



A More Sensitive Edit-Distance for Measuring Pronunciation Distances and Detecting Loanwords

Master's Thesis, MSc Languages Communication Technology,

Faculty of Arts,

University of Groningen,

The Netherlands

Faculty of Information and Communication Technology,

University of Malta,

Malta

19th January 2018

Liqin Zhang

Email: l.zhang.13@student.rug.nl

Supervisor/university:

Prof. Dr. John Nerbonne

University of Groningen

Co-assessor/university:

Prof. Ray Fabri

University of Malta

TABLE OF CONTENTS

1. INTRODUCTION	1
1.1 Linguistic Background	3
1.2 Related Work	4
2. METHODOLOGY	6
2.1 Overall Design of the Experiment	6
2.2 Algorithms.....	8
2.3 Data	11
3. PMI-BASED LEVENSHTein ALGORITHM FOR MEASURING SOUND DISTANCE.....	12
3.1 Introduction to PMI-based Levenshtein Algorithm.....	13
3.2 Realizing PMI-based Levenshtein Algorithm	14
3.3 Results and Evaluation	20
3.3.1 <i>Determining the Threshold</i>	21
3.3.2 <i>Cross Validation</i>	24
3.4 Discussion	25
4. SPECTROGRAM-BASED LEVENSHTein ALGORITHM FOR MEASURING SOUND DISTANCE.....	26
4.1 Introduction to Spectrogram-based Levenshtein Algorithm	28
4.1.1 <i>Barkfilter, Cochleagram, and Formant Tracks</i>	28
4.1.2 <i>Measuring Segment Distances Acoustically</i>	30
4.1.3 <i>Applying Segments Distance to the Levenshtein Algorithm</i>	31
4.2 Realizing Spectrogram-based Levenshtein Distance.....	31
4.3 Result and Evaluation.....	34
4.4 Cross Validation.....	36
4.5 Discussion	36
5. SCA SOUND ALGORITHM FOR MEASURING SOUND DISTANCE.....	37
5.1 Introduction to the SCA Sound Distance Algorithm	37
5.1.1 <i>Alignment- introduction to PSA</i>	38
5.1.2 <i>Distance- ALINE Algorithm and its Modification</i>	40
5.2 Realizing SCA Sound Distance Algorithm	41
5.3 Result and Evaluation.....	43
5.3.1 <i>Determining Threshold</i>	44
5.3.2 <i>Cross Validation</i>	44
5.4 Discussion	45
6. COMPARISON OF THE THREE ALGORITHMS	45
6.1 Comparing Performances	45
6.1.1 <i>Thresholds and Performances</i>	45
6.1.2 <i>Determining Threshold Using Lower Outlier Boundary</i>	48
6.1.3 <i>Cross Validation</i>	49
6.2 Comparing Distribution of Distance Values	50
6.3 Comparing Results of Detected Loanwords	51
6.3.1 <i>Result Analysis</i>	51
7. CONCLUSION.....	54
BIBLIOGRAPHY.....	56

ABSTRACT

Loanwords exist in almost every language. Identifying loanwords manually is time-consuming. Previous studies present the possibility of detecting loanwords by comparing the similarity of pronunciation, or pronunciation distance, between words from two unrelated languages. The principal hypothesis is that, if a pair of words is made up of a loanword and its source in another language, the pronunciation distance between these two words should be significantly smaller than another pair that does not consist a loanword. The pronunciation distance is measurable by the edit distance. The Levenshtein algorithm is one of the main algorithms to calculate an edit distance. In order to accurately represent the pronunciation distance, a more sensitive edit distance for measuring pronunciation distance is desired. There are various refined Levenshtein algorithms which are implemented for sound-sensitive edit distance. The purpose of this dissertation is to apply three refined Levenshtein algorithms to calculate the pronunciation distance and discover a more advanced algorithm for loanword detection. The three refined Levenshtein algorithms calculate sound distances by respectively considering pointwise mutual information (PMI) between segments, measurements extracted from Spectrogram, and sound class alignment (SCA) between strings. The performances of each refined Levenshtein algorithm in loanword detection are compared. Their performances are evaluated by precision/recall analysis as well as cross validation. As a result, applying SCA-based algorithm outperforms the other two algorithms according to the evaluations.

1. INTRODUCTION

A loanword is a word that is borrowed from one language and adopted in another. To support the development of computational applications in linguistics, a more convenient and well-performing loanword detection model is desirable. Loanword detection has been studied using various methods, such as the phylogenetic method (Delz, 2013), ancestral state reconstruction (Köllner & Dellert, 2016), and the comparison of pronunciation distances (van der Ark et al., 2007; Mennecier et al., 2016). Distinguishing from previous research involving comparing pronunciation distances, this study aims to apply superior edit distance algorithms with more sensitive pronunciation distances for the purpose of loanword detection. This study aims at detecting loanwords within the Turkic and the Indo-Iranian language families with three refined edit distance algorithms, whose performances are compared with each other. The measurement of the sound distance between two words is decisive in detecting loanwords. The principal hypothesis is that the pronunciation distance between a loanword and the corresponding borrowed word should be significantly smaller than the distance between two words that do not have such relation (Van der Ark et al., 2007; Mi et al., 2014; Mennecier et al., 2016). Generally, if a word W_a in a language A shares the same concept represented as W_b in language B, either W_a or W_b is probably a loanword given:

1. W_a and W_b are phonologically similar;
2. Language A and language B are unrelated.

For Rule 1, “phonologically similar” means that the pronunciations of two words are similar. The similarity is described by a sound distance between two words. If the sound distance between two words is small, the two words are defined as “phonologically similar”. But how small should the sound distance be to define a loanword? The threshold for determining a loanword is decided by comparing the predicted loanwords against a gold standard in which loanwords are classified by experts. Rule 2 indicates that it is inappropriate to identify W_a and W_b as loanwords if language A and language B are closely related. Dutch and German, for instance, share numerous words with similar pronunciations. This is probably the result of genealogical relatedness since they are both Germanic languages and geographical neighbors. Hence, identifying loanwords merely by considering sound similarity is not sensible in this case.

The data used in this study was collected by Mennecier et al. (2016). The pronunciations of the words in the Turkic and Indo-Iranian language family representing concepts from the Swadesh list are included in the data. Swadesh list is a list containing approximately 200 concepts that are argued to exist in all languages (Swadesh, 1955; Swadesh, 1972). The pronunciations are written in phonetic transcriptions. For instance, the concept ‘one’ is written as /bɪr/ and /jak/, concept ‘big’ as /ʉlkin/ and /kalɔn/, in the Turkic and Indo-Iranian language families respectively. The data records numerous phonetic transcriptions representing a concept in both families because of the various languages in a language family, and potential variety of the pronunciations in a language. The loanword is detected by comparing phonetic transcriptions in the Turkic family against phonetic transcriptions in the Indo-Iranian family. According to the principal hypothesis, one of the phonetic transcriptions in a pair of transcriptions is a loanword if their pronunciations are similar enough.

The Levenshtein algorithm is successfully applied in calculating the differences between word pronunciations (Heeringa, 2004). The distance between two strings is called the Levenshtein distance if the Levenshtein algorithm is applied. The pronunciation distance between two words

is represented by the edit distance between the two phonetic transcriptions. The pitfall of applying the Levenshtein distance directly to represent pronunciation distance is that it fails to reflect the actual “dissimilarity” between two pronunciations. For instance, the sound distance between /dɔg/ and /dɒt/ is two, while between /dɔg/ and /dɪg/ is one if the Levenshtein distance is applied. But /dɔg/ is more similar to /dɒt/ than to /dɪg/ according to human perception. Hence, refined Levenshtein algorithms are desirable for the purpose of representing sound distance.

Although a simple Levenshtein distance can be used in loanword detection, the performance is not satisfactory because of the error rate (Van den Ark et al., 2007; Menecier et al., 2016). The edit distance of phonetic transcriptions between two words is decisive in loanword detection, and applying different (or refined) edit distance algorithms should lead to different results. In order to explore the possibility of improving the performance in loanword detection, three refined Levenshtein algorithms are evaluated in the application of loanwords detection. They are the pointwise-mutual-information-based algorithm (Wieling et al., 2009), the Spectrogram-based algorithm (Heerigna, 2004), and the sound-classes-based algorithm (List, 2012).

These algorithms have been applied in various areas, including dialectometry, accent measurement, and language relatedness (Wieling & Nerbonne, 2015; Wieling et al., 2014; Menecier et al., 2016). Hence, the purpose of this experiment is to compare the performances of above-mentioned algorithms in the application of loanword detection. It should be noticed that this method cannot be used to identify the donor language (the language that “gives” the word) because the distance between two pronunciations is not directed. Also, the method fails to determine if a third language is involved, since there is a possibility that a detected loanword is a cognate from a third language rather than one of these two. For instance, the English word ‘tofu’ is borrowed from Japanese, but the Japanese word originates from Chinese (Table 1).

	Word	Pronunciation in IPA
English	tofu	/ˈtəʊfuː/
Japanese	とうふ	/tɔːɸɯβ/
Chinese	豆腐	/tɒfu/

Table 1. The concept “tofu” written in English, Japanese, and Chinese, and their respect pronunciation.

It is possible to detect that one of the three words representing the concept “tofu” is a loanword when any two of these three languages are compared, but it is impossible to determine the donor language of this word and identify the origin of ‘tofu’ without considering the related facts about tofu.

