et al. 2014, p. 2) which is convenient when handling either enormous representations or underrepresentations of languages. Another positive aspect of DNNs is that they are able to represent data very compactly when compared to GMMs (Gonzalez-Dominguez et al. 2014, p. 2). This is very helpful for implementing a complex speech recognition system in a limited storage space of a phone, for example. As explained before, GMMs try to predict input and model their data according to these predictions. DNNs, on the other hand, mostly approach data modeling in a discriminatory way (Gonzalez-Dominguez et al. 2014, p. 2) which is helpful for neutral modeling of raw input. Furthermore, DNNs are able to model non-linear input as opposed to GMMs (W. Liu et al. 2017, p.16). The advantages of DNNs come at a price, however, since in order to build capable complex DNNs, large amounts of computational power are needed, which is why the field of

DNNs emerged not too long ago when stronger systems became available (W. Liu et al. 2017, p. 16).

3.2 Acoustic Models

At the base of every speech recognition framework is the acoustic model (AM). The acoustic model consists of a linguistic phone dictionary containing all known phones, a speech corpus or pronunciation lexicon containing the representation of each phone from the dictionary (Bhuvanagirir and Kopparapu 2012; kmaclean 2010), and a statistical representation of each phoneme (kmaclean 2010). Using this information, sound waves can be transcribed into the corresponding phonemes when detected and the system is able to detect the borders of words as well. In a PDA, the acoustic model is one of the first tools used to analyze the query and understand it. The transcribed input is then further analyzed by language models.

Acoustic models can either be language-dependent or language-independent. When dealing with multiple languages, language dependence or independence determines the way the input is analyzed and the language that is recognized from the input.

In a language-dependent acoustic model (LDAM), the input is first parsed through the language phone recognizers of which there is one for every language the AM supports (Corredor-Ardoy et al. 1997). The phone recognizers assign a score depending on the number of familiar phones found. Then, the input is parsed through language-dependent phone bigram models (Corredor-Ardoy et al. 1997). Again, there are as many bigram Models as there are supported languages (Corredor-Ardoy et al. 1997). These models also assign a score which indicates the likelihood that the input phoneme bigrams belong to the respective language. In the end, the two scores from the phone recognizers and bigram Models are summed up and represent the scores for each possible language (Corredor-Ardoy et al. 1997). This way, the single phones, as well as possible combinations, are taken into account for each language. This means that a language score will be low for languages which have a low recognizer score as well as a low bigram Model score. In turn, the language score will be high for a high recognizer and bigram Model score and medium for either a high recognizer or bigram Model score.

A language-independent acoustic model (LIAM), in contrast, only parses the input through a single phone recognizer which has information about every phone of the supported languages (Corredor-Ardoy et al. 1997). The phone recognizer assigns scores that are language-dependent, so a phone can have a higher score for a specific language depending on the likelihood of this phone belonging to the language. After that, the input is parsed through the language-dependent phone bigram models, just as in the language-dependent approach (Corredor-Ardoy et al. 1997). The scores of these two parts are then combined and the language with the highest score is selected (Corredor-Ardoy et al. 1997).

3.3 Multi-Pass and One-Pass

In order to analyze an incoming query, a PDA has to be able to recognize the spoken sound waves from the user. There are essentially two ways of analyzing multilingual input: Multi-pass and one-pass (Bhuvanagirir and Kopparapu 2012).

In a multi-pass system, the input is analyzed to identify the exact moments where the language in the query switches (Bhuvanagirir and Kopparapu 2012). After that, the input gets segmented according to the languages used and can be parsed by their respective language and acoustic models. The acoustic models determine the exact wording of the utterance and the language models determine the semantics of their language segment. The output of these models is then unified to retrieve the semantic meaning of the entire input.

While the multi-pass system works with a divide-and-conquer approach, the performance quality depends on the individual performance of many different systems. The one-pass approach avoids this by being built in a way that the language and acoustic models are already unified and there is only one system needed to recognize the sound waves and learn the semantic meaning of the utterance (Bhuvanagirir and Kopparapu 2012). However, the one-pass approach is often used for only two languages and needs mixed training data which is often hard to come by (Bhuvanagirir and Kopparapu 2012, p.3).

3.4 Input Recognition Models

In this subsection, I will introduce four approaches to input recognition for multilingual inputs. I will describe the general framework used in the experiments as well as the results and implications for this thesis.

The first model was reported by Bhuvanagirir and Kopparapu 2012. They built a one-pass system which is trained to work with English and Hindi input. For their approach, they used a specifically created pronunciation lexicon consisting of two-thirds Hindi and one-third English and proper names in order to mimic the distribution of languages in India (Bhuvanagirir and Kopparapu 2012, p. 3). By using the mixed pronunciation dictionary they could avoid building a special AM for the one-pass system and instead used the SphinxASR (Automatic Speech Recognition) framework with either an AM for English or an AM for Hindi phonemes. For a language model, they used the "well-known n-gram LM" (language model) (Bhuvanagirir and Kopparapu 2012, p. 3).

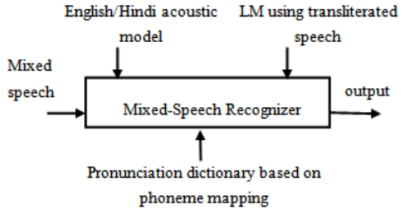

Figure 4: The mixed language recognition approach as reported by Bhuvanagirir and Kopparapu 2012.

Figure 4 shows the idea of the proposed model. Several experiments were performed with different constructions of the pronunciation dictionary where every group of words (English, Hindi, proper nouns) was either denoted by using the CMU language toolkit (available here: http://www.speech.cs.cmu.edu/cgi-bin/cmudict#about, last accessed

19.05.2021), which transliterates Hindi words to English and represents them by the phoneme representation of the English version, or by using Approximate Phoneme Mapping (APM) which transliterates Hindi words to English, segments them accordingly to the Hindi phones and then replaces the Hindi phones with one or more English phones to obtain the representation (Bhuvanagirir and Kopparapu 2012, p. 4). In total, ten experiments were performed, nine of them used the English AM, and one used the Hindi AM and the Hindi phone set to construct its dictionary (Bhuvanagirir and Kopparapu 2012, p. 4). The setup as well as the results of the experiments are shown in Figure 5.

Experiment	English words	Hindi words	Proper Nouns
Expt 1	CMU	CMU	CMU
Expt 2	CMU	APM	CMU
Expt 3	CMU	APM	APM
Expt 4	CMU	CMU	APM
Expt 5	APM	APM	APM
Expt 6	APM	APM	CMU
Expt 7	APM	CMU	APM
Expt 8	APM	CMU	CMU
Expt 9	APM+CMU	APM+CMU	APM+CMU

(a)

Experiments	Accuracies			
	English words (100-% correct)	Hindi words (100-% correct)	Proper nouns(100-%correct)	Overall accuracy (WER)
Expt 1	53.31%	53.19%	86.36%	67.93%
Expt 2	47.45%	46.71%	84.09%	57.29%
Expt 3	44.61%	46.29%	60.61%	53.64%
Expt 4	53.08%	53.08%	86.47%	67.93%
Expt 5	49.53%	46.67%	63.30%	56.16%
Expt 6	52.71%	49.22%	86.47%	60.48%
Expt 7	55.33%	53.41%	60.90%	65.16%
Expt 8	60.18%	54.97%	88.72%	70.40%
Expt 9	40.74%	29.25%	64.66%	44.96%
Hindi-ASR	49.50%	40.62%	45.31%	53.39%

(b)

Figure 5: The experimental setup (**a**) and the results (**b**) of the experiments as reported by Bhuvanagirir and Kopparapu 2012.

The results illustrate that performance for English words and proper nouns was generally better if they were represented using CMU, while the Hindi words were best represented by APM. This implies that representing unknown phones as combinations of known ones is a reliable technique and should be kept in mind for other multilingual approaches. The combination of CMU and APM representations yielded the best results for the English and Hindi words as well as overall performance. This shows that combining different features can lead to better results, however, it is important not to overload the system until performance drops again. Interestingly, using the Hindi AM and phone set, which is a superset of the English phone set (Bhuvanagirir and Kopparapu 2012, p. 4), also decreased the word error rate and increased general performance. It follows, that when it is not necessary to replace any phones for recognizing a language, performance increases, although not as much as when more different word features are used. What can be gained from this approach is that when working with numerous languages, creating a special dictionary which contains

the representations of words with entire word and phoneme-based representations can lead to promising results. There is no need for a separate dictionary for each language, which can simplify the model since fewer additional resources are required.

While the first paper demonstrated how a one-pass system can yield good results when given appropriate training data, the second paper highlights some of the shortcomings of one-pass systems. It contains several different approaches for multilingual language recognition. The models are reported by H. Lin et al. 2012 and focus on recognizing utterances from an English, French and Chinese test set. All recognizer models use a GMM-HMM approach.

In the beginning, the baseline monolingual systems are described and their performance is recorded. Performance is at 60.5% accuracy for the English recognizer, 46.2% for the French recognizer, 41.9% for the Chinese recognizer, and 49.5% overall. These results, however, are not only based on single language recognizers but also on the fact that the system does not need to recognize the used language as a first step before analyzing the actual input.

After that, the researchers built a one-pass system for all three languages by merging the phone and training sets and building the pronunciation dictionary by representing each word with one pronunciation for every language it appears in (which is due to overlap of words in different languages) (H. Lin et al. 2012, p. 2). This system performed not only two times slower than the monolingual systems but also only achieved an accuracy of 38% on average (H. Lin et al. 2012, p. 2). These results imply that a one-pass system on its own faces many difficulties when it comes to satisfactory performance in speech recognition since performance speed was low and accuracy was far from being suitable for a good user experience.

The next model presented in the paper uses a multi-pass approach by including a language identifier as a first step in recognition, which directs the identified input to the corresponding recognizer. This language identifier, according to the authors, needs to provide a very low-latency decision process (H. Lin et al. 2012, p. 2) in order to not inhibit processing speed. Low-latency is achieved by combining a GMM and a support vector machine (SVM), a machine learning technique that separates different classes by either a straight line or a hyperplane (Suthaharan 2016). The GMM models each utterance whose features are then represented as a "super vector" (H. Lin et al. 2012, p. 2) and classified by the SVM. This SVM identifier was tested against a maximum entropy model, which outperformed the SVM system by about 20% (H. Lin et al. 2012, p. 2). However, it was concluded that the amount of training data used affected the outcome more than the actual approach itself (H. Lin et al. 2012, p. 2). In this case, the maximum entropy model, which makes selections based on the maximum uncertainty of outcomes, was able to use 40 times more training data than the SVM model (H. Lin et al. 2012, p. 2, Fig. 1). The authors then indicated that they used the best performing language identifier to test their approach. However, it is not clear whether this means that they used the best performing identifier based on their SVM idea or if they used the maximum entropy model with the most training examples. The multi-pass approach using the language identifier and the different monolingual recognizers achieved on average 43.5% accuracy, which is better than the pure one-pass approach. This result could be made even better by including so-called "margins" (H. Lin et al. 2012, p. 3) to account for language bias. These margins are value thresholds

that have to be surpassed in order for the system to consider the probability significant.