The description of the linguistic background is below in this chapter. This is followed by a discussion of related works in loanword detection, as well as works in which refined Levenshtein algorithms are applied to calculate sound distances and other purposes. The second chapter, methodology, starts with the overall design of the experiment. The data used in this project is explained in the methodology section as well. The last part in the methodology chapter briefly introduces the algorithms used in this project, as well as the fundamental Levenshtein algorithm. The following three chapters explain the three refined algorithms in detail, along with their realization and performance in loanword detection. Results, evaluations, and simple discussions are included in these three chapters. Finally, there is a chapter for comparing the performances of the algorithms, and the conclusions of the project.

Before going further, here are several terms used in this dissertation. They are explained for the sake of clarity.

- Concepts: the concepts of word meanings in different languages. For instance, /mɔdar/ and /ɔna/ are pronunciations in Uzbek and Tajik, respectively, and they both mean ‘mother’. Here, ‘mother’ is the concept. The concepts are in English in the dataset used in this experiment.
- Segment: a phonetic unit such as /a/, /ɔ/, or /p/.
- Phonetic symbol: is a symbol belonging to the International Phonetic Alphabet (IPA). IPA symbols are used to represent the pronunciation of phonetic segments.
- Phonetic transcription: the representation of word pronunciation. A phonetic transcription is a sequence of phonetic symbols (/mɔdar/ is a phonetic transcription for instance).
- Segment distance: the distance between two phonetic symbols according to various features, such as phonological features, phonetic features, or acoustic features (intensity, frequency, etc.).

1.1 Linguistic Background

A loanword is a word borrowed from one language and adopted in another. Borrowing words is a common phenomenon in a language, and probably loanwords exist in every language in the world (Haspelmath & Tadmor, 2009). In distinction to similar concepts such as calque and code-switching, a loanword is the result of phonological adaptation rather than semantical adaptation (translation) or direct borrowing. A loanword is adapted phonologically in order to fit the phonological pattern in the recipient language (Peperkamp & Dupoux, 2003). Here is a sentence in Cantonese containing a loanword, a calque, and code-switching.

我	book	咗	巴士	去	跳蚤	市場	(Character)
Ngo5		zo2	baa1 si2	heoi3	tiu3 sat1	si5 coeng4	(Jyutping)
/ŋɔ:/	/bʊk/	/tso:/	/pa:si:/	/høy/	/tʰi:ʊsəd/	/si:tsʰ æ:ŋ/	(IPA)
I	book	(perfective)	bus	Go	flea	market	(English)
‘I have booked a bus to flea market.’							

‘巴士’ is a loanword from English ‘bus’. ‘跳蚤市場’ is a calque, which is the result of literal translation of ‘flea’ (‘跳蚤’) and ‘market’ (‘市場’). The verb ‘book’ is code-switch using English ‘book’ (as a verb) in the Cantonese sentence directly. The example of concept ‘bus’ in Cantonese shows that the loanword is similar to the original word phonologically without semantic adoption (/pa:si:/ in Cantonese vs. /bʌs/ in English), since neither the meaning of /pa:/ (巴) nor /si:/ (士) is related to bus¹. Hence, the phonological similarity is a feature of loanword, and the loanword is “highly predictable from a phonological perspective” (Paradis & LaCharité, 1997). This feature supports the principal hypothesis of the project mentioned above.

¹ An example of loanwords in Cantonese with both phonological and semantic adaptation is “啤酒”, meaning “beer” in English. In this case, “啤” (/pɛ:/) is the phonological adaptation of /bɪə/ (beer), and “酒” (/tsou/) means alcohol.

Necessity of loanword detection

Language contact is one of the main concerns in historical linguistics, and it reflects the interactions between language and ethnic groups. Loanwords arise as a consequence of language contact, and the study of loanwords contributes to the understanding of language history and word origin. Besides, the loanwords can influence a recipient language in various aspects, such as syntax, morphology, phonology and writing system. Maltese is a typical example reflecting how the study of loanwords is beneficial to the study of a language's development.

Maltese originated from Siculo-Arabic in the 11th century when settlers from Sicily moved to Malta. After expelling the Muslims in the 13th century, Maltese was separated from Arabic and it became an isolated language. It has developed along with Sicilian and Italian since then. From the colonization of Britain in the 19th century until now, Maltese had been influenced greatly by English. Consequently, over 50% of the vocabulary in Maltese are derived from either Italian, Sicilian, or English.

The many loanwords influence the language in various aspects. For instance, new sounds are introduced to Maltese. “televizjoni” is a loanword from English “television”, and /ʒ/ is introduced in Maltese phonology system through the borrowing of the word ‘television’. The writing system is influenced as well. Controversy has been raised arguing whether Maltese spelling, “televizjoni”, or English spelling, “television” should be used in Maltese. Meanwhile, loanwords influence the syntax of a language. The Maltese word of “to park” is “ipparkja” which is borrowed from English “park”. A new rule is designed to include English verbs in Maltese by adding prefix “ip-” and suffix “-ja”, which is different from the morphology rule applied to Maltese verbs. The syntactic adoption and the phonology adoption mentioned above lead to decreasing similarity between a loanword in the recipient language and the corresponding borrowed word in the donor language with time.

To a great extent, the development of Maltese involves borrowing words from other languages, and the loanwords have been an important component of Maltese. Therefore, Maltese speakers probably fail to distinguish loanwords from original Maltese words. On the other hand, it is controversial to classify a loanword if the word has been part of a language for a long period. Loanword detection is one of the most important steps to study the phenomenon and solve these problems.

Loanwords can be detected manually, but it is time-consuming and occasionally controversial. A more efficient method to detect loanword is desired. Hence, there is no doubt that applying a computational approach can be effective and efficient to detect loanwords.

1.2 Related Work

This dissertation involves mainly three aspects: loanword detection, sound distances derived from Levenshtein algorithms, and the application of sound distance derived from Levenshtein algorithm to the loanword detection. Previous works on the topics of loanword detection, applying (refined) Levenshtein algorithms to calculate sound distances, and applying sound distances to loanword detection are investigated here.

Previous work on applying sound distance to loanword detection

Van den Ark et al. (2007) calculate sound distances with a VC-sensitive Levenshtein algorithm in which the cost of substituting a vowel for a consonant (or vice versa) is higher than a vowel for a vowel (or a consonant for a consonant). Their work aims to classify groups of the languages in Central Asia and detect loanwords. In terms of loanword detection, the result shows that using Levenshtein distance for loanword detection is not perfect due to the tradeoff of precision and recall. Another work in which sound distance is applied is conducted by Menecier et al. (2016), in which pronunciations of central Asian languages are collected and applied to measure language relatedness and loanword detection. It applies the same method as that in van den Ark et al. (2007) to detect loanwords. Comparing to the work of van den Ark et al. (2007), the data used by Menecier et al. (2016) is more suitable for loanword detection (the dataset is also used in this dissertation and will be discussed in the later section). However, the results show that there is room for improvement. Around 30%-50% of the loans fail to be detected.

Previous works on loanword detection not applying sound distance

Various approaches are used to detect loanwords. One approach to detect loanword is to apply ancestral state reconstruction. State reconstruction is used in phylogenetics, and it aims to find the common ancestors of individuals. Köllner & Dellert (2016) apply this method to trace the cognates of words in order to discover the words sharing identical cognates so that loanwords are detected. The assumption is that, “if a cognate class is reconstructed for some node v , but a different class is reconstructed for its immediate ancestor”, “all leaves under v are possibly having undergone borrowing”. The algorithm is tested on the database of Indo-European Lexical Cognacy Database, and it is a database of cognate judgments in the Indo-European languages. The performance is not ideal due to several limitations, such as the size of the database, the failure of detecting borrowing outside the language sample, etc.

Besides above mentioned approaches, Mi et al., (2014) utilize the string similarity to represent pronunciation similarity to detect Chinese and Russian loanwords in Uyghur. The approach is based on the idea that a loan word shares similar pronunciation with the corresponding word from the donor language. This principle is applied by van den Ark et al. (2007) and Menecier et al. (2016). The difference is that Mi et al. (2014) use string similarity to represent the similarity of pronunciation. There are two major challenges to use string similarity to detect loanwords. One is the change of spelling, and the other one is suffixes of Uyghur words. Mi et al. (2014) apply characters alignment to solve the problem of the change of spelling, and classification-based models to solve the later. The application of these two methods enables Mi et al. (2014) to measure the string similarity between two words, and a loanword is identified on a given threshold. Mi et al. (2014) apply different string similarity algorithms to compare the performances of them on loanword detection. The method is evaluated by a corpus introduced by Mi et al. (2014). The corpus is trained from city names mapping table, and the test set includes materials from web. The result is considered as “efficiently” with the highest F1 score 73.18 for detecting Chinese loanwords, and 76.93 for detecting Russian loanwords.

Previous work on sound distance

Sound distances between words have been utilized in different linguistic topics for years, and the Levenshtein algorithm is the main algorithm to calculate sound distance. Moreover, refined algorithms have been introduced in order to provide more sensitive sound distance measurement. This method has been commonly used in the study of dialects, including relatedness and classification (Nerbonne et al., 1996; Nerbonne et al., 1999; Nerbonne & Kretzschmar, 2003; Heeringa, 2004). Van der Ark et al. (2007) apply the Levenshtein algorithm to obtain sound distances for language classifications instead of dialect classification. Also, a refined Levenshtein algorithm is applied to measure foreign accents strength (Wieling et al., 2014).

Conclusion

Loanword detection has been pursued by previous studies, but there are plenty of rooms for improvements upon the previous studies. Comparing the above mentioned works, the approaches used by Köllner & Dellert (2016) and Mi et al. (2014) to detect loanwords might not be as effective as applying sound distance. The work of Köllner & Dellert (2016) has shown that the performance of applying ancestral state reconstruction to detect loanword is not good. On one hand, the success of applying ancestral state reconstruction for loanword detection highly rely on the accuracy of the expert notations in the database. On the other hand, Köllner & Dellert (2016) states that purely considering cognate class is not sufficient to detect loanwords. Hence, considering phonological representations instead of cognate class is encourage by Köllner & Dellert (2016). In the case of Mi et al. (2014), using the phonetic transcriptions of the words might be more representative of the pronunciations of the words than strings. In other words, calculating the sound distance directly from the phonetic transcriptions is more accurate than string similarity used by Mi et al. (2014).

Van der Ark et al. (2007) and Menecier et al. (2016) apply Levenshtein algorithm to calculate the sound distance between words to detect loanwords. However, loanword detection is not the primary purpose of applying Levenshtein algorithms to measure sound distances in these two studies. Moreover, the performance of applying the original Levenshtein algorithm is not satisfactory, even when taking care to avoid aligning vowels and consonants. Since more refined algorithms have not been applied to loanword detection yet, the purpose of this dissertation is to explore the possible improvement of applying refined Levenshtein algorithms. Nevertheless, Menecier et al. (2016) state that the Levenshtein algorithm has no bias to words in one language group. Meanwhile, Menecier et al. (2016) suggest applying more sensitive measurements of sound distances in loanword detection.

2. METHODOLOGY

2.1 Overall Design of the Experiment

The data used in this experiment consists of phonetic transcriptions of words from various locations in Central Asia. The concepts underlying these words are from the Swadesh list, a list containing approximately 200 concepts that are argued to exist in all languages. The data are from

two language families, Turkic and Indo-Iranian. This experiment aims at comparing the performances of three edit distance algorithms in detecting loanwords between these two language families. There is more than one pronunciation per word per language family because there are various languages in a language family and various recordings of multiple informants at each data collection site. The pronunciation of every word of every informant in Turkic family is compared to every pronunciation in the Indo-Iranian family. The sound distance between two pronunciations from either language family is calculated by three different algorithms. For each algorithm, a list of tuples is generated containing the information of concept, the distance value, and a pair of phonetic transcriptions (Table 2). To generalize, the pairs of pronunciations of words for a concept, c , is represented as:

$$(t_1, i_1), (t_1, i_2), \dots, (t_1, i_{n-1}), (t_1, i_n), (t_2, i_1), \dots, (t_{m-1}, i_1), (t_{m-1}, i_2), \dots, (t_m, i_{n-1}), (t_m, i_n)$$

Given:

t : Phonetic transcription of the word for c in the Turkic family;

m : The number of phonetic transcriptions of the words for c in the Turkic family;

i : Phonetic transcription of the word for c in the Indo-Iranian family.

n : The number of phonetic transcriptions of the words for c in the Indo-Iranian family.

Hence, a generated tuple is represented as:

$$(c, \text{distance}(t_x, i_y), t_x, i_y) \quad x \in m, y \in n$$

A loanword is detected when the sound distance between two words is smaller than a given threshold. Finally, the results are compared to the gold standard. The gold standard is part of the dataset used in this dissertation. An expert marks the origin of the words with a code in the dataset. Within one concept, if a word from the Turkic family and a word from the Indo-Iranian family share the same code, it means that one of the words is loaned from the other. Precision/recall and F1 score are used to evaluate the performance of an algorithm used to detect loanwords. Three algorithms are applied to calculate the sound distance. Therefore, there are three groups of precision/recall and F1 score values. The performances of these three algorithms are evaluated by comparing their respective evaluation values. In order to evaluate the robustness of algorithms dealing with independent data, cross validation is applied.

```

('animal', '0.0273098', 'zanwar', 'hajvɔn')
('animal', '0.0277859', 'zanwar', 'hajβɔn')
('animal', '0.0228481', 'zanwar', 'hajwɔn')
('animal', '0.0312264', 'zanwar', 'zindene')
...
('animal', '0.00959365', 'hajwan', 'hajvɔn')
('animal', '0.0100698', 'hajwan', 'hajβɔn')
('animal', '0.005132', 'hajwan', 'hajwɔn')
('animal', '0.0309691', 'hajwan', 'zindene')
...
('back', '0.0362989', 'arqa', 'puʃt')
('back', '0.0346792', 'arqa', 'mijɔn')
('back', '0.0364408', 'arqa', 'pəʃt')
('back', '0.0342844', 'arqa', 'mijɔn')
...

```

Table 2. An example of results when applying the PMI-based algorithms to calculate edit distance.

2.2 Algorithms

The Levenshtein algorithm is the basis of the refined algorithms used in this dissertation. The Levenshtein distance is a popular algorithm to measure the edit distance between two strings as well as determine the alignment (Levenshtein, 1965). The edit distance between two strings is represented by the minimal operations to transform one string to another. There are three different operations: insertions, deletions, and substitutions in the simplest version of the algorithm. Each operation is assigned a cost of 1. The algorithm proceeds as follows. A matrix is created, with the source string being placed in the column vertically, and the target string placed in the row horizontally (Table 2). An extra symbol ‘#’ is added in the first place of each string. Denote s_1 as the source string with a length of m , s_2 as the target string with a length of n , i as an index in s_1 , j as an index in s_2 , and $d[i, j]$ as a cell in the matrix. $d[i, j]$ represents the string distance between substring of s_1 from 0 to the i th character, and sub-string of s_2 from 0 to the j th character. In the first row ($d[0, j]$, as j is 0 to n) and first column ($d[i, 0]$, as $i=0$ to m) of the matrix, fill in the integer from 0 to n , and 0 to m , respectively. For each value of $d[i, j]$ (as $i=1..m$, $j=1..n$), it is equal to the minimum value of the values adjacent, or diagonal to it (specifically $d[i-1, j]$, $d[i, j-1]$, and $d[i-1, j-1]$), plus the substitution cost. Substitution cost is 0 if $s_1[i]=s_2[j]$, otherwise it is 1. This procedure iterates until the matrix is completely filled. The value in the right bottom corner ($d[m, n]$) is the Levenshtein distance between these two strings. Tracing back to the path leading to the Levenshtein distance in $d[m, n]$ can extract the alignment of the strings as well as the operations used to obtain the result. The pseudo-code for realizing Levenshtein algorithm is presented in Code 1.

```

function LevenshteinDistance(string s1, string s2):
    #This function shows the transfer from s1 to s2
    m=len(s1)
    n=len(s2)
    #Define a table d[i,j] with a size of (m+1)X(n+1).
    for i from 0 to m:
        for j from 0 to n:
            #d[i,j]represents the distance between string s1[:i] and string s2[:j].
            declare int (d[i,j])

    for i from 0 to m:
        d[i,0]:=i
    for j from 0 to n:
        d[0,j]:=j

    for j from 1 to n:
        for i from 1 to m:
            if s[i]=s[j]:
                sub_cost:=0
            else:
                sub_cost:=1
            d[i,j]:=min(d[i-1,j]+1,           #deletion
                       d[i,j-1]+1,         #insertion
                       d[i-1,j-1]+sub_cost) #subsitution

    return d[m,n]

```

Code 1. Pseudo-code for Levenshtein algorithm.

The following example shows the application of Levenshtein distance on an IPA transcription (Table 3). In this example, it takes at least three operations to transform “/mɔdar/” to “/ɔna/” (deleting [m], replacing [d] by [n], and deleting [r], yielding the alignment shown as below). Hence, the edit distance between /mɔdar/ to /ɔna/ is 3. The algorithm guarantees that the minimal-cost alignment is found.

	#	ɔ	n	a
#	0	1	2	3
m	1	1	2	3
ɔ	2	1	2	3
d	3	2	2	3
a	4	3	3	2
r	5	4	4	3

m	ɔ	d	a	r
	ɔ	n	a	
1	0	1	0	1

Table 3. Applying the Levenshtein algorithm to transform phonetic transcript /mɔdar/ (“mother” in Tajik) to /ɔna/ (“mother” in Uzbek). The gray boxes in the table indicate the path of transformation from one string to the other.

Obviously, the Levenshtein distance fails to reflect the pronunciation distance between two words in fine detail. For example, the Levenshtein distance between /mɔdar/ and /mɔdar/ is the same as the Levenshtein distance between /mɔdar/ and /mɔdar/. However, the pronunciation of /mɔdar/ is more similar to /mɔdar/ than to /mɔdar/ in people’s perception. The cost of substituting /o/ for /ɔ/ should be smaller than the cost of substituting it for /u/. Hence, it is necessary to modify operation cost in Levenshtein algorithms so that a more sensitive edit distance can be applied to measure pronunciation distance. The refined algorithms used in this project change the cost of the

operations in Levenshtein algorithms by considering various aspects related to pronunciation measurement. For instance, Heeringa (2004) proposed a VC-sensitive² Levenshtein algorithm, which avoids the substitution between vowel and consonant. This is realized by increasing the substitution cost between vowel and consonant to 2, for instance (Code 2).

```
#This function is used to check if two characters are both vowels or
#consonants.
function check_vowel_consonant(character a, character b):
    if both a and b are vowels or both a and b are consonants:
        return True
    else:
        return False

function LevenshteinDistance(string s1, string s2):
    #This function shows the transfer from s1 to s2
    m=len(s1)
    n=len(s2)
    #Define a table d[i,j] with a size of (m+1)X(n+1).
    for i from 0 to m:
        for j from 0 to n:
            #d[i,j] represents the distance between string s1[:i] and string s2[:j].
            declare int (d[i,j])

    for i from 0 to m:
        d[i,0]:=i
    for j from 0 to n:
        d[0,j]:=j

    for j from 1 to n:
        for i from 1 to m:
            if s[i]=s[j]:
                sub_cost:=0
            else if check_vowel_consonant(s[i],s[j])==True:
                sub_cost:=1
            else:
                sub_cost:=2
            d[i,j]:=min(d[i-1,j]+1,           #deletion
                      d[i,j-1]+1,           #insertion
                      d[i-1,j-1]+sub_cost)  #substitution

    return d[m,n]
```

Code 2. Pseudo-code for VC-sensitive Levenshtein algorithm.