The last two approaches presented by H. Lin et al. 2012 use a multi-pass approach as well, in which the monolingual recognizers are used in parallel and the confidence scores of every recognizer are taken into account (H. Lin et al. 2012, p. 3, 4). The difference between the two approaches is that the last one also adds the one-pass system and uses its confidence score as additional information, since it seems to make "different errors than the monolingual systems" (H. Lin et al. 2012, p. 3). The results of these models are shown in Figure 6. The first line in each table shows the accuracy using only the confidence scores of the recognizers. The second line includes the margin scores which were also used for the language identifier model. The third line also uses an agreement feature which considers whether recognizers agree on certain language confidence scores, therefore making use of redundant information. The fourth variant uses an SVM with a Gaussian kernel which was trained on the development set to choose the language based on the confidence scores, agreement features, and the results of the language identifier module from before. The last variant is different from the others because it knows the language used beforehand and therefore has an advantage in identifying and recognizing the language input (H. Lin et al. 2012, p. 3). However, the accuracy for these models surpasses the baseline 49.5% accuracy. This is not only the case for the variant with a priori knowledge about the used language but also the one without a priori knowledge, as shown in the fourth line of Figure 6b. This shows that combining different approaches leads to far better results than only focusing on one approach. Especially combining vastly different approaches, such as the one-pass and multi-pass approach provides great performance. The biggest issue that the system has to manage, as the authors of the paper described, is that the input is often very short (3.5 seconds on average (H. Lin et al. 2012, p. 3)) which makes it very hard for the model to make an educated decision. Designing a model which is able to work with minimalistic input as well as provide low-latency results and precise results will be one of the hardest obstacles to conquer for my personal approach.

The next two papers focus on the use of LIAMs and the difference that the amount of training data has as well as on the performance of GMM-HMM-based approaches compared to DNN-based approaches. The following approach for multilingual speech recognition was reported by Knill et al. 2014. The authors created a LIAM on the basis of a GMM-HMM structure, trained it with a Multi-Layer Perceptron (MLP), and used it to transcribe audio input from an unknown language to train an unsupervised LDAM. The goal was to investigate whether it is possible to use information from other languages to substitute the missing knowledge when dealing with low resource languages, or even unknown languages. The presented models use very limited training data as well as limited audio transcriptions for their languages. The LIAM uses language information from a unified phone set from all 7 training languages (Knill et al. 2014, p. 2-3). In order to avoid learning wrong pronunciations for words that are represented in many languages, language-specific pronunciation infor-

System	SACC (%)			
	En	Fr	Zh (Zh/En/Mix)	Avg
En+Fr+Zh (conf)	58.0	37.0	42.1 (42.4/43.7/28.6)	45.7
En+Fr+Zh (conf+margin)	58.0	41.6	41.2 (41.9/39.9/22.1)	46.9
En+Fr+Zh (conf+margin+agree)	58.0	41.6	41.3 (41.9/40.5/22.1)	46.9
En+Fr+Zh (conf+agree+LangId,SVM)	58.9	44.9	42.1 (42.4/42.3/29.1)	48.6
En+Fr+Zh (oracle)	60.7	47.1	43.0 (42.6/51.4/32.6)	50.3

(a)

System	SACC (%)			
	En	Fr	Zh (Zh/En/Mix)	Avg
En+Fr+Zh+Mix (conf)	57.9	39.9	42.0 (42.0/45.6/30.1)	46.6
En+Fr+Zh+Mix (conf+margin)	57.8	42.4	41.5 (41.8/43.0/27.0)	47.3
En+Fr+Zh+Mix (conf+margin+agree)	59.0	44.3	42.4 (42.5/45.3/28.9)	48.5
En+Fr+Zh+Mix (conf+agree+LangId,SVM)	59.8	46.7	42.8 (42.8/46.9/31.7)	49.8
En+Fr+Zh+Mix (oracle)	63.4	51.5	46.4 (45.8/55.7/37.4)	53.8

(b)

Figure 6: The results from the multi-pass approach without (shown in **a**) and with (shown in **b**) the additional one-pass system as presented by H. Lin et al. 2012.

mation is stored by generating phonetic alignments (Knill et al. 2014, p. 2) for every specific language. Accordingly, every word is stored with as many pronunciations as there are languages it appears in. To extract the important features for training this LIAM, an MLP is trained. This MLP is also trained language-independently (Knill et al. 2014, p. 2).

The "LI" part of the results shown in Figure 8 focuses on the performance of the LIAM. The "UN" part is focusing on the performance of the unsupervised LDAM. The training procedure of the LDAM is shown in Figure 7. For training, only a single target language is chosen from the training set. The MLP extracts the important features of the input and then the LIAM, as well as the target language-specific lexicon and LM, gives its recognition score. The resulting transcriptions given by the LIAM are split up to form a new training set depending on their confidence score (Knill et al. 2014, p. 3). This new training set, transcribed by the

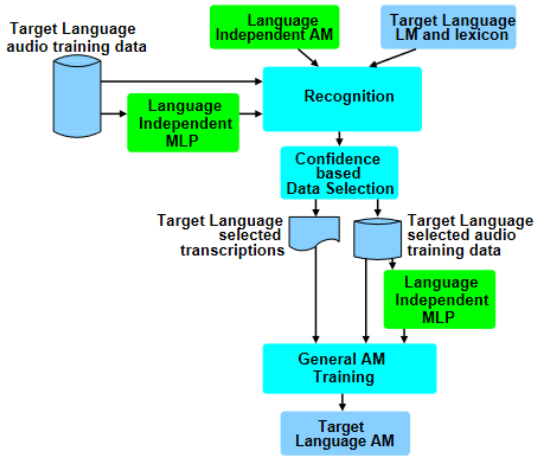

Figure 7: The training process of the unsupervised LDAM as presented by Knill et al. 2014.

trained LIAM, is used for training the LDAM. Here, the MLP is used again to extract the important features of the audio input.

Figure 8 shows the performance results for both of the acoustic models. It should be noted that not only were these models tested on three languages which were unknown to them, the languages also had several phones which were not covered by the training set (Knill et al. 2014, p. 3). Furthermore, the training set only consists of about 13 hours of audio per training language, which for 7 languages adds up to 91 hours in total; clearly less than commonly used. The unsupervised model was trained on about 25 hours of newly transcribed data for each language (Knill et al. 2014, p. 3). For every language, a standard LDAM was used to obtain a baseline accuracy.

AM		TER	MTWV		
Data	Type	(%)	IV	OOV	Tot
LD	fMPE	68.5	0.3173	0.0987	0.2504
LI	fMPE	81.1	0.1929	0.0775	0.1573
UN	ML	74.9	0.2226	0.1059	0.1872
	ML-MAP	75.1	0.2310	0.1034	0.1920

AM		TER	MTWV		
Data	Type	(%)	IV	OOV	Tot
LD	fMPE	61.7	0.4673	0.2347	0.4317
LI	ML	78.8	0.2126	0.0756	0.1916
	MPE	78.4	0.2067	0.0884	0.1885
	fMPE	77.2	0.2250	0.0966	0.2058
UN	ML	70.4	0.3118	0.1560	0.2880
	MPE	71.7	0.3021	0.1682	0.2815
	fMPE	71.3	0.2956	0.1524	0.2736
	ML-MAP	70.6	0.3123	0.1723	0.2911

AM		TER	MTWV		
Data	Type	(%)	IV	OOV	Tot
LD†	fMPE	69.3	0.1962	0.1081	0.1851
LI	fMPE	87.6	0.0255	0.0268	0.0257
UN	ML	84.7	0.0141	0.0109	0.0137
	ML-MAP	84.8	0.0138	-0.0277	0.0080

Figure 8: The results of the LIAM and the newly unsupervised trained LDAM as presented by Knill et al. 2014. It shows the results for Bengali (12 phones not in training set) in the upper part, the results for Haitian Creole (2 phones not in training set) in the middle part and the results for Vietnamese (7 phones not in the training set) in the lower part. LD: language-dependent acoustic model; LI: language-independent acoustic model; UN: Unsupervised language-dependent model; TER: Token error rate; MTWV: Mean term weighted value (keyword search); IV: In vocabulary; OOV: Out of vocabulary; Tot: Overall value; ML: Maximum likelihood; MPE: Minimum phone error; fMPE: feature-space projection MPE; ML-MAP: Maximum likelihood mapped.

Unfortunately, many details of the results are not explained in the paper; for example, what effect the MPE, ML, fMPE and ML-MAP types have, how

they are integrated into the system, and their general definition. Therefore, I will focus on the general high-level information gained from the tables.

Generally speaking, performance for all systems is subpar which can largely be credited to the low amount of training resources. Additionally, the language-independent and unsupervised language-dependent AMs are performing worse than the standard language-dependent AMs. What is interesting, however, is that the best performing unsupervised language-dependent models perform better than the best performing language-independent models for Bengali and Haitian Creole. Keeping in mind that the unsupervised language-dependent models worked with transcriptions written by the LIAMs for a language unknown to them, seeing these models trained on weak transcriptions outperform the original transcribing system bears some interesting implications. First of all, it shows that for building a multilingual PDA that can deal with underrepresented languages, missing information can be filled in by using agreement and recurring features from other languages. Naturally, the languages that are offered should be sufficiently supported in the training set, but even if a language is underrepresented it may still be used in a multi-pass approach. Furthermore, these results suggest that a working system can still exist even if the transcription from the language identifier is not good, which again makes for a viable multi-pass approach. However, I want to enforce the importance of enough training data being available for the models in order to obtain a model with reasonable performance.

After investigating the effects that the underrepresentation of languages can have and seeing the performance of GMM-HMM-based LIAMs, I will introduce one last paper for multilingual speech recognition which uses a DNN-based multi-pass approach. The model was reported by Gonzalez-Dominguez et al. 2014 and consists of a server-based language identification module as well as a set of monolingual recognizer modules. When receiving an input, the incoming audio stream is sent to the server-based modules directly. These modules work in parallel in order to not depend on each other's output. The language identification module is a DNN which is trained as an LIAM and returns scores for the possible languages of the input. The recognizer modules are also DNN-based and parse the input regardless of the fact whether they are trained for the specific language or not. They return their transcription of the input as well as a confidence score indicating the probability that the transcription is correct. All the incoming transcriptions and scores from a time frame are queued up and the best choice is selected by a strategy module according to specific rules. The selected transcription is then presented to the user in real time in order to show that the input is recognized and to enable the user to track the speech recognition process (Gonzalez-Dominguez et al. 2014, p. 3). The system also includes a monolingual path, but I will focus on the "multi-recognize" path.

As mentioned, the system's modules are based on DNNs. Both the language identifier and the recognizer systems are using an almost identical structure. In order to enable a recognizer scoring of distinct segments of the input, it is divided into 26 frames per segment and the models are trained using 5-gram language models. I mention this because using 5-grams as a window for parsing

has also proven useful in Bojanowski et al. 2017, p. 7-8.