The VC-sensitive Levenshtein algorithm needs improvement. The example of /modar/, /modar/, and /modar/ shows that the substitution of vowels for vowels should be discriminated more finely. This is the reason a more sensitive sound distance algorithm that explores more features must be used. Heeringa (2004) utilizes spectrograms to reflect phonetic and acoustic features which are applied to measure segment distance, and other variants are motivated in similar ways.

² VC-sensitive means vowel and consonant sensitive.

The substitution cost in the Levenshtein algorithm is 1, but the refinements vary the substitution costs instead of always using one. Actually, insertion or deletion is considered a substitution between a segment and an empty segment. Basically, the substitution cost is decided according to the segment distance between two segments involved in the substitution. They are summarized as follow:

- PMI-based Levenshtein algorithm (Wieling et al., 2009): the segment distance is decided by pointwise mutual information (PMI) value between two segments. The PMI value depends on the strength of co-occurrence of two segments in a corpus of alignments.
- Spectrogram-based Levenshtein algorithm (Heeringa, 2004): the segment distance is decided by the acoustic features of those segments. Acoustic features of a sound are related to the intensity or loudness of various frequencies at a specific time, and these features are normally visualized in a spectrogram.
- Sound-Class-Based Levenshtein algorithm (List, 2012): Segment distance is decided by the “sound class” of segments. Sounds are grouped into different sound classes according to specific phonetic and phonology theories. Segment distances vary according to the substitution within or between sound classes.

2.3 Data

Menecier et al. (2016) conducted a survey to explore the language variety of the Central Asian region and utilized the data to measure the relatedness of languages and detect loanwords. The data is documented and it is publicly available for the study of loanwords detection in this experiment. The data was collected from 23 sites in three Central Asian countries, namely Uzbekistan, Kyrgyzstan, and Tajikistan. These sites are regarded as displaying “complex human and linguistic geography”. There are 88 informants and they are from the three countries. Generally, the informants are males who are over 40 years old, for genetic testing reasons. Genetic testing is a common strategy in collecting similar data. It allows tracing the linguistic history and genetic signal of an informant to determine the common genetic history, the similarity of culture (language), and the degree to which the two signals match. The mother tongues of these informants are: Kazakh, Kyrgyz, Karakalpak, Uzbek, Tajik, and Yaghnobi. These languages are from two language families, Turkic and Indo-Iranian. (Table 4). Besides their mother tongues, these informants also understood Russian well since they all went to school during the times of the Union of Soviet Socialist Republic (USSR).

A 200-word extended Swadesh list in Russian is shown to the informants and they are required to translate the words in the list orally into their mother tongues. Each word in the Swadesh list represents a concept. The pronunciations are digitally recorded and manually transformed into phonetic transcription. In total, each informant produces 200 pronunciations, resulting in more than 17000 recordings. There are approximately 88 phonetic transcriptions for a Russian word representing a concept. Sometimes, the number of pronunciations for a word is lower than 88 because the data is not valid. For example, some informants did not pronounce the word related to the concept properly.

Turkic	Indo-Iranian
Kazakh	Tajik
Kyrgyz	Yaghnobi
Karakalpak	
Uzbek	

Table 4. The languages in the dataset and their classification in language families.

The expert classification is marked in the dataset as well. Within a concept, each pronunciation is marked with a letter and the pronunciations with identical letters are considered cognates. Hence, a word from the Turkic family (or the Indo-Iranian) bearing the same letter as another word in the Indo-Iranian family (or the Turkic) means that one of the words is a loanword³. Notably, it is common that pronunciations of a concept in one language family are assigned to different cognate classes, because there are multiple languages in a language family, and informants may know multiple ways to translate a Russian word representing a concept. In this dissertation, within one concept, if a word from the Turkic family and a word from the Indo-Iranian family share the same cognate, it means that one of the words is loaned from the other language. This is the gold standard presented in the data.

The concepts in the Swadesh list include numbers, adjectives, nouns, verbs, and pronouns. It should be noted that the order of the concepts in the data is arranged in a certain pattern. For example, numbers come first in the list, and they are followed by commonly seen adjectives (such as “big” and “long”). Appendix I is the list of the concepts in the data used in this dissertation.

3. PMI-BASED LEVENSHTAIN ALGORITHM FOR MEASURING SOUND DISTANCE

In terms of measuring sound distances, the original Levenshtein algorithms in which each operation cost one is not entirely appropriate. The fact that it does not take specific features of sounds into consideration results in a failure to provide sensitive sound distances. For instance, a correct pairwise alignment of the phonetic transcriptions is one of the keys to generate sound distance, but it is not guaranteed by using the original Levenshtein algorithm. Considering the string /soʊfa/ and /ʃafa/, the pronunciations in English and Chinese, respectively for the word ‘sofa’. The Levenshtein algorithm leads to three different alignments:

s o u f a	s o u f a	s o u f a
ʃ a f a	ʃ a f a	ʃ a f a
1 1 1 0 0	1 1 1 0 0	1 1 1 0 0
A	B	C

All the edit distances of alignments are three if the operation cost is always one. However, it is obvious that alignment A is not appropriate since /o/ aligns with /ʃ/. A slight modification (VC-sensitive Levenshtein algorithm) is applied to forbid a vowel from being aligned to a consonant by raising the cost of substitution between vowel and consonant, so that situation A can be avoided (Heeringa, 2004). Unfortunately, this slight improvement fails to offer preference in case of

³ Normally, “cognate” is different from “loanword”. However, Menecier et al. (2016) indicate that “cognate” includes “borrowing” in this case because they use the word “cognate” loosely.

alignment B or C. Therefore, more features should be included to make a decision. In addition to this VC-sensitive Levenshtein algorithm, Wieling et al. (2009) introduce pointwise mutual information (PMI) as a segment weighting to generate edit distance between phonetic transcriptions. The core of the algorithm is that the substitution cost is decided by PMI-based segment distances rather than one. Further in this chapter include the introduction of this PMI-based Levenshtein algorithm, followed by its implementation. The application of PMI-based Levenshtein algorithm in the sound distance is explained in detail. Finally, the result and evaluation of the algorithm are presented.

3.1 Introduction to PMI-based Levenshtein Algorithm

PMI (Church & Hanks, 1990) is used to measure the strength of the tendency of two objects to co-occur. It is reflected by the probability of the co-occurrence of objects x and y ($p(x, y)$), and the independent occurrence of x ($p(x)$) and y ($p(y)$) respectively, assuming x and y are independent. PMI is defined as:

$$PMI(x, y) = \log_2 \left(\frac{p(x, y)}{p(x)p(y)} \right)$$

In the case of applying PMI to measure segment distance,

- $p(x, y)$: the probability of segment x and segment y that are aligned together. It is estimated by the number of times x and y that are aligned together divided by the total number of aligned segment pairs in the dataset.
- $p(x)$ or $p(y)$: the probability of segment x (or y) occurring. It is estimated by the number of times x (or y) occurs divided by the total number of segments in the dataset. The assumption is that x and y are independent.

A greater PMI value indicates more frequent co-occurrence of two segments. It is intuitive that a more frequent co-occurrence of two segments reflects smaller segment distance between them. In other words, a greater PMI value indicates a smaller segment distance as well as smaller edit cost. Normalization is used to scale the PMI value to represent segment distance. The PMI values of each segment pair are subtracted from the maximum PMI value so that the minimum segment distance is 0 and the values can represent segment distance intuitively (Table 5). When substituting one segment with the other, the substitution cost is the segment distance led by PMI value. The operation of insertion and deletion can be regarded as special cases of substitution, which is substituting an empty segment with a segment (insertion), or substituting a segment with an empty segment (deletion).

Applying PMI values and segment distances extracted from it can lead to a more accurate alignment and a more sensitive sound distance between two phonetic transcriptions. Wieling et al. (2009) introduce the procedure to obtain the alignments with PMI values. The first step is to create string alignments using the VC-sensitive Levenshtein algorithm. With these alignments, the PMI values of each segment pairs are calculated and normalized to segment distance using the method introduced above. The Levenshtein algorithm is applied again to generate new alignment pairs, using the new segment distances as substitution cost. Iteratively calculating PMI values and segment distances over the new alignment, and generating new alignments with new segment

distances can lead to better alignment coverage. This means that the procedure stops when alignments remain unchanged.

Alignments									
abcd	dabc	bdca	dbac	dbac	dbac	dabc	dbac	abcd	cabd
acbb	bcda	dcab	dcab	badb	dcab	dcab	acdb	dcab	dbca

(x,y)	Nr(x,y)	Nr(x)	Nr(y)	p(x,y)	p(x)	p(y)	pmi(x,y)	distance
(a,b)	3	20	22	0.075	0.25	0.275	0.125531	1.94847
(a,c)	5	20	19	0.125	0.25	0.2375	1.074001	1
(a,d)	4	20	19	0.1	0.25	0.2375	0.752072	1.321928
(b,c)	11	22	19	0.275	0.275	0.2375	2.074001	0
(b,d)	6	22	19	0.15	0.275	0.2375	1.199531	0.874469
(c,d)	1	19	19	0.025	0.2375	0.2375	-1.17393	3.247928

Table 5. Here is an example how PMI-base segment distances are calculated according to alignments. Assume a, b, c, and d, are segments and the alignments of sequences containing these segments are shown in the top table (alignment). The PMI values and related parameters that are needed to PMI values are presented in the table. In the column of PMI, the greatest value is 2.07 for segment pair (b, c). Hence, the PMI value of each pair is subtracted from 2.07, and the respective results are their distances. As a result, the smallest distance is always 0.