It is necessary to keep in mind that DNNs are much more compact than GMM-HMM systems for a multi-pass approach, since the space needed for every single language recognizer grows drastically with the number of supported languages. Moreover, a more compact system leads to a decrease in decoding latency which becomes increasingly important for a larger language pool. This is crucial as well since the processing time can drastically increase for recognizers which have to deal with an unknown language (Gonzalez-Dominguez et al. 2014, p. 3).

The strategy module is designed to counteract precisely this issue. The authors of the paper present three strategies for timeouts when waiting for final outputs from speech recognizers:

1. Always wait for every final recognizer score to come in. As discussed, this can lead to latency in presenting a transcription for a query.

2. Have the timeout depend only on the fastest recognizer. A timer starts after receiving the first final transcription and confidence score. Until this timer runs out other recognizers can still send their outputs to the front end, anything arriving after that will not be considered for presentation. The researchers indicated that setting the timer to one second provided the best results for this strategy (Gonzalez-Dominguez et al. 2014, p. 4). This strategy is very viable especially when dealing with many languages since this way the only final transcriptions taken into account are most likely to be the correct ones since they came in the fastest.

3. The timeout is based on the available saved partial and final scores. While a partial transcription state can always be given if the highest partial score changes, taking into account final scores can prove to be more difficult. In order to solve this problem and also account for generally difficult inputs, the strategy module takes into account the difference between the highest available final score and the highest available partial score from a recognizer that is still processing and computes a context-dependent timer for the timeout.

The use of such a strategy module to deal with latency issues is particularly useful for a framework dealing with a large number of languages, which is the aim of this thesis. Nevertheless, when dealing with code-switched input there has to be some measure to ensure that the entire input is processed in order to not lose information. This idea will be further discussed in Section 4.

The results for this approach are presented in Figure 9. The first column reports accuracy when the strategy module only selects the first incoming final recognizer, which is still better than chance (Gonzalez-Dominguez et al. 2014, p. 7). The second column shows accuracy when the strategy module selects only based on the recognizer confidence scores. The most interesting result, however, is that *a)* if the strategy module only decides based on the language identifier scores accuracy rises to 88%, as can be seen in the third column, and

b) when combining the identifier and recognizer scores the system can achieve an accuracy of 90%. These results show two things. On the one hand, combining different scoring systems and architectures provides a considerable boost in accuracy for the system. On the other hand, it shows how capable DNNs are for different tasks while also using the same construction.

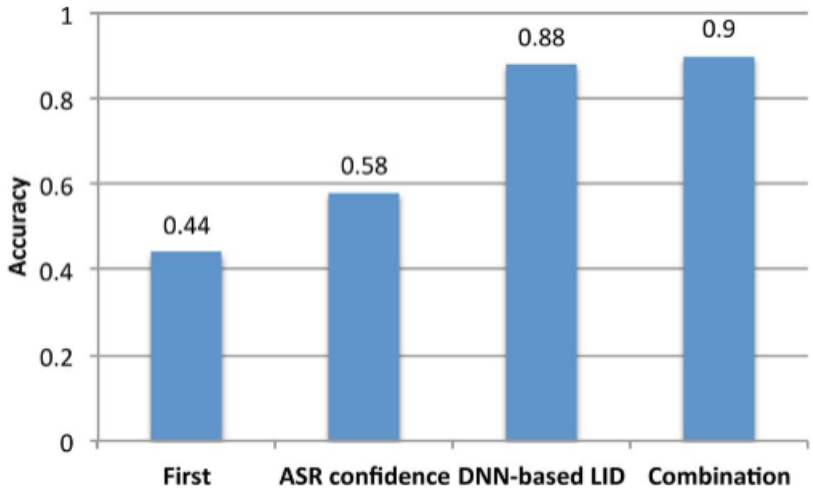

Figure 9: The results for a multi-language speech recognition system as presented by Gonzalez-Dominguez et al. 2014.

In this subsection, I examined speech recognition models. I introduced the architectures as well as their performance results and implications for my own work. The main aspects to take from these research results are *a)* that combining different representations and architectures leads to more successful performances, *b)* it is possible to substitute unknown phones with known ones according to Bhuvanagirir and Kopparapu 2012, *c)* exploiting redundant features and using agreement between different recognizer scores can lead to promising results, *d)* a working recognizer model can still exist even if the transcription from the acoustic model is not perfect, *e)* DNNs provide flexible frameworks which can be used successfully for more than one task while using the same setup, and *f)* using a strategy module for handling big amounts of different scores and results helps keeping the system flexible and user friendly. In the following subsection, I will describe how user input is used and explore some models for pre-processing and handling multilingual written input.

3.5 Pre-Processing Models

In this subsection, I want to examine models for the LU module. However, I will not introduce general LU models, but rather my own design which in principle consists of inserting a pre-processing module before the LU algorithms start working. I decided to focus on building a pre-processing module instead of a whole multilingual processing module because, as seen in Section 3.4, achieving satisfactory performance for multilingual applications is very difficult. Since the performance of the entire PDA depends on the performance of all the modules, I believe it is beneficial to facilitate the PDA's work whenever possible. When recognizing multilingual vocal input, there is hardly any facilitation possible since the entire input has to be recognized before continuing to work with it. However, having the LU module process multilingual and code-switched input and possibly introducing a higher error probability is avoidable. My pre-processing module is designed to identify and unify the query languages. This ensures that the domain classifier and slot and intent taggers only have to work with monolingual input, which is how PDAs work already with great success.

My approach consists of three steps:

1. A named entity recognizing step where the input is analyzed for named entities
2. A language detection step where the input is analyzed for the use of languages and to detect if there are multiple languages used
3. A translation step which only becomes active if the language detection step has recognized switching or mixing in the input. In this step, the languages are unified and thus become easier to analyze for the LU algorithms

I will examine the papers of every single step in their own subsection.

3.5.1 Named Entity Recognition

Named entity recognition is the first step taken in the language understanding module. In this step, the written or transcribed input is analyzed and named entities are removed from further analysis because named entities can often be of a different language than the rest of the input (e.g. "Where is the "Berliner Tor"?"). This could complicate further analysis since these named entities will then be flagged as a different language and translated. Oftentimes, the names of named entities in different languages are not a word-for-word translation, which can lead to unwanted results. Therefore, recognizing named entities is an important step in the analysis.

The first model I will present was reported by Saito and Nagata 2003. They use a HMM framework for their approach, a named entity recognition (NER) system for Japanese, Chinese, Korean, and English. The idea was to split the system into a language-dependent and a language-independent part, as well as to convert the character coding to make it possible to work with input from different languages. This idea makes the system universally applicable to any

language as long as training data is present and the coding is convertible (Saito and Nagata 2003, p.2, 8).

Figure 10 depicts the architecture of the proposed system. The input is first converted from the local language coding into Unicode to allow for analysis of any language coding. The local coding can either be specified by the user or automatically detected. Next, the lexical analyzer uses several language-specific features to identify words and word candidates for named entity recognition. It pays attention to the different use of spaces as well as word length and creates candidates for words that are not in the dictionary by using substrings of the input (Saito and Nagata 2003, p. 2-5). The identified word candidates are then sent to the language-independent analysis part which performs a morphological analysis and selects the n-best morpheme sequences. This is done in order to circumvent the ambiguity that often comes with the segmentation of Asian languages (Saito and Nagata 2003, p. 5, 6). The n-best morpheme sequences are presented as a word graph structure which is analyzed by the NER system. The NER system itself is based on BBN's HMM NER system for English, called IdentiFinder (Miller et al. 1998), which offers a very high accuracy (Saito and Nagata 2003, p. 5). The morphological analysis and the NER are performed at the same time (Saito and Nagata 2003, p. 5).

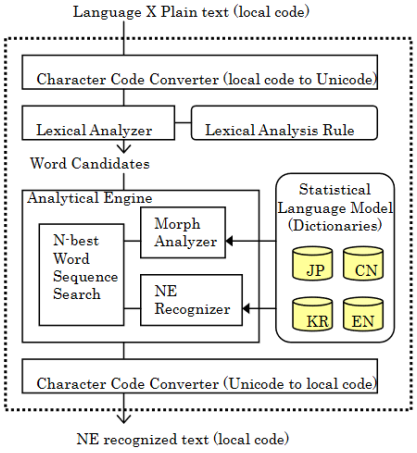

Figure 10: The system architecture of the NER system as presented by Saito and Nagata 2003.

The performance results of the model are presented in Figure 11. The proposed approach was compared to a state-of-the-art SVM NER system. The presented model performed equivalently well as the SVM system, however, the authors mentioned that their system runs ten times faster than the SVM system, which, in conjunction with the high accuracy, makes this approach a very practicable model for my own system. Converting the input and thus making the system able to handle almost any kind of language coding as well as the possibility to adapt this system for more languages very easily only increases its viability.

Another approach is presented by DiPadova and Jiang 2018 and it uses a Long-Short-Term-Memory (LSTM) network for NER. These LSTM networks are variations of a recurrent neural network (RNN) and are able to learn long-term dependencies (Olah 2015). This makes them very useful for identifying named entities since they are often implied by the sentence structure, which can also be seen from the paper's results. In the paper, they use a bi-directional LSTM, which means that the system goes through the input from the beginning to the end and from the end to the beginning. This enables the system to be

	HMM	SVM
EN	88.2	84.7
JP	81.0	57.3
CN	84.5	89.5
KR	79.9	82.1

Figure 11: Performance results in F-measure % for the NER system as reported by Saito and Nagata 2003.

universally applicable for any language since it can handle sentence structures from many languages where, for example, the word order is different (DiPadova and Jiang 2018, p. 5). The exact structure of the network is shown in Figure 12.

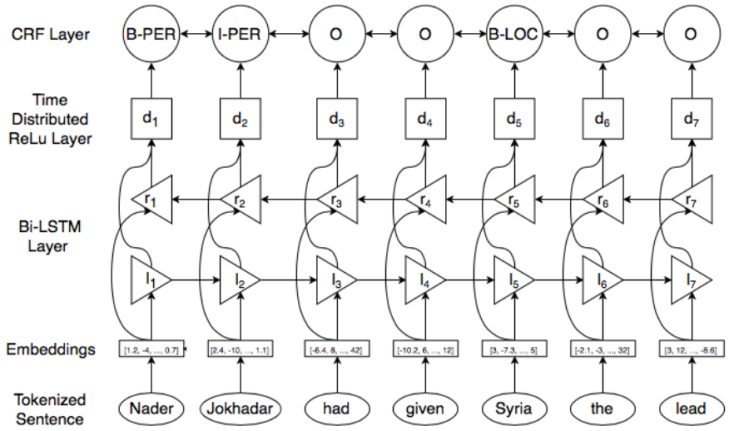

Figure 12: The system architecture of the NER as reported by DiPadova and Jiang 2018.

The input is embedded using the Facebook Research Multilingual Unsupervised Supervised Embedding (MUSE). MUSE is used to create word embeddings from Wikipedia entries to obtain a vector space for each of 30 languages, which are then unified into a single vector space with vectors of 300 dimensions (DiPadova and Jiang 2018, p. 3). This vector space makes use of shared information between languages since words with similar meanings are closer together across all languages. If the system encounters an unknown word, a new random vector is generated in a 0.25 radius hyperball around the origin of the vector space. This is done every time an unknown word is encountered, even for the same word, in order to avoid overfitting on unknown words (DiPadova and Jiang 2018, p. 3, 4).

After embedding the input words, they are passed on to the bi-directional

LSTM which forwards its outputs to a time-distributed dense layer with hidden nodes. The last layer in the model is a Conditional Random Field (CRF) classifier. The CRF layer labels every word from the input using the IOB tagging system (DiPadova and Jiang 2018, p. 2). The tags I, O, and B stand for Inner, Outer, and Beginning, which signify if a word is a part of (Inner), the start of (Beginning), or not a part of (Outer) a named entity. Additionally, the system is also able to classify different classes of named entities, such as people (PER), organizations (ORG), locations (LOC), and miscellaneous (MISC) (DiPadova and Jiang 2018, p. 2).

For a part of the experiments, the model was trained on the English, German and Spanish datasets from the Conference of Computational Natural Language Learning (CoNLL) from 2002 and 2003 (DiPadova and Jiang 2018, p. 2). Another part of the experiments only used the English dataset for training. Furthermore, the researchers tested the model's performance for two different tasks: multiple classification, meaning the system has to assign the full labels to every word, or binary classification in which the system only has to identify if a word was part of a named entity or not (DiPadova and Jiang 2018, p. 5).

The results of the experiments are shown in Figure 13. In general, the results demonstrate that it is easier for the system to work with a binary classification, which is perfect for my own model design since the named entity recognition step only has to detect if there are any named entities appearing in the present input. As to be expected, when trained on only the English data set, classifying English named entities is a lot easier than classifying named entities in other languages. However, the decrease in accuracy, precision, recall, and F1-score from using more than the English data set for identifying English named entities is very minimal, making this approach very useful for many different languages. In a more detailed analysis of the results, the authors identified miscellaneous named entities to be the biggest problem for the system and that the system was almost perfectly able to identify words which were not part of a named entity (DiPadova and Jiang 2018, p. 6). I will not go into the details about those since my focus is on the general identification of named entities. One thing that I believe will be very useful is the embedding of words of different languages in the same vector space, again making use of redundant information between languages. This is further emphasized by an additional observation made by the researchers. They tried using an Italian sentence that makes use of the Italian embedding vectors. Their system was able to correctly identify every named entity in that sentence, even though it was never trained on an Italian data set (DiPadova and Jiang 2018, p. 7). This is a very interesting finding since it implies that with a good implementation of word embeddings there is no need to train a system on every available language, further establishing that exploiting similarities between languages holds many possibilities for building a multilingual PDA. As mentioned above, the system was able to identify named entities that were not even present in the word embedding vector space (and thus completely unknown) by learning the "semantic structures" which imply the use of a named entity (DiPadova and Jiang 2018, p. 7). This is a very useful finding since a voice or written query, which includes a named entity, might also

have certain semantic characteristics which can be used by the algorithm to quickly recognize named entities and then exclude them from further language detection and translation.

		English				German				Spanish			
	Data	Acc	Prec	Rec	F1	Acc	Prec	Rec	F1	Acc	Prec	Rec	F1
All Tags	English	**0.990**	**0.887**	**0.742**	**0.801**	0.868	0.224	0.344	0.219	0.954	0.585	0.402	0.452
	Full	**0.990**	0.879	0.741	0.792	**0.979**	**0.723**	**0.440**	**0.517**	**0.975**	**0.817**	**0.680**	**0.723**
Binary Tags	English	**0.991**	0.939	**0.881**	**0.908**	0.870	0.404	0.594	0.424	0.961	0.832	0.653	0.717
	Full	**0.991**	**0.947**	0.874	0.907	**0.980**	**0.874**	**0.660**	**0.737**	**0.980**	**0.885**	**0.874**	**0.879**

Figure 13: Performance results for the LSTM NER system as reported by DiPadova and Jiang 2018.

The results of the two presented named entity recognition approaches, binary and multi-class, are very promising for a multilingual PDA. Both of them are able to achieve very high accuracy. I believe that *a)* using the redundant information between languages is the key to success, *b)* it is important to account for different uses of grammar rules and spacing, and *c)* instead of identifying and classifying named entities, better results can be achieved by merely identifying them.

3.5.2 Language Detection

After named entities are detected in the input and excluded from further analysis, the query has to be analyzed for possible code-switching and mixing in order to decide if further translation is necessary. In the following section, I will introduce two approaches that focus on detecting language switches and also labeling the found languages in order to make the job of the translation module easier.

The first approach was presented by Rijhwani et al. 2017. The goal of the researchers was to build a code-switching detector that works on a word level and is able to detect code-switching for an arbitrarily large set of languages. Their system is called a generalized word-level language detection system (GWLD) and features language detection for *a)* an arbitrarily large set of languages, *b)* any number of languages and language switches in a single input, *c)* without the need for manually annotated data, and *d)* without having to know the used languages a priori (Rijhwani et al. 2017, p. 3).

The researchers use an HMM framework for their approach. It takes a text as an input and has a set of labels as an output of which every label corresponds to a token from the input (Rijhwani et al. 2017, p. 3). The general architecture of the model is shown in Figure 14. Every language has its own HMM with two states (l_i and xl_i), each of which labels one word before transitioning to the next state. The language-dependent HMMs are connected in a bigger Meta-HMM which makes language switches possible. The input is tokenized and every token has a label assigned to it in the process. The l_i label stands for a specific language label and the xl_i label stands for a universal token belonging

to the last known language. A universal token is defined as anything that is not a natural language expression, such as URLs, usernames, and sounds (such as "hahaha") (Rijhwani et al. 2017, p. 3). The HMM begins at the start state *s*. Then, the first language is recognized and the corresponding HMM becomes active and labels the first token. Next, a transition to the next state can be made for labeling the next token. The possible transitions are either *a)* staying in the same language and label the next token with the same label, *b)* transition to the universal token label and label a universal token of the same language, or *c)* transition to a different language because a switch has occurred. If the last labeled token was a universal token, the only transitions available are back to the same language or detecting a language switch since in the tokenization step that took place before all successive universal tokens have been grouped together to one bigger universal token (Rijhwani et al. 2017, p. 4). The end state *e* is responsible for the label sequence output.

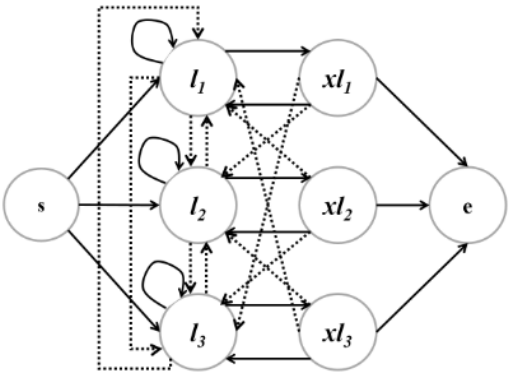

Figure 14: The architecture used for the language detection system as reported by Rijhwani et al. 2017.

The system was trained on seven languages (Dutch, English, French, German, Portuguese, Spanish, and Turkish) on a corpus taken from Twitter. This corpus included tweets which were identified as belonging to one of the seven languages the system was trained on by the Twitter LD API. The tweets were labeled using the "CovertSet" algorithm, a naive Bayes classifier (Rijhwani et al. 2017, p.5). If the label assigned from twitter and the "CovertSet" algorithm were the same, the tweet was included in the training set with its assigned label. Furthermore, for re-estimating the weights of the HMM system, an unlabeled set was taken from the direct twitter corpus. In the end, the authors used a training set with 700k (100k per language) automatically labeled monolingual tweets and a re-estimation set of 100k (containing all seven languages) unlabeled tweets which may be monolingual or code-switched or have the wrong label assigned by Twitter (Rijhwani et al. 2017, p. 5). The language detection system was then trained using the training set and for a different part of the experiment, the weights were re-estimated by using the unlabeled set and the

Baum-Welch algorithm, which is a variation of the Expectation-Maximization algorithm (Kwok 2019; Rijhwani et al. 2017, p. 4). The system was compared to three other systems which do not perform word level language detection, but return a confidence score for possible languages, which can be used to estimate if the tweet was labeled as monolingual or code-switched and which languages are involved (Rijhwani et al. 2017, p. 6). Furthermore, the model was compared to two dictionary-based approaches, which used the most frequently used language label for a single word, or the most frequently used language label from a pre-selected set of possible languages as language labels (Rijhwani et al. 2017, p. 6). The results of the experiments are shown in Figure 15.

Systems	*Acc*	L_1L_2Acc	*IsMix*
Dictionary-based Baselines			
MAXFREQ	0.824	0.752	0.600
MINCOVER	0.853	0.818	0.733
Existing Systems			
LINGUINI	NA	0.529	0.783
LANGID	NA	0.830	0.783
POLYGLOT	NA	0.521	0.692
GWLD: The Proposed Method			
Initial	0.838	0.825	0.837
Reestimated	**0.963**	**0.914**	**0.88**

Figure 15: Performance results of the language detection system presented and as reported by Rijhwani et al. 2017, *Acc* stands for the fraction of words labeled correctly, L_1L_2Acc stands for the fraction of tweets with their language(s) labeled correctly, *IsMix* stands for the fraction of tweets correctly labeled as code-switched or monolingual.

The results show that out of all experimental settings the proposed model performed best at 96.3% accuracy when the weights had been re-estimated using the unlabeled set. Even without re-estimating the weights, the system performed worse only by a fraction of the compared models. These results indicate that a multi-pass system, of which every part has been trained on a monolingual corpus, is able to correctly detect languages and also detect language changes and label them correctly if it has seen a few mixed and code-switched inputs. Following the experiments, the researchers also tested the robustness of their model by reducing the number of tweets in the training and unlabeled sets. It turned out that even when only using 0.25% of the training set (around 250 tweets per language), the system performance was almost at 96% accuracy. Moreover, when using only around 10% of the unlabeled set the performance of the model rose to almost 96% as well (Rijhwani et al. 2017, p. 7). This shows that the proposed method is very flexible when it comes to training and

is easily and quickly trained. What is interesting is that if the model is only trained on two languages from the start and then tested for accuracy for labeling tweets from these two languages, its accuracy is very high. However, when adding more languages to the system, the labeling accuracy does not decrease significantly (Rijhwani et al. 2017, p. 7), making this model very useful for a multilingual scenario with many supported languages. Additionally, the system works at a very high speed, labeling 2.5M tweets per hour (Rijhwani et al. 2017, p. 8), which makes around 694 tweets per second. This high processing speed is very useful for a PDA setting since accuracy and performance speed are of the essence there.