Consider again the example of “sofa” in English and Chinese, in order to determine a more suitable alignment between alignment B and C, it is possible to utilize a PMI-based segment distances list generated from a big data set. Initially, the substitution cost is always one in any circumstance. Assume there is a dataset containing a list of English and Chinese words in phonetic transcriptions. Applying the VC-sensitive algorithm can generate a list of word alignments from English and Chinese, respectively. After that, PMI-based segment distances are calculated according to these alignments. Applying the Levenshtein algorithm again with new segment distances generates new word alignments. This procedure is iterated until the alignments remained unchanged. Simultaneously a list of segment distances is generated as well. The segment distance between [a] and [u], and between [a] and [o] should be included. If the segment distance between [a] and [u] is smaller than [a] and [o], the substitution cost to use [a] to substitute [u] is smaller than to substitute [o]. In other words, alignment C should be the alignment of /soʊfa/ and /ʃafa/ because the total cost of operation is smaller than alignment B. Code 3 is the pseudo-code presenting the algorithm which results in the most appropriate alignment between two strings.

3.2 Realizing PMI-based Levenshtein Algorithm

This algorithm is implemented by utilizing the RuG/L04 tool developed by Peter Kleiweg from the Groningen group⁴. RuG/L04 is designed for dialectometrics and cartography, and it has been applied to derive pronunciation distances between sounds with the original Levenshtein algorithm (Van Der Ark et al., 2007), as well as with the PMI-based Levenshtein Algorithm (Wieling et al., 2014) in the previous studies. The procedure to utilize RuG/L04 to realize the PMI-based

⁴ <http://www.let.rug.nl/~kleiweg/L04/>

Levenshtein algorithm is based on the work of Wieling et al. (2014), in which they apply the algorithm to measure foreign accents strength in English. Since RuG/L04 allows defining the operation cost, the operation cost is defined by PMI values to realize the PMI-based Levenshtein algorithm.

```
dataset=[string a, string b, string c, ...]
segment_list=[]

#Assign the strings in the dataset to a list called segment_list.
for string in dataset:
    for segment in string:
        segment_list.append(segment)
    end
end

#For all possible pair combinations of segments from segment_list, initialize
#the segment distance between segments in a pair as 1.
segment_pairs_table:= all possible segment pairs
for segment_pair in segment_pairs_table:
    segment_distance_list[segment_pair]=1

string_pair_table := all possible combination in dataset
#alignment(X,Y) function is to generate alignment using Levenshtein
#algorithm.
#Argument X is a list of string pairs. Argument Y is a segment distance list
#indicating the substitution cost.
alignment_list=alignment(string_pair_table,segment_distance_list)

#PMI_based_segment_distance(X) function is to calculate the segment distance
#using PMI values.
#The argument X is a list of aligned strings.
segment_distance_list=PMI_based_segment_distance(alignment_list)

do:
    new_alignment_list=alignment(string_pair_list,segment_distance_list)
    segment_distance_list=PMI_based_segment_distance(new_alignment_list)
until:
    new_alignment_list doesn't change anymore.
```

Code 3. PMI-based Levenshtein algorithm for generating alignment between two strings.

In this dissertation, the two language families, Turkic and Indo-Iranian, play the role of “locations” in RuG/L04. The various phonetic transcriptions under each concept in the dataset are considered different “dialects” in each “location”. The purpose of this experiment is to identify loanwords at the word level. A detected loanword is one of the phonetic transcriptions in a pair of phonetic transcriptions. In other words, one of the words in the detected pairs is loaned. Hence, the “locations” should contain only one pronunciation so that the distance is calculated by a pair of phonetic transcriptions rather than a pair of “phonetic transcription lists”.

Defining initial operation cost is required beforehand in RuG/L04. The initial operation cost satisfies the requirement of VC-sensitive algorithms. All operations cost one, except the case of substitution between vowel and consonant (the cost is higher than one). According to Wieling et

al.'s (2009) method, a VC-sensitive Levenshtein algorithm is applied to the whole dataset so that a list of PMI-based segment distances is outputted. In the case of substitution, the PMI value is a measurement of co-occurrence of two segments. Insertion and deletion are considered as a special case of substitution, that one of the segment is an empty string. The newly generated PMI-based segment distances are used as the operations costs to calculate the pronunciation difference of each phonetic transcription pairs between Turkic and Indo-Iranian. Eventually, a list of pronunciation distances is generated.

In this pair, a collection of phonetic transcriptions of words from Central Asian languages in the Swadesh list is used. It is worth noticing that the diacritics in the phonetic transcriptions have been ignored for the sake of effectiveness (Wieling et al., 2014). It means there is no difference among [z̤], [ẓ] and [z]. Following the above procedure, there are two sections to calculate the distance between phonetic transcriptions: generating PMI-based segment distances, and calculating distances. There are four steps to generate PMI-based segment distances:

1. Transforming the original data file to the appropriate format. As mentioned above, a list of PMI-based segment distances should be generated with the whole dataset. The data arrangement is designed as Table 6. The first column is the name of the two language families, and the first row is the 183 concepts in English (only three concepts are shown in Table 6). The phonetic transcripts are filled in the cells and they are split by '/'. Missing data exists and it is filled with empty space. For example, there is missing data in the cell of concept 'one' in Turkic. Hence, two consecutive "/" are found in the cell because of missing data between "/".
2. Generating a csv file for each concept. Based on the data format in Table 6, a csv file is generated for each concept by a provided python script in RuG/L04. A csv file of a concept contains the phonetic transcripts for that concept in both language families (Table 7 left).
3. Tokenizing data and generating initial segment distances. Each phonetic symbol in the data files is transformed to be represented by an integer (Table 7 right). Meanwhile, initial segment distances are calculated according to feature values considering all phonetic transcriptions in the dataset.
4. Generating the PMI-based segment distance. In order to generate PMI-based segment distance, an initial distance value, which is generated in the last step, should be provided. The program "leven" not only outputs the distance between two "locations" but also updates segment distance table by outputting a file containing the PMI-based segment distances. Meanwhile, "leven" allows setting a parameter of small fraction number. This number is added to the frequency avoiding frequency of zero (1e-80 is used as shown in the tool manual). The file containing PMI-based segment distances is used for calculating the distance between phonetic transcriptions.

In summary, RuG/L04 requires data in the format of Table 6. Each concept in Table 6 has its own file as shown in Table 7. RuG/L04 compares the sound between the Turkic and the Indo-Iranian family per concept. In Table 6, there is more than one pronunciation under a family. Applying "leven" to those concepts actually outputs the distance between two groups of pronunciations. In other words, the distance is calculated in the concept level (later in this chapter). In order to calculate the distance between two pronunciations, the data needs to be rearranged so that there is only one pronunciation under a family per concept.

	‘one’	‘two’	‘three’	...
Turkic	bir / brıw / bırw / / brıw / brıw / bir / bər / bər / bər / bər / bir / bir / bir / bır / bır / bir / bir / bir / bir / bır / bər / bır / bır / bir / bər / bər / bər / / bər / bər / bər / bər / bər / bər / bər / bər / bır / ber / bər / / bır / bər / bər / bər / bır / bır / / bır / bır / bır / bər /	jekə / jekə / jıkjew / jıkə / jeka / jekıew / jıkə / jıkə / jıkə / jıke / jıkə / jeki / jeki / jeki / jekı / jeki / jeki / jeki / jeki / jeki / jıkıe / ekkı / ekı / ekkı / jeke / jıkə / jıkə / jıkə / / jekke / jıka / jıkıe / jekə / jıkə / jıkə / jıkə / jıkə / jıkə / jıkə / jıkə / ikkıe / ikkıe / ikkıe / / ikkıe / ikkıe / ikta / ikkıe / ikkı / ikki / ikkıe /	uʃ / uʃ / uʃjɔ : / uʃ / uʃ / ʃu : / uʃ / yʃ / yʃ / øʃ / uʃ / yʃ / yʃ / yʃ / yʃ / yʃ / yʃ / / yʃ / yʃ / yʃ / uʃ / yʃ / yʃ / / yʃ / yʃ / uʃ / uʃ / uʃ / yʃ / / uʃ / yʃ / uʃ / uʃ / uʃ / uʃ / uʃ / yʃ / uʃ / uʃ / uʃ / yʃ / yʃ / yʃ / uʃ / uʃta / uʃ / uʃ / / uʃ / uʃ /	...
Indo-Iranian	jak / jak / jak / jakta / jak / jak / jak / jak / jak / jak / jak / / jakta / jakte / jak / jak / jak / / jak / jakta / jak / jak / jak / jak / jak / jak / jakta / jak / jakta / jak / jak / jak / jakto / jak / jakta / jak / i : / i : / i : / i : / i : /	du / du / du / dutta / do / dɔ / / do / dɔ / dɔ / dɔ / du / dutɔ / / dutɔ / do / du / do / dɔ / dutta / du / dɔ / du / dɔ / dɔ / du / dytta / dɔ : / dɔtta / dɔ : / du / du / dɔtto / dy : / dutte / du / do : / dɔ : / do / du / do : /	sɛ / se / se / setta / sɛ / se / / sɛ / se / sɛ / sɛ / se / sɛte / / sɛte / se / se / se / se / sɛ / sɛta / se / se / se / se / sɛ / / se / sɛta / se : / se :ta / sɛ : / se / se / seta / se / sɛta / sɛ / saraj / traj / tɛraj / traj / / tɛraj /	...