The next model was presented by Chee Chang and C.-C. Lin 2014, and it uses a Recurrent Neural Network (RNN) framework. The system is based on Jordan and Elman-type RNNs (Chee Chang and C.-C. Lin 2014, p. 3) which are special types of RNNs. Generally, RNNs are related to LSTMs, they are able to capture the context of an input. Elman-type RNNs use a special context layer for doing this which saves the outputs of the hidden layer and feeds it back into the hidden layer in the next time step. The Jordan-type RNNs also use a context layer, called a "state layer", which captures not the outputs from the hidden layer but from the output layer. This context is passed back into the hidden layer in the next time step as well as to the state layer itself to capture not only the immediate context but also long-term dependencies (Cruse 1996, p. 115). The goal of the researchers was to build an RNN that works with character level n-grams and pre-trained word embeddings while being able to capture the context of an input and modeling long-term dependencies. The system architecture is shown in Figure 16. The input and three context words on either side of the current word are captured and transformed into a seven-dimensional 1-hot vector (Chee Chang and C.-C. Lin 2014, p. 3). A 1-hot vector is a representation of a word that consists of an n-dimensional vector with every cell set to 0 except for one which is set to 1 (Uriarte-Arcia, López-Yáñez, and Yáñez-Márquez 2014, p. 4). However, it is not clear from the paper if the seven-dimensional context vector is the 1-hot vector or if the seven-dimensional context vector contains a 1-hot vector for each of the seven words. For the first option, I do not see how it will contribute useful information. It is also not clear how the beginning of a sentence and the end of a sentence is handled when there are no context words for one side. In general, it does not become clear why the 1-hot vector has been chosen as the vector representation.

Additionally, the current word's n-grams are extracted and appended to the context vector. The n-gram vector consists of the tri- and bigrams from the beginning and end of the current word which are also represented as 1-hot vectors, leading to the character n-gram vector to become a 12-dimensional 1-hot vector. Again, it is not clear why a 1-hot vector representation is chosen. After creating the context and n-gram vector, they are projected into a 100-dimensional vector using a supervised embedding layer, which is not described in further detail. After the constructed vector has been projected, it is fed into the hidden layer with the additional output of the word2vector module. Word2vector is a skip-gram word embedding that focuses on grouping words

together depending on context (Chee Chang and C.-C. Lin 2014, p. 4). This is done across languages to generalize words since it is assumed that words share contexts across languages. The hidden layer is also provided with context information either from the previous hidden layer time step (Elman-type) or from the previous output layer time step (Jordan-type) and feeds its outputs to the output layer.

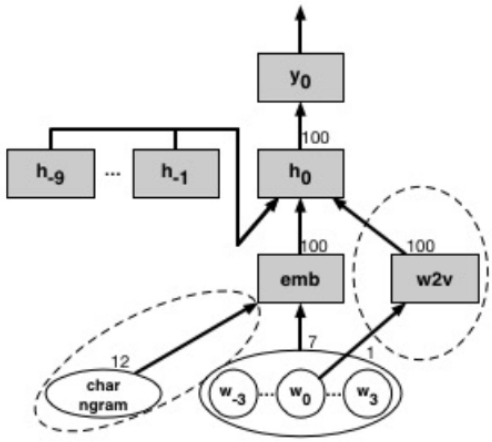

Figure 16: The Elman based RNN system architecture as reported by Chee Chang and C.-C. Lin 2014. For Jordan-type version the recurrent structures ($h\_9...h\_1$ in the image) are connected to the output layer before feeding back into the hidden layer.

The proposed system was first tested for general performance with different configurations. While standard Jordan and Elman-type RNNs performed at around 88 to 89 F1-scores, using the n-gram embeddings provided an increase to around 92 F1-score. Using the word2vector embedding for the Elman-type RNN provided another increase on F1-score to around 93.5 (Chee Chang and C.-C. Lin 2014, p. 5), showing that using different metrics again provides further improvement in performance.

The Jordan and Elman-type RNNs together with the n-gram and word2vector embeddings were compared to state-of-the-art SVMs as well as to LangID and Lexical as baselines. The results are shown in Figure 17. The systems were trained on tweets taken from the EMNLP Code-Switching Workshop dataset. The proposed systems outperformed every system (performing at around 95 to almost 96% accuracy) they were compared to, proving that using different metrics provides better results but also that RNNs with their memory structures provide a very useful approach to language detection. Using character-level information as well as exploiting shared information between languages helps to increase performance accuracy. It is not reported how quickly the systems performed, which is unfortunate, since performance speed is also an essential part

of a PDA. The fact that the system was trained on five different languages (Chee Chang and C.-C. Lin 2014, p. 3) and only evaluated when having to detect two of them and still performed so well is another point in favor of RNNs.

Systems	English-Spanish	English-Nepali
Jordan+ngram+w2v	**95.2%**	**96.6%**
Elman+ngram+w2v	**95.2%**	96.4%
SVM (workshop: dcu-uvt)	92.5%	96.3%
SVM (workshop: TAU)	94.2%	not reported
Baseline (LangID)	75.9%	70.0%
Baseline (Lexical)	72.6%	68.5%

Figure 17: Performance results for the proposed language detection system as reported by Chee Chang and C.-C. Lin 2014.

In this subsection, I examined two different approaches to language detection and code-switching tagging. Both of the systems used Twitter as a data source but approached the topic differently. Nevertheless, an important shared point is that both systems used context to label their inputs. Additionally, the presented systems both performed very well (96.3% accuracy for the HMM-based model and 95.9% average accuracy for the RNN based model), but the HMM system performed very fast as well. Both systems also have simple ways to extend their supported languages, the HMM-based system has to add another HMM for every new language and the RNN based system has to add the word embeddings to the word2vector module.

The next step in my design for a multilingual PDA is meant for instances where a language switch has been detected. In the next section, I will describe an approach to a translation disambiguation model which can be used to unify the used languages of an input to facilitate analysis.

3.5.3 Translation

The last step before the general processing of an incoming query is the translation step. By now the input has been analyzed for named entities, which, if any are found, have been taken out of further processing. After that, the input also has been analyzed for language switches and labeled accordingly. The only step missing now is the step in which the used languages are unified in order to facilitate further processing. Fung, X. Liu, and Cheung 1999 present an approach for exactly that. The aim was to create a system for cross-language information retrieval which is able to identify a primary and a secondary language in a mixed language query and translate the secondary language parts so that the translations are unambiguous and the semantics of the sentence are not changed. To achieve this, the researchers propose a simple co-occurrence metric (Fung, X. Liu, and Cheung 1999, p. 2). The model should be able to solve three identified steps (Fung, X. Liu, and Cheung 1999, p. 2):

1. generating translation candidates in the primary language
2. weighing translation candidates
3. pruning translation alternatives

To solve these tasks, the following ideas are employed: An online dictionary is used to generate translation candidates, the Mutual Information (MI) metric is used to weigh the translation candidates in combination with the context words since words with similar meaning should occur in similar contexts, and three different ideas for pruning the candidates are proposed (Fung, X. Liu, and Cheung 1999, p. 3). The three pruning methods are:

1. Single neighboring word: The direct neighbor to the right or left of the target word is used to get the MI score with every translation candidate, the candidate with the highest score is chosen. This method is used as a baseline in the experiments. (Fung, X. Liu, and Cheung 1999, p. 3)
2. Voting: The MI of every word from the sentence or query and every candidate is computed. For every context word the translation candidate with the highest MI score gets a "vote", represented as a 1 in the co-occurrence matrix, while all other translation candidates receive a 0 from this context word. In the end, the translation candidate with the most votes from all the context words is selected. (Fung, X. Liu, and Cheung 1999, p. 4)
3. 1-best contextual word: To account for syntactic importance (a context word like "the" is not as expressive as "food", for example) after every MI is computed the same way as for the voting method, the "disambiguation contribution ratio" is computed. This is done by using the highest scoring and second-highest scoring MI from a single context word. The ratio is obtained by $\frac{MI_{highest}}{MI_{2ndhighest}}$ (Fung, X. Liu, and Cheung 1999, p. 5). If the result is much greater than one, it is considered to yield more disambiguation power. After this is done for every context word, the one with the highest contribution ratio is chosen and the translation candidate with the highest MI with this particular context word is selected as the final translation candidate. (Fung, X. Liu, and Cheung 1999, p. 5)

The system was trained on a monolingual English corpus taken from the Wall Street Journal 1987-1992. The mixed language queries were automatically created by taking 500 sentences from the ATIS corpus, a corpus with transcribed queries about flight information, and a portion of words in every query is randomly chosen to be translated to Chinese. This ratio ranges from 10% to 65% at which point the primary language effectively switches to Chinese (Fung, X. Liu, and Cheung 1999, p. 5). One experiment was done with every pruning method. The performance results are shown in Figure 18. Both the voting and 1-best method performed significantly better than the best neighbor baseline, with the 1-best method performing best out of all three. Performance was

best for 90% primary language words with close to 90% accuracy, but even at 50% primary language words performance is almost at 85% accuracy which is impressive for the time the paper was being written. The results imply that using context words for translation disambiguation is a viable method, especially when weighted according to their contribution power. Additionally, it implies that a system can perform well even when only trained on one language and using an online dictionary to fill in gaps caused by different languages. In order to make this idea more suitable for a PDA system, I will suggest some changes and ideas which I will describe in more detail in Section 4.

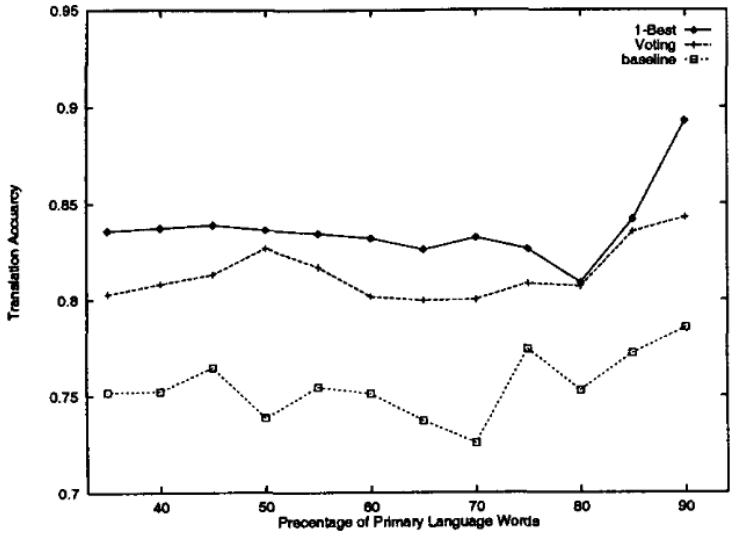

Figure 18: Performance results for the translation disambiguation system as reported by Fung, X. Liu, and Cheung 1999. The y-axis shows the translation accuracy, the x-axis shows the percentage of primary language words. The full line shows the results of the 1-best method, the dotted line shows the results for the voting method and the dotted line with squares shows the performance of the best neighbor method.

The presented paper for the Translation step is the only one I will include in this subchapter. The general research field of the topic is cross-language information retrieval. However, almost all papers found up to the point of writing this thesis focused on queries which were entirely in a single language and had to be translated completely. This is not the focus I need for my approach since the system I aim to build should be able to handle many monolingual queries by design. Most papers that I found did not address code-switched queries, which is the main theme in this chapter: A very specific case in which a query is not monolingual even after excluding possible named entities in a different language. The few papers that I found were: *1)* a master's thesis, which was supervised by one of the authors of the presented paper, that proposed almost identical ideas

and used the same graphics as well, *2)* a paper written by the same authors from 5 years later where they extended their idea from their 1999 paper, but their model performed worse, and *3)* a paper that dealt with English-Arabic mixed language queries for cross-language information retrieval. However, the paper focused on the documents returned by the model, not on translating the query to make it monolingual or on the quality of a query translation, which is why this paper was excluded as well. Using a paper which was written in 1999 is not necessarily representative of today's standards, however, the paper introduces a very simple metric which, if used correctly, can yield very promising results. I will go into further detail on how this could be done in the next section.

4 Discussion

In the preceding chapters, I have presented different approaches for multilingual speech recognition, as well as named entity recognition, language detection, and translation disambiguation. In this chapter, I will examine other aspects of a multilingual PDA, such as where the system is based and the use of the user profile before comparing the results of the literature review and designing a structure for my own working speech assistant.

A PDA is a complex system, drawing from many different sources in order to offer a satisfying user experience and using a number of modules with complex algorithms. In order to use all this information appropriately and to ensure that the PDA does not take up too much space on the device it is used on, PDAs are server-based (Sarikaya 2017) as well as proposed system architectures for multilingual PDAs (Gonzalez-Dominguez et al. 2014). In my approach, I aim to keep it this way both for space as well as processing power reasons. Having a server back-end means that a lot more languages can be supported and the processing speed can remain high enough for an adequate user experience, provided the user has a working internet connection.

Not every user will make use of every supported language when using the PDA. Most users will use one, two, or up to three languages at a time. This means that processing speed and accuracy can be further increased by making use of the user profile mentioned in Section 2. The user profile can either be set up in more detail by the user when setting up the PDA for the first time or the user profile can collect data about used languages and build a database of preferred languages in order to facilitate the processing step (which is also a more flexible solution). This way it is easier to account for possible changes in the languages spoken by the user, if they learn a new language, for example. The data from the user profile introduces a certain bias into the processing step, enough to make everyday use easier but not enough to inhibit adaption to new situations. The bias itself can consist of specific language margins, as seen in H. Lin et al. 2012, which become higher for languages which are not used often so that during processing, it is less likely to get a transcription of a language that is unlikely to be part of the query. If, however, the user should use a lesser-used language and it is detected appropriately, the system can still

react and process this input which will take longer than normal, but should not render the transcription useless. In addition to the language information, the general user profile can be extended to contain often used queries in different languages. This would facilitate processing even further since once a known query is detected the system can already choose the correct domain and only has to fill the slots with the query-specific information.

Since language is ever-evolving and changing and the use of code-switching is as well, I propose regular updates to training provided by the users which will be used to adapt the algorithms of the PDA to eventual changes in language so that continuous support through time and linguistic development is guaranteed. This additional training can be done by announcing the upcoming training phase to all users. The announcement would include the reason for the announcement itself, indicating that user traffic through the PDA will be used for optimizing the performance and an option to agree or disagree with the procedure. The users who agree will then contribute to the optimization of the system. There is a threat of overfitting certain languages especially at the beginning when data is not as widely available for every language and some are underrepresented. In this case, the data that is collected has to be pre-processed and languages have to be weighted appropriately. Depending on the development team, the processing and identification of languages could be done manually.

While the adaption of languages can certainly impact the performance of the PDA, the length of an input can also greatly affect performance. As H. Lin et al. 2012 have shown in their paper, the duration of spoken queries can influence the outcome of a speech recognition system. Figure 19 shows the length of utterances taken from different Google speech recognition data sets which consist of spoken language speech assistant queries. When only taking into account the actual query without the voice activation, utterances are only a bit over 2 seconds long, so the system I aim to build has to detect the used languages with minimal context. This process can be facilitated by including the voice activation command and using it to set the primary language of the query. By including different activation words (such as "Hello" for English and "Hola" for Spanish) a focus can be set for the query before processing begins. This would mean that the speech assistant has to be able to work with multiple languages but detecting part of the query is determined by the word choice of the user. Naturally, if the acoustic models do not detect the pre-determined language, the PDA has to be able to still provide a correct transcription, however, using a bias can speed up the transcription process for the biased language. This also means that for general processing there is more context for the acoustic models to work with since by including the voice activation commands the mean length of a spoken query increases to almost 4 seconds.

Now I will describe what my design for a multilingual PDA looks like. Before going into detail, I will shortly summarize the central findings of the papers I examined and use them to describe the system architecture of my speech assistant. I will focus on the best possible user experience by looking for a setup which combines high accuracy with high performance speed and disregard needed server space for the most part.

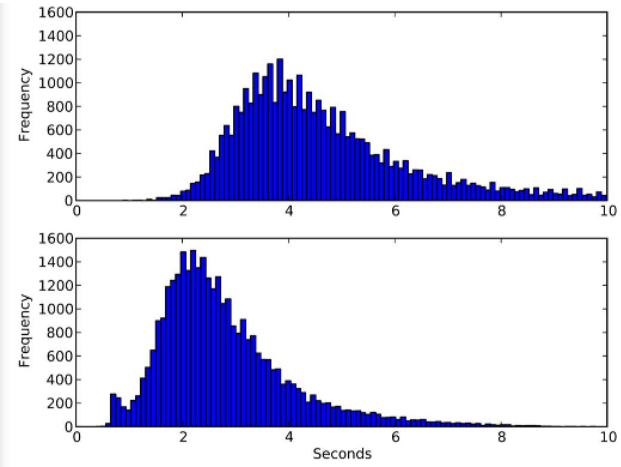

Figure 19: Utterance durations for speech assistant queries taken from the Google 5M LID data set and Google Multilang Speech Recognition set as reported by Gonzalez-Dominguez et al. 2014. Lengths shown including (above) and excluding (below) the voice activation command.

The first module I will recap shortly is the speech recognition module. Several approaches have been presented using one-pass and multi-pass approaches. The first model was reported by Bhuvanagirir and Kopparapu 2012. It is a one-pass model for English and Hindi and uses a specifically crafted pronunciation dictionary. The researchers recorded accuracies up to 55% (100 - recorded WER). While the performance was not optimal, one of the most interesting results was that missing phonemes in one language could be replaced by combining different existing phones and phonemes from the other language with the system still performing adequately. Nevertheless, it was also reported that when using a pronunciation dictionary comprised of a phone superset, performance increased.

The second model introduced in this paper was reported by H. Lin et al. 2012. The researchers built several models, one-pass as well as multi-pass. They used a GMM-HMM framework and trained their models to recognize English, French, and Mandarin. In the end, the best-performing model was a combination of one-pass and multi-pass systems. This system performed at almost 50% accuracy. The most interesting findings of this paper were that if different system architectures are combined and different features used, performance increases. One metric I find particularly interesting is the use of margins in order to counteract bias in languages which also increased performance accuracy but were not included in the final best performing model.

The third paper was by Knill et al. 2014. The researchers built an MLP with a one-pass approach and an LIAM with a GMM-HMM framework and used them to train an unsupervised LDAM. Training was done using very small

training sets. In general, performance was not good, which enforces the fact that for an adequately performing model enough training data has to be present and possible variations have to be sufficiently presented in the training set as well. Interestingly, the unsupervised model which was trained on faulty transcriptions from the supervised models performed better than the supervised models in some cases. This implies that a working speech recognition system can still exist even if transcriptions are not perfect.

The lastly introduced model was reported by Gonzalez-Dominguez et al. 2014. The system uses a DNN framework and a multi-pass approach. The researchers introduced a strategy module which collects language scores and transcriptions along with confidence scores for the transcriptions, decides which transcription is the correct one and when to cut off processing. The system performed at an accuracy of 90% for eight trained languages. Furthermore, the researchers introduced the Google 5M LID data set and the Google Multilang Speech recognition set, which consists of queries for speech assistants, which would make for an amazing training set for my model, since it includes 25 languages. The most useful takeaway from this paper is the strategy module which promises to be very useful for my design.

Taking the results of the papers into account, I propose the input recognition module of my multilingual PDA to be based on a DNN framework and use a multi-pass approach. The performance accuracy for voice inputs has to be very high and according to the results shown in the presented papers a DNN can provide such accuracy. The general architecture of the input recognition module is close to the one proposed by Gonzalez-Dominguez et al. 2014, since it showed the most promising results. The LIAM will be responsible for assigning scores to the possibly used languages of the input. Depending on the processing power available this can be extended to include LDAMs as well, since H. Lin et al. 2012 showed that using both LIAM and LDAM can lead to better performance. The LIAM's dictionary has to have multiple pronunciations for different words that are shared between languages to make the system sensitive towards different pronunciations. Moreover, the dictionary should be comprised of different representations of phonemes, since replacing unknown phonemes in one language through known ones in another can help with cross-language information according to the findings by Bhuvanagirir and Kopparapu 2012. The voice input will then be forwarded to phone bigram recognizers. One of these will be language-independent and using the same dictionary as the LIAM. The other ones will be language-dependent and are supposed to parse the input regardless of the language they are trained for. However, I would have the recognizers assign confidence scores not only on the query level but on the token level as well. This way, a possible switch in languages can be accounted for and the transcription that is shown to the user will include possible code-switching. The language-independent recognizer will furthermore assign language labels to the tokens in order to make a comparison of confidence scores possible. Furthermore, I will use a strategy module as well, since it proved very valuable according to Gonzalez-Dominguez et al. 2014. The strategy module will not only collect the scores of the LIAM, LDAMs, and the phone recognizers but

also introduce possible confidence margins which are given by the user profile's information about most frequently used languages. These margins would be higher for languages which are not often used so that there is a bias towards recognizing frequently used languages. Additionally, there has to be a measure to avoid overfitting on one specific user profile in case the languages used by the user change or someone else is using the PDA. Essentially, the margins would only help to make the decision process faster in case of everyday use.

The mentioned strategy module would apply a variable cutoff rule as seen in the paper by Gonzalez-Dominguez et al. 2014, but would also take into account overall confidence scores by the AMs and the bigram recognizers and the token level scores provided by the recognizers to ensure that a transcription is still possible when code-switching takes place. After the cutoff has taken place (or not, depending on the return speed of the recognizers), the strategy module will take the highest scoring tokens to build the complete query transcription which will be presented to the user. The token selection would be based on the AM scores and the bigram recognizer scores. However, for every token used in the transcription which has the highest overall language score, the scores will be compared to the scores of the other submitted token confidence scores, the ones provided by the language-independent and the language-dependent recognizers. If a score by a different language-dependent recognizer is higher for a specific token and the language-independent recognizer has chosen the same score with a similar confidence score, the token in question will replace the token in the chosen transcription. Since the queries are normally quite short (as seen in Figure 19), this should not take too much additional time. After every time step, a partial transcription will be shown to update the user on the current progress before forwarding the transcription to the input parsing module. The proposed model is illustrated in Figure 20.