Table 6. The first few columns of the arrangement of data used by RuG/L04.

“one”	
%utf8	%utf8
: turkic	: turkic
- bir	+ 36 34 5
- brıw	+ 36 5 34 4
- bırw	+ 36 34 5 34 4
-	
- brıw	+ 36 5 34 4
- brıw	+ 36 5 34 4
- bır	+ 36 34 5
- bər	+ 36 39 5
...	...
: iranian	: iranian
- jak	+ 7 2 23
- jak	+ 7 2 23
- jak	+ 7 2 23
- jakta	+ 7 2 23 32 2
- jak	+ 7 2 23
- jak	+ 7 2 23
...	...

Table 7. Every concept has its own file. This is an example of concept ‘one’. Phonetic transcripts for concept ‘one’ (left), and its transformation in integers (right). Notice that there is missing data on the left, and it is ignored in integer format.

With the generated PMI-based segment distances, the distance between phonetic transcriptions can be calculated. Another data arrangement is designed as Table 8. Importantly, the duplicate phonetic transcriptions in a language family per concept need to be deleted to avoid repetition results. The first row is the title showing the concepts and indexes. The title is indicated as index-concept-title here. For instance, the title ‘0_0_one’ represents a phonetic transcription pair, /bɪr/ and /jakta/, the first phonetic transcriptions in the Turkic family of concept ‘one’, and the first phonetic transcriptions in the Indo-Iranian family of concept ‘one’ (the index starting from 0). In total, there are more than 25 thousand unique phonetic transcription pairs. Based on the data format presented in Table 8, a group of csv files are generated by the python script as mentioned above. The content of the file is a pair of phonetic transcriptions corresponding to the index-concept title (Table 9, the first column of each title). Each unique phonetic symbol in the data files is transformed to be represented by a unique integer (Table 9, the second column of each title)⁵. The distances of those 25+ thousand unique phonetic transcription pairs are calculated with the “integer data” and the PMI-based segment distances.

	‘0_0_one’	‘0_1_one’	...	‘1_0_one’	‘1_1_one’	...
Turkic	bɪr	bɪr	...	bɪrw	bɪrw	...
Indo-Iranian	jakta	jaktə	...	jakta	jaktə	...

Table 8. For concept ‘one’, the arrangement of data used by RuG/L04.

“0_0_one”		“0_1_one”		...
%utf8	%utf8	%utf8	%utf8	
: indo_iranian	: indo_iranian	: indo_iranian	: indo_iranian	
- bɪr	+ 36 34 5	- bɪr	+ 36 34 5	
: turkic	: turkic	: turkic	: turkic	
- jak	+ 7 2 23	- jakta	+ 7 2 23 32 2	

Table 9. The first column of “0_0_one” and “0_1_one” is the content of the csv file “0_0_one” and “0_1_one” respectively. They are the phonetic transcriptions pairs corresponding to their respective index-concept title. The second column is the content of the files containing integers corresponding to phonetic segments.

The “leven” program is applied to each phonetic transcription pair inputting the PMI-based segment distance and integer data. There are more than 25 thousand files containing one pair of phonetic transcriptions in each of them. As a result, more than 25 thousand pronunciation distances are generated (Code 4). For the purpose of future evaluation, the distance values, concepts, and phonetic transcription pairs are arranged in tuples as shown in Table 1, such as:

(‘one’, ‘0.0355222’, ‘bɪr’, ‘jak’)

Besides considering the pronunciation distance of a pair of single phonetic transcriptions, it is also worthwhile to explore the loanwords on the concept level by considering pronunciation distance between language families per concept. “On the concept level” means to discover the concept that is “borrowed” from a language. For instance, most of the words representing the concept “person” are loanwords (either from Turkic family to Indo-Iranian family, or opposite). Hence, the concept

⁵ In practice, the integer data can be extracted based on the integer data in Table 7 by deleting the duplicated integers and rearranging them into the format of Table 9.

“person” is considered as a “borrowed concept” in a language. As mentioned previously, the various phonetic transcriptions per concept in the dataset are treated as different “dialects” in each “locations” as regards the software. Therefore, each concept is possibly represented by a pronunciation distance between “dialects” in Turkic and “dialects” in Indo-Iranian family. In other word, there is more than one phonetic transcription per language family per concept, which means there are multiple variants within one language family per concept. Hence, the phonetic transcriptions in a family are grouped together per concept, and each concept can be associated with a (mean) distance value as a result. Similarly, if the distance value of a concept is small enough, the word representing the concept in the Turkic family is probably borrowed from the Indo-Iranian family, or vice versa.

It should be noticed that, when there is a list of phonetic transcriptions representing a concept in a language, the distances of a concept between two language families is related to the distances of natural pairs and the number of phonetic transcriptions representing the concept from each family. A natural pair is a pair of transcripts formed by one transcript from each list, and it has the smallest sound distance among all possible pairs. In RuG/L04, the distance between two language families per concept is equal to the minimal non-zero distance of natural pair divided by the number of natural pairs⁶ (Nerbonne & Kleiweg, 2003).

⁶ More details are introduced in Nerbonne & Kleiweg (2003).

```

#Use the python script in RuG/L04 to transform the dataset(as in Table 5).
#one,two...are the concept titles.
one,two,...:=python.py(dataset_concept)

#The integer format of the data are generated, as well as the initial segment
distance.
one.integer, two.integer,...,initial_distance:=
tokenize([one.data, two.data, three.data,...], configuration_file):

#Output the pmi-based distance.
leven_distance_between_families, pmi_distance:=
leven(data=[one.integer, two.integer,...], distance=initial_distance)

#The index_concept_title are generated as in Table 8.
#Every index_concept_title has unique phonetic transcription pair (in inte-
ger) per concept.

0_0_one.integer, 0_1_one.integer,...,0_0_two.integer,0_1_two.integer,...:=
    deleting duplicated phonetic transcriptions and rearranging[one.inte-
ger,two.integer,...]

#distance_list stores all the sound distances of each lexicon meaning.
distance_list=[]
for file in [0_0_one.integer, 0_1_one.integer,0_2_one.integer,...]:
    file.dis:=
        leven(data=file,distance=pmi_distance,output_segment_distance=False)
    distance_list.append(file.dis)

```

Code 4. The procedure of realizing PMI-based Levenshtein algorithm using RuG/L04, and calculating distance between phonetic transcription pairs.

3.3 Results and Evaluation

As a result of calculating the sound distance between words from Turkic and Indo-Iranian family, each phonetic transcription pair has a file containing the PMI-based sound distance value between the two language families. The distances are extracted and handled by a python script. Before determining loanwords according to these distances values, the statistical description and related facts of the distances values are explored (Table 10).

Min	0.00000
1st Qu.	0.02909
Median	0.03325
Mean	0.03113
3rd Qu.	0.03559
Max.	0.04420

Table 10. Statistical description of the sound distances derived by the PMI-based algorithm.

Figure 1 is the histogram of sound distances derived from all the word pairs between Turkic and Indo-Iranian generated by all concepts. The figure shows that the distribution of sound distances is skewed to the right, which means that most distance values are relatively large. According to

our assumption of determining loanwords using sound distances, loanwords are the minority on the left side of the histogram. This corresponds to the fact that the loanwords are the minority according to expert classification. In this experiment, two approaches are applied to determine the threshold which is the boundary of loanwords or not. One is determining the threshold by examining several hundred potential thresholds. The other one is to calculate the outlier and use it as the threshold.

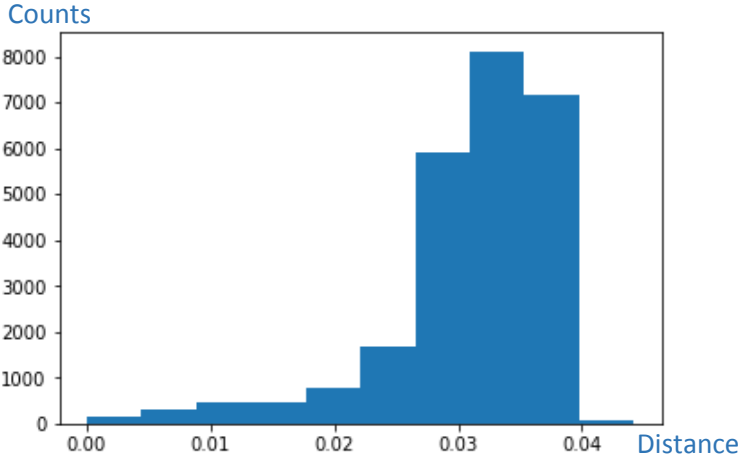


Figure 1. Histogram of sound distances derived by PMI-based algorithm.

3.3.1 Determining the Threshold

Determining a threshold to detect loanwords means that the word represented by one of the phonetic transcriptions in a pair is classified as a loanword if the pair has a sound distance value smaller than a selected threshold. The predicted loanwords are compared to the gold standard and they are evaluated by precision/recall as well as the F1 score (Manning et al., 2008). Precision is the percentage of true positive, which means the percentage of correctly detected loanwords among the words detected as loanwords. It reflects how “precise” the detection is. The recall is the percentage of loanwords that are correctly detected. It reflects the ability to extract the relevant items (loanwords). F1 score considers both precision and recall to measure the quality of the detection.

a = No. of correctly detected loanwords (true positives)

b = No. of incorrectly detected loanwords (false positives)

c = No. of all the words detected as loanwords (true positives + false positives)

d = No. of loanwords (true positives + false negative)

$$\text{Precision} = \frac{a}{c}, (c = a + b)$$

$$\text{Recall} = \frac{a}{d}$$

$$\text{F1score} = \frac{2 \cdot (\text{precision} \cdot \text{recall})}{\text{precision} + \text{recall}}$$

Inspired by Mennecier et al. (2016) and Van der Ark et al. (2007), a group of evenly spaced values over an interval between the minimum value (other than 0) and the average value of the distance list are chosen as potential thresholds in order to spot a suitable threshold. Since the histogram skews to the right, it is a reasonable assumption that using the mean value (or above mean) as threshold leads to a low possibility to achieve the best performance. Theoretically, using a larger number to divide the potential threshold interval ensures that the most suitable threshold is obtained because of its coverage of more threshold values. However, it raises the cost of computation simultaneously. In this case, there are 200 potential thresholds chosen from the interval between 0.0037 (the minimum value other than 0) and 0.0311 (average value). Figure 2 is the plot of precision, recall, and F1 score against 200 potential threshold values.

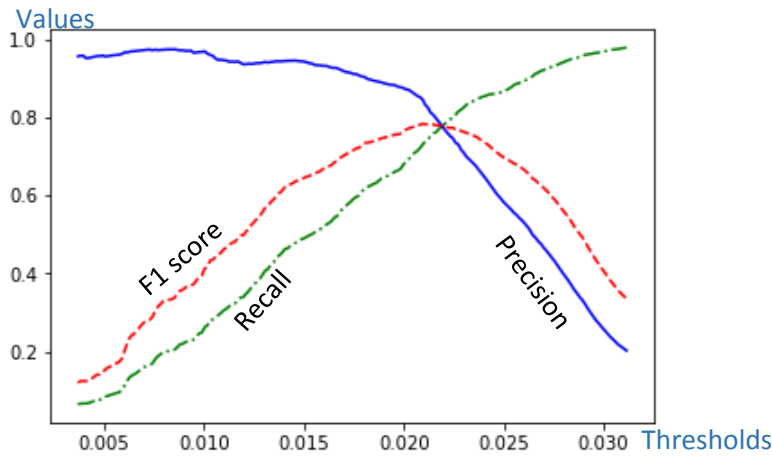


Figure 2. Precision, recall, and F1 score against the 200 potential thresholds.

Precision is over 0.974 at the small threshold value, while recall is very low. In fact, only around 6% loanwords from the gold standard are detected at the low threshold. Precision slightly drops with the rising of the threshold, while the recall value climbs dramatically. The absolute slope value of recall is obviously greater than precision. At a threshold value of around 0.02, Precision suddenly drops significantly, and the recall keeps climbing steadily. After the intersection between precision and recall, precision keeps dropping and recall almost reaches 1.0 at a threshold of over 0.03. As for the value of the F1 score, it climbs in the same way as recall before intersection of the lines and drops in the same way as precision after the intersection.

Figure 2 shows that a tradeoff exists between precision and recall: the higher the recall, the smaller the precision. Meanwhile, the threshold value is positively correlated to recall because a higher threshold value has a higher tolerance and allows more words to be classified as loanwords. It is difficult to determine the performance of loanword detection by solely applying precision or recall. Hence, the F1 score is used to reflect the performance. F1 score peaks at 0.784 at the threshold 0.021. The precision and recall are 0.846 and 0.730, respectively. It is summarized as:

Best threshold	0.021
F1 score	0.784
Precision	0.846
Recall	0.730

Table 11. Summary of the performance using the PMI-based Levenshtein algorithm to detect loanwords in the data including all the concepts.

Meanwhile, the loanwords at the concept level are detected as well (as mentioned at the end of Section 3.2). The detected concepts which are considered “loanwords” are reflected at the word levels. Each phonetic transcription pair is represented by a concept in English. Among the phonetic transcription pairs predicted as loanwords, the concepts presenting those phonetic transcriptions are analyzed. The distribution of the number of concepts existing in the predicted loanwords set shows that the concepts appearing frequently are the concepts predicted as being represented by loanwords.

Which loanwords fail to be found and which words are incorrectly classified as loanwords? Table 12 presents several randomly-chosen examples of these two situations. In general, the reason for failing to detect the loanwords is that the pronunciations of the relevant words are significantly different. The first part of the table shows that the phonetic transcriptions of the words are not similar, which leads to a large sound distance. This shows that the phonological adaptation of loanwords possibly results in a huge pronunciation difference. Meanwhile, the idiosyncratic pronunciation of the informants contributing to the data might affect the result as well. For instance, the words for the concept “flower” are generally loanwords, and most of the pronunciations are /gul/, /gyl/, or /guł/. However, one of the informants whose mother tongue is Tajik says /kul/, which is not detected. There are also words which are incorrectly classified as loanwords. The second part of Table 12 presents words with similar pronunciations which are not loanwords. Actually, the sound distances of these phonetic transcription pairs are quite close to the threshold (0.021), and this is probably the reason of misclassification (for instance, the distance between /bajlau/ and /bastak/ is 0.0205; the distance between /pustlɔq/ and /pust/ is 0.0167).

Loanwords that not detected	
Concept	Phonetic transcription pair (Turkic, Indo-Iranian)
‘person’	/adam/ (Karakalpak), /ɔdamzɔt/ (Tajik)
‘correct’	/tɔrɔs/ (Karakalpak), /durust/ (Tajik)
‘flower’	/gul/, (Kazakh), /kul/, (Tajik)
‘breast’	/kukrik/, (Kazakh), /quqrak/, (Tajik)
‘to dig’	/tʃaβlamɔq/, (Uzbek), /kɔftan/, (Tajik)
Words incorrectly detected as loanwords	
Concept	Phonetic transcription pair (Turkic, Indo-Iranian)
‘bark’	/pustlɔq/ (Kyrgyz), /pust/ (Tajik)
‘bone’	/sujak/ (Uzbek), /suyun/ (Tajik)
‘worm’	/kurt/ (Kyrgyz), /kirm/ (Tajik)
‘to tie’	/bajlau/ (Karakalpak), /bastak/ (Tajik)
‘narrow’	/taɾ/, (kyrgyz), /tank/ (Yaghnobi)

Table 12. Several examples that loanwords are not detected and words are incorrectly detected as loanwords using the PMI-based algorithm.

3.3.2 Cross Validation

In order to evaluate the performance of the model when it deals with independent data, a cross validation is conducted. Cross validation is a commonly used model validation technique. Assume a k-fold cross validation is conducted. Firstly, the dataset is split equally into k pieces. One of the pieces is used as a testing set, and the other k-1 pieces are used as a training set. After the model is trained by the training set, the testing set is applied to the model to evaluate its performance. The procedure is iterated for k times, and a different piece is used as the testing set each time. Eventually, there are k evaluation results (precision, recall, or F1 score).

In this experiment, a 10-fold cross validation is conducted. The dataset is shuffled before being split because the concept is arranged in a specific order. For instance, the first five concepts in the data are numbers; at the end of the data, the concepts are mainly pronounced and interrogatives.

After the dataset is split into a training set and a testing set, the phonetic transcription pairs in the training set are used to find the threshold outputting the highest F1 score using the method introduced in Section 3.4. For the sake of computational cost, the “best threshold” is found by investigating 10 potential thresholds rather than 200. The “best threshold” is then used to decide the loanwords in the testing set. Comparing the predicted loanwords against the expert-classified loanwords in the testing set returns evaluation results including ten precision values, ten recall values, and ten F1 score values. Finally, the mean values of the ten precisions, the ten recalls, and the ten F1 scores are the results that reflect the performance of the model. Figure 3 is a diagram illustrating this process. The result of the 10-fold cross validation is shown in Table 13.

Precision	0.760
Recall	0.795
F1 score	0.768

Table 13. Mean scores in 10-fold cross validation for the PMI-based Levenshtein algorithm.

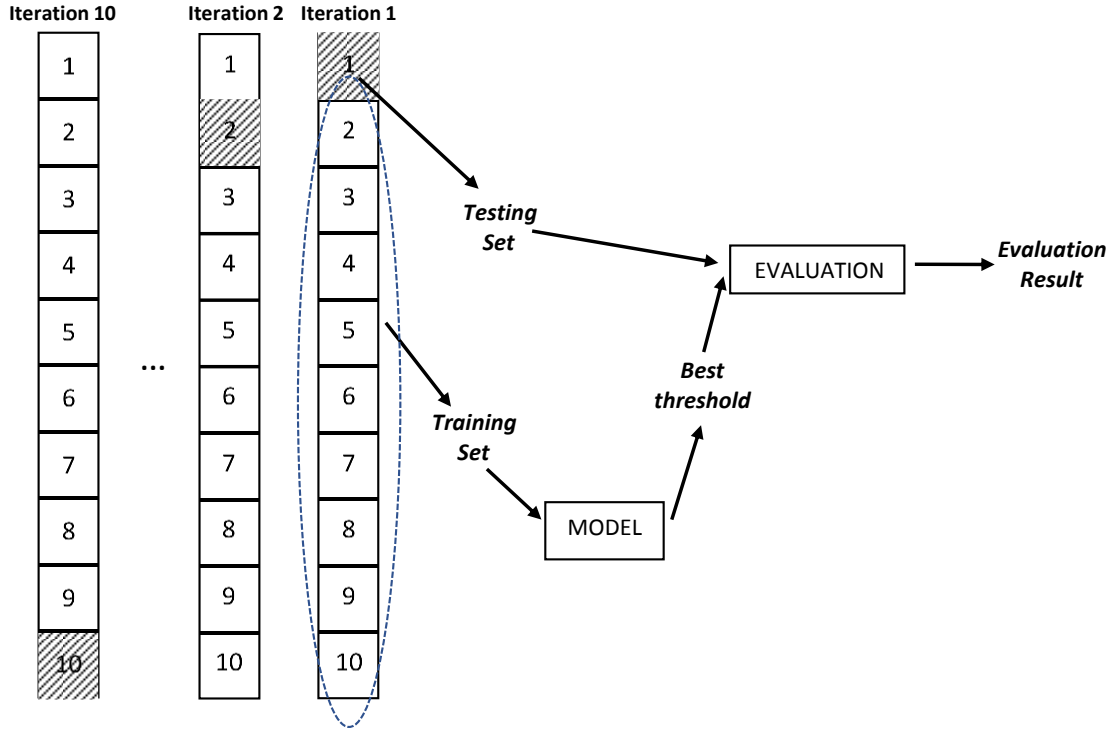


Figure 3. Diagram illustrating the process of conducting cross validation.