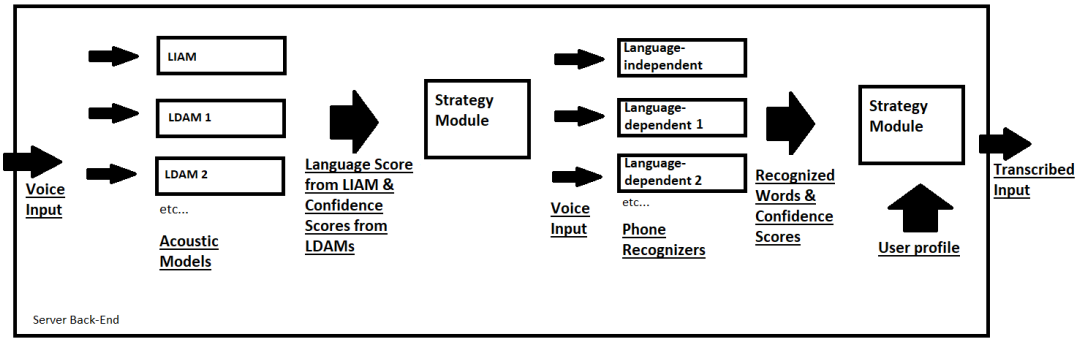

Figure 20: Proposed System Architecture for Speech Recognition Module.

The final transcription that the input parsing module receives will have a language label assigned to every token, or, in case of a monolingual query, the

entire query in order to facilitate further processing. Since a PDA not only has to be able to process voice inputs but also written inputs, the input parsing module itself also needs a way to identify languages from an input. This is the second step in building the model which I proposed for the multilingual PDA. This step will be skipped for vocal queries since the languages are already identified on a token or query level. Before language identification can take place, however, a possible obstacle has to be eliminated. Suppose a user in a foreign country wants to know how to get to a specific landmark. It is very likely that the name of the landmark is not in their language. This means that a named entity in the query for the PDA is of a different language than the rest of the query. If this named entity stayed part of the query, the language detection step would label it as a different language and it would be translated as part of the third step of input parsing: Translation. However, oftentimes named entities' translations are not literal translations (an example would be the movie "Thor - The Dark World" and the German title being "Thor - The Dark Kingdom" - the same language but different words). A translation of these named entities as part of the query would therefore change the meaning of the named entity and most likely lead to poor results. This is why the first step in my proposed model is Named Entity Recognition. In short, the transcribed vocal query or the written query first goes through a step of NER, followed by Language Detection for written queries, and, if there are still more than one language in the remaining query to be parsed, Translation. I will summarize the models I have presented for these steps and discuss the most interesting results.

The first two papers were focusing on NER. The first model was reported by Saito and Nagata 2003. The researchers used an HMM framework and divided their system into a language-dependent and a language-independent part. The system performed almost equally as well as an SVM, however, the processing speed was ten times faster than the SVM, which makes this approach very useful for my idea, since the three steps I proposed are essentially a pre-processing step before the general PDA algorithms continue processing. Additionally, the system can in principle be extended to any language as long as there is a properly tagged training corpus available. What is not necessary for my own approach is the conversion of input characters to Unicode and back into local code, since the processing and recognizing all take place on the same device, and the character encoding for the PDA can be pre-specified as Unicode from the start.

The second NER model was reported by DiPadova and Jiang 2018. The researchers used an LSTM framework. A vector space was used for word embedding which is a very interesting concept that makes it easy to handle many languages but also adding new languages since the new vectors can simply be added to the vector space to expand it. The system was tested for multi-class NER and binary NER. While the performance for multi-class NER was not optimal, performance in the binary NER was even better than for the first NER model. Moreover, the researchers indicated that performance could further improve upon additional training. Furthermore, their system was able to generalize semantic structures to detect named entities in languages it was not

even trained on and which were not part of the vector space dictionary. This, along with the vector space dictionary, is the most interesting finding for my own approach. The final version of the NER step will be discussed after the other papers have been summarized.

The next two papers were presented for the language detection step of the pre-processing module. The first model was reported by Rijhwani et al. 2017, and uses an HMM framework. In their model, every language has its own HMM with two states, one for assigning a language label to a token and one for assigning a universal label for emoticons, usernames, and more to a token. Switches between languages are made possible by cross HMM weights. The central finding from this paper is the structure of the HMMs and Meta-HMM which provides a fast, robust, and high accuracy performance.

The second model which focuses on language detection was reported by Chee Chang and C.-C. Lin 2014 and uses an RNN framework. The best performing model configuration was the Jordan type RNN with the n-gram vector and word2vector embeddings. This model shows very well the use of different metrics and representations to obtain a good performance accuracy. However, it seems that the entire system has to be retrained when a new language is added, which does not seem to be the case for Rijhwani et al. 2017.

The final step of my proposed pre-processing module is Translation. This step only becomes active if the query still has language labels from different languages even after excluding named entities. For this step, I have only presented one model since it was the only one I found at the point of writing this thesis that fits the task adequately. The model was reported by Fung, X. Liu, and Cheung 1999. It addresses the three identified tasks for query translation disambiguation defined by the researchers: translation candidate generation, candidate weighing, and candidate pruning. The model was tested with different strategies for weighting and pruning translation candidates, and the best strategy turned out to be 1-best contextual. It turned out that even at a very low primary language ratio the model achieved a very good performance, which is impressive since the model was proposed already in 1999. This simple decision system can be very useful for my own model since it provides a low-latency decision step. The most interesting result from this model is certainly its simplicity by using an online dictionary as well as the 1-best contextual word strategy.

I will now discuss the final version of my input parsing pre-processing module. As mentioned before, the module will consist of three steps: Named Entity Recognition, Language Detection, and Translation. Since this module only performs a preprocessing step, high performance speed is essential. This constraint makes the HMM framework for the NER step a more proper candidate since its performance was not significantly worse than the LSTM approach but the processing speed was very fast. In the case of my model, I do not need a conversion step since the character encoding can be controlled by the PDA immediately, meaning that transcriptions, as well as written queries, can be encoded in Unicode so that character conversion does not take additional time. Additionally, I believe it would be advantageous to use the unified named entity vector space that is used by DiPadova and Jiang 2018 and apply the labeling to a struc-

ture similar to the ones used by Rijhwani et al. 2017. In this case, a binary labeling as used by DiPadova and Jiang 2018 can be used. I believe that this might further increase accuracy and speed as seen in the paper by DiPadova and Jiang 2018. There will only be one HMM with two states which label a token each. One for "named entity" and one for "not named entity". Transition steps are possible between different states as well as a loop within the same state. Instead of using a lexical analyzer and character converter, there would be an embedding step before the labeling begins and the labels are put out at the end and applied to the tokens. This would also solve the problem of different space handling between languages since if a space is a token, it will be labeled as "not named entity" and if it is not a token it will simply be skipped. For handling out-of-vocabulary words, I would propose a combination technique of Saito and Nagata 2003 and DiPadova and Jiang 2018: substrings of an unknown token or unknown tokens can be used to look up possible named entities in the vector space, and the one that is closest to the substrings is chosen as the correct one, otherwise a new random vector is created. This combination of techniques should lead to a high precision but also very high processing speed due to the HMM framework. Additionally, adding languages can be quite easy since the word embeddings can simply be added to the vector space and retraining the system is achieved quickly. This step of NER is done for spoken queries as well as for written ones. After NER has been executed, the input is forwarded to the Language Detection step. This is only true for the written input, however, the vocal input will already have language labels on the token or query level. Written queries, on the other hand, still have to be analyzed.

For the Language Detection step, I propose a system like the one presented by Rijhwani et al. 2017. Due to its simplicity and high accuracy, I believe it might be the most suitable candidate out of the two presented models for this task. This system does not require any major changes. The way of modeling long-term dependencies by Chee Chang and C.-C. Lin 2014 through the use of an RNN framework is interesting, however, this is also achieved by the system proposed by Rijhwani et al. 2017. I will only remove the specifics of the model for handling data consisting of usernames and other tokens which do not occur in PDA queries. However, having universal tokens for named entities would be very useful. Since the tokens that are forwarded to the Language Detection system are already tagged accordingly, it would be very easy to assign a universal label to the tokens with the named entity label in a similar way as in the originally proposed model in order to model long-term dependencies. Moreover, as mentioned before, adding new languages would be very easy due to every language having its own HMM and only having to be trained in its own language. The only weights that have to be adjusted are the ones that make language switching possible. The Language Detection system will put out a series of language labels which are applied to the tokens before forwarding the query to the last step: Translation.

Before this step commences, the query has to be analyzed for language switches. This can be done quite easily by simply using the language labels and excluding the tokens with a named entity label. If there is still at least one

language switch detected, the Translation step commences. Since I only introduced one approach for this step, my own model will mostly use this approach as well. The only thing that is different compared to the model presented in the paper is that my system is trained on several languages. This means that a situation in which the primary language makes up less than 50% of the query will not happen because in this case the primary language will simply switch. In case of a query consisting of exactly 50% primary and secondary language, the user profile can be used to determine the most used languages of the user and decide based on this information. If there is no user profile available, the most frequent language will be chosen. For doing this, statistics like the ones from Techs 2021 can help. The decision will be based on the 1-best contextual word method since it showed the best performance. For generating translation candidates, since only monolingual information is needed at this point, it might be practical to use the sentences used for training the PDA modules since they are readily available and already in a format which is used in a PDA context. The proposed pre-processing module is illustrated in Figure 21. After unifying the languages, the input can be forwarded to the actual input parsing module where domain classification and slot and intent tagging can take place. This, however, is not part of my thesis anymore.

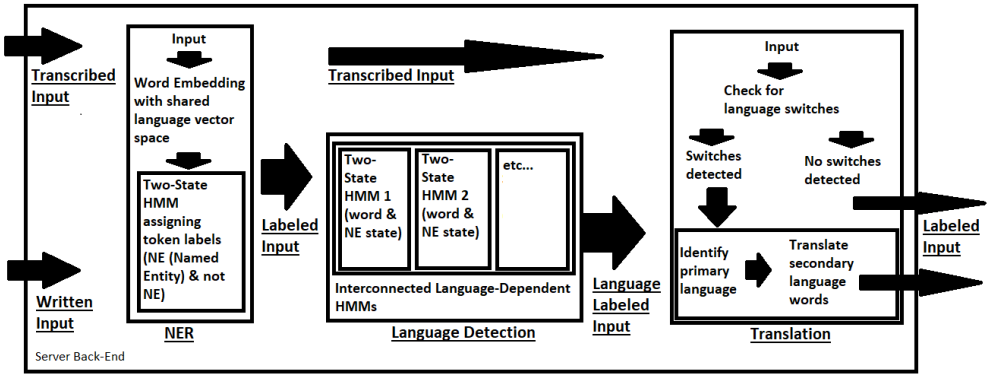

Figure 21: Proposed System Architecture for Pre-Processing Module.

My idea focuses on creating a multilingual input recognition module as well as an input pre-processing module for input parsing. After the described modules have finished their work, the PDA will use the same algorithms as standard PDAs which are in use nowadays and only work with one language at a time. Nevertheless, my ideas effectively render the PDA multilingual, since it is now able to work with any language as well as with code-switching as long as the language is supported. In detail: *1)* A vocal input will first be confidence scored

by an LIAM and the LDAMs of all the supported languages. *2)* The input stream for a specific time step will be forwarded to the language-dependent and language-independent phone recognizers which transcribe the input to the best of their knowledge and score the transcription on a token and query level for their confidence of correctness. *3)* The transcriptions, as well as confidence scores, will be collected by the strategy module which also applies possible language margins from the user profile to decide which partial transcription is the correct one to be displayed and in the end decides which final transcription will be used for further processing as well as if there are any language switches happening. *4)* Either the transcribed voice input or the written input will be forwarded to the NER step of the pre-processing module, where an HMM decides on a token level which token belongs to a named entity and which one does not and labels them accordingly. *5)* The labeled query, if it is a written one and not transcribed, will be forwarded to the Language Detection step of the pre-processing module where a series of HMMs label each token with the respective language and the named entities as a universal token. *6)* Before the last preprocessing step takes place the labeled query is analyzed for any remaining language switches, excluding the possible different languages of the named entities. *7)* If no language switches are detected the query is simply put forward to the parsing module, if any language switches are detected, the Translation step comes into action. The query will be analyzed for the primary language and the tokens of the secondary language translated into their primary language versions. If the language ratio is 50/50, the primary language will be either determined by the user profile or statistics for frequent online languages. *8)* After this step, the labeled and translated query will be sent to Input Parsing. This is where the general algorithms for domain classification and intent and slot tagging take over and the work of my proposed systems is finished.

5 Limitations of the Proposed System

The proposed system is able to turn any PDA into a multilingual PDA. However, the ideas I have introduced and discussed also have certain flaws or limitations which I will address in this section.

First, the system works on the basis of many different algorithms. Using the multi-pass approach for speech recognition bears the same limitations which are listed by Bhuvanagirir and Kopparapu 2012 (p. 93, 94), namely that the performance of the entire system depends on the performance of several smaller systems. In general, the performance of multilingual speech recognition, as can be seen from the results of the presented papers, is quite poor. Especially with an increasing number of supported languages and without knowing the query language a priori, performance is likely to drop with current speech recognition systems.

Second, another issue that is also addressed in the presented papers is the gathering of a suitable training corpus for a multilingual speech recognition system. Especially if it is supposed to be able to recognize code-switching and

if the system is based on a DNN framework, this can be a difficult problem to solve. My idea for using user data to keep training the system, which can also be observed when using, for example, the Google Assistant, is able to mitigate this problem partially. However, this only works if there are already a number of users present. The system also has to have its parameters trained before it can be offered to any users; and gathering this training corpus will require a lot of work by a lot of people, which brings me to my third limitation.

Training the PDA system on user data at the beginning of its release can potentially lead to strong training biases towards the (possibly) few input languages. The system learns that certain languages are not as important as other languages, which in turn leads to the PDA being a less attractive product to be used by the users of these underrepresented languages which starts the cycle anew. This problem can be somewhat mitigated by weighing the new training samples accordingly, however, I do not know yet how to deal with the possible scenario of several languages missing from the training corpus. Further research has to be done in order to address these issues.

6 Conclusion

In this thesis, I explored the idea of creating a theoretical framework for a multilingual PDA based on a literature review. I have defined the Input Recognition module and an Input Parsing pre-processing module as my central modules. To build them, I reviewed several papers with different approaches and combined ideas which I believe will lead to the most successful application. The most important ideas for Input Recognition, which I extracted from the papers, were that combining different metrics for processing leads to higher accuracy and that DNNs provide a robust framework for speech recognition. For the pre-processing module, it was key to combine low latency with high accuracy for which I chose HMM-based frameworks as well as a simple metric based on Mutual Information.

The final version of my proposed multilingual PDA thus consists of a DNN-based speech recognition module, which uses a combination multi-pass approach to identify and label incoming vocal queries on a token and query level and is controlled by a strategy module. Furthermore, my approach uses a pre-processing module that consists of an HMM-based NER step, an HMM-based Language Detection step for written queries, a simple control step for detecting language switches, and a Mutual Information-based Translation step. After the input has completed all these steps, it should only contain a single language, apart from possible named entities, and be easy to analyze by standard PDA algorithms for domain classification and slot and intent tagging. Since the system does not need prior information about the used language(s) but is still able to analyze the input regardless of whether it consists of a single or code-switched language, a PDA using this system is multilingual.

Additionally, I have identified limitations and problems my idea could face. These limitations are *a)* a language bias due to underrepresented languages, *b)*

the issue that overall performance depends on the individual performance of algorithms and language models, and *c)* the issue of collecting enough training data in the first place. However, I believe with advancing technology and new innovative ideas for language modeling these obstacles do not pose an unsolvable problem.

Investigating first combinations of the mentioned approaches can also help improving the proposed system. I would like to gain more insight into the effect of combining multilingual speech recognition systems and using a language-independent dictionary. Should I be able to optimize performance in this part, the system can be tested with more languages. My interest lies in combining European languages and Asian languages since they are not often researched together. Future work should focus on details about the systems used in the PDA. There are still many details to work out, for example the parameters for each neural network and HMM, how information is shared between them, and how the language models for the speech recognition module work in detail. Finally, the weighting of languages in order to adapt to the user is a topic which needs attention since wrong weighting can render the PDA useless for users or make speaker adaption impossible and stop the PDA from being as user-friendly as it could be.

However, should the proposed architecture prove to be useful and perform well, it can be released and used by users worldwide. Due to the easily extendable approaches I chose, it would also be easy to adapt the system to basically all common languages and even uncommon ones, provided enough training data can be collected. Furthermore, owing to the fact that the approach is mostly server-based and a user profile can be created anywhere as long as some storage space is available, the proposed system can be used in any device from mobile phones, to computers, cars, and even as tourist information counters in large cities. Due to the multilingual nature of the system, it can be used by anyone, anywhere, anytime and will ultimately lead to a more globally connected world.

References

Bacon-Shone, John and Kingsley Bolton (1998). "Charting multilingualism: Language censuses and language surveys in Hong Kong". In: *Language in Hong Kong at century's end*, pp. 43–90.

Bengio, Yoshua (2009). "Learning Deep Architectures for AI". In: *Foundations and Trends® in Machine Learning* 2.1, pp. 1–127. ISSN: 1935-8237. DOI: 10.1561/2200000006. URL: http://dx.doi.org/10.1561/2200000006.

Bhuvanagirir, Kiran and Sunil Kumar Kopparapu (2012). "Mixed language speech recognition without explicit identification of language". In: *American Journal of Signal Processing* 2.5, pp. 92–97.

Boies, Daniel et al. (Apr. 2016). *Techniques for updating a partial dialog state*. US Patent 9,318,109.

Bojanowski, Piotr et al. (2017). "Enriching word vectors with subword information". In: *Transactions of the Association for Computational Linguistics* 5, pp. 135–146.

Chee Chang, Joseph and Chu-Cheng Lin (2014). "Recurrent-Neural-Network for Language Detection on Twitter Code-Switching Corpus". In: *arXiv e-prints*, arXiv–1412.

Corredor-Ardoy, Cristobal et al. (1997). "Language identification with language-independent acoustic models". In: *Fifth European Conference on Speech Communication and Technology*.

Cruse, Holk (1996). *Neural networks as cybernetic systems*. Thieme Stuttgart.

DiPadova, Tony and Steven Jiang (2018). "Multilingual Named Entity Recognition". In:

Donakey, Andrea (2007). "Language planning and policy in Manchester". In: *Unpublished master's thesis, university of manchester. Manchester: United Kingdom*.

Fung, Pascale, Xiaohu Liu, and Chi-Shun Cheung (1999). "Mixed language query disambiguation". In: *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pp. 333–340.

Gonzalez-Dominguez, Javier et al. (2014). "A real-time end-to-end multilingual speech recognition architecture". In: *IEEE Journal of selected topics in signal processing* 9.4, pp. 749–759.

kmaclean (2010). *What is an Acoustic Model?* URL: http://www.voxforge.org/home/docs/faq/faq/what-is-an-acoustic-model. (accessed: 10th May 2021).

Knill, Katherine et al. (2014). "Language independent and unsupervised acoustic models for speech recognition and keyword spotting". In:

Kwok, Raymond (2019). *Baum-Welch algorithm for training a Hidden Markov Model — Part 2 of the HMM series*. URL: https://medium.com/analytics-vidhya/baum-welch-algorithm-for-training-a-hidden-markov-model-part-2-of-the-hmm-series-d0e393b4fb86. (accessed: 22nd June 2021).

Lin, Hui et al. (2012). "Recognition of multilingual speech in mobile applications". In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 4881–4884.

Liu, Weibo et al. (2017). "A survey of deep neural network architectures and their applications". In: *Neurocomputing* 234, pp. 11–26.

Miller, Scott et al. (Jan. 1998). "ALGORITHMS THAT LEARN TO EXTRACT INFORMATION BBN: DESCRIPTION OF THE SIFT SYSTEM AS USED FOR MUC7". In:

Nair, Harikesh, Pradeep Chintagunta, and Jean-Pierre Dubé (2004). "Empirical analysis of indirect network effects in the market for personal digital assistants". In: *Quantitative Marketing and Economics* 2.1, pp. 23–58.

O'Dea, S. (2021). *Number of smartphones sold to end users worldwide from 2007 to 2021*. URL: https://www.statista.com/statistics/263437/global-smartphone-sales-to-end-users-since-2007/. (accessed: 16th July 2021).

Olah, Christopher (2015). *Understanding LSTM Networks*. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. (accessed: 24th June 2021).

Rabiner, Lawrence and Biinghwang Juang (1986). "An introduction to hidden Markov models". In: *ieee assp magazine* 3.1, pp. 4–16.

Reynolds, Douglas A (2009). "Gaussian Mixture Models." In: *Encyclopedia of biometrics* 741, pp. 659–663.

Riehl, Claudia Maria (2019). "Code-Switching". In:

Rijhwani, Shruti et al. (2017). "Estimating code-switching on twitter with a novel generalized word-level language detection technique". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1971–1982.

Saito, Kuniko and Masaaki Nagata (2003). "Multi-language named-entity recognition system based on HMM". In: *Proceedings of the ACL 2003 workshop on Multilingual and mixed-language named entity recognition*, pp. 41–48.

Sarikaya, R. (2017). "The Technology Behind Personal Digital Assistants: An overview of the system architecture and key components". In: *IEEE Signal Processing Magazine* 34.1, pp. 67–81. DOI: 10.1109/MSP.2016.2617341.

Suthaharan, Shan (2016). "Support vector machine". In: *Machine learning models and algorithms for big data classification*. Springer, pp. 207–235.

Techs, W3 (2021). *Usage statistics of content languages for websites*. URL: https://w3techs.com/technologies/overview/content_language. (accessed: 9th July 2021).

Uriarte-Arcia, Abril Valeria, Itzamá López-Yáñez, and Cornelio Yáñez-Márquez (2014). "One-hot vector hybrid associative classifier for medical data classification". In: *PloS one* 9.4, e95715.

Vailshery, Lionel Sujay (2021). *Number of digital voice assistants in use worldwide from 2019 to 2024 (in billions)**. URL: https://www.statista.com/statistics/973815/worldwide-digital-voice-assistant-in-use/. (accessed: 16th July 2021).

Declaration of Authorship

I hereby certify that the work presented here is, to the best of my knowledge and belief, original and the result of my own investigations, except as acknowledged, and has not been submitted, either in part or whole, for a degree at this or any other university.

signature

Osnabrück, 29.08.2021

city, date