3.4 Discussion

Generally, choosing an appropriate threshold for this task is not that intuitive. Van der Ark et al. (2007) detect loanwords at the word level with a similar method and dataset but applying a simple Levenshtein distance rather than a PMI-based sound distance. Van der Ark et al. (2007) suggests that the determination of the threshold relies on the tolerance of “noise”. Since this experiment aims to compare the performances of various sound distance calculation algorithms in detecting loanwords, the F1 score is an appropriate indicator of algorithm performance because of its consideration of precision and recall simultaneously. The approach applied in this experiment to determine the threshold is computationally expensive. An alternative is to detect the outliers in the set of distance values. Then the threshold is simply equal to first quartile minus 1.5 times the difference between the first quartile and the third quartile (the lower outlier boundary). Although the threshold leading to the highest F1 score value is not guaranteed, it reduces the cost of computation.

4. SPECTROGRAM-BASED LEVENSHTAIN ALGORITHM FOR MEASURING SOUND DISTANCE

Using the original Levenshtein algorithm fails to reflect the sound features of phonetic transcriptions. Heeringa (2004) introduces a so-called VC-sensitive sound distance which raises the penalty of substituting a vowel with a consonant, and vice versa. However, it is necessary to consider more features for the sake of determining segment distances between vowels and between consonants, as well (See the “sofa” example in Chapter 3). Hence, Heeringa (2004) takes advantage of the spectrogram to investigate the acoustic representation of vowels and consonants so that a more sensitive sound distance scheme between phonetic transcriptions can be realized. Speech sounds have their specific acoustic characteristics (Reetz & Jongman, 2011), and the spectrogram is an ideal visualization of the acoustic features of sound samples.

Similar to the PMI-based algorithm (Wieling et al., 2009), Heeringa (2004) determines the segment distances between two phonetic symbols based on their acoustic features and applies the segment distances to the Levenshtein algorithm. Three sound representations based on a spectrogram are introduced by Heeringa (2004) in order to measure segment distances acoustically. Instead of a common spectrogram, two variants of the spectrogram are applied as acoustic representations of sounds. The first variant is Barkfilter, in which Bark-scale is used in the y-axis of a spectrogram rather than linear Hertz scale as in a normal spectrogram. Bark-scale has merit in matching the human perception of sound frequency. The second variant is cochleagram, a modification of Barkfilter. Bark-scale is used in cochleagram as well, but the loudness of each frequency is given instead of intensity. Besides these two variants of the spectrogram, formant track is used to represent acoustic features of sounds by giving formants values of each time step as the representation of sounds (Figure 4). In this part of the experiment, these three spectrogram-based representations of sounds are implemented to measure the segment distances and apply those segment distances in Levenshtein algorithms for loanwords detection. The sound distances calculated by these three representations of sounds are called spectrogram-based sound distances. Further in this chapter, the mechanism of measuring spectrogram-based segment distance is introduced. Furthermore, a tool for implementing the spectrogram-based Levenshtein algorithm is explained by presenting the procedure of applying the tool to calculate the sound distances between the Turkic and the Indo-Iranian family per concept. Finally, the result of the predicted loanwords based on the spectrogram-based sound distances is presented and analyzed so that the performance in loanword detection is evaluated.

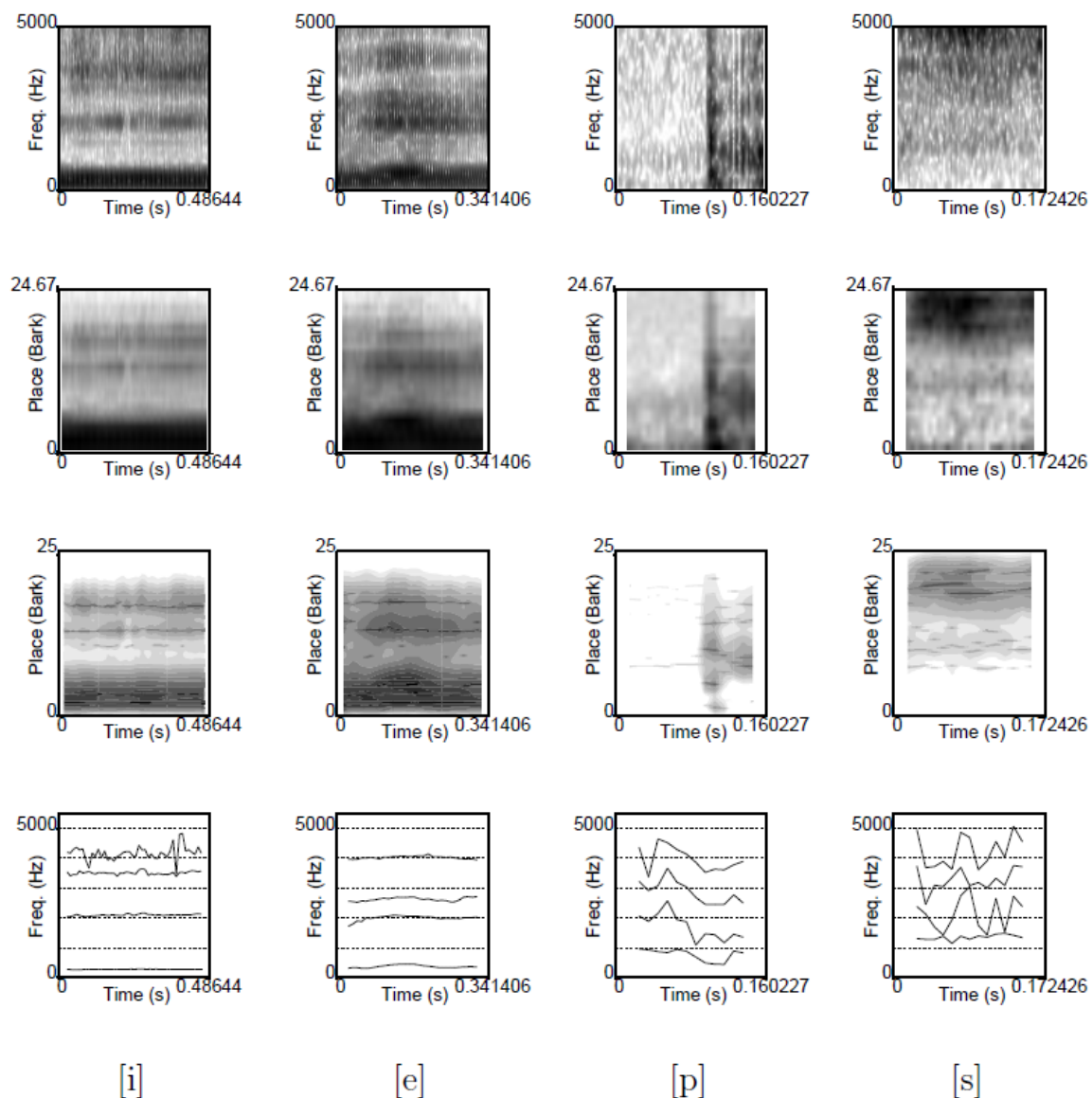


Figure 4. Different acoustic representation of sounds pronounced by John Wells (Figure Heeringa, 2004). There are four different sounds represented in the (from top to bottom) spectrogram, Barkfilters, cochleagrams, and sound tracks respectively. The x-axis of the graphs is time. The y-axis of the graphs is frequency in Hertz (in case of Spectrogram and formant tracks) or Bark (in case of Barkfilter and cochleagram).

4.1 Introduction to Spectrogram-based Levenshtein Algorithm

A spectrogram is a three-dimensional visual representation of a sound sample (Figure 4 Row 1). The x-axis represents time, and the y-axis represents frequency in Hertz. The darkness of the colors reflects the intensity of frequency at a specific time. One of the applications of a spectrogram is to identify sounds phonetically since different sounds have different acoustic features reflected in a spectrogram. For instance, the distance between formants in the lowest band (F1) and the second lowest band (F2) varies in different sounds (Figure 5, see section 4.1.1 about formants), which means formant values are representative of some sounds. Hence, the acoustic features of sounds can be utilized to calculate the segment distances and these segments distances are applied in the Levenshtein algorithm. As mentioned above, Heeringa (2004) introduces two modifications of spectrograms, Barkfilter and cochleagram, to measure segment distances.

4.1.1 Barkfilter, Cochleagram, and Formant Tracks

The common spectrogram represents frequencies in linear Hertz scale. However, the human perception of frequency is not linear, but logarithmic. Therefore, Heeringa (2004) uses the Barkfilter as sound representation (Figure 4, Row 2). In Barkfilter, the Bark-scale is used instead of a linear scale, because the Bark-scale is closer to human perception of frequency. Heeringa (2004) uses the software Praat⁷ to obtain intensity values in Bark-scale, and the intensity values are used to calculate the sound distances. In Praat, Schroeder et al.'s (1979) formula is used for the purpose of transforming Hertz values into Bark-scale values:

$$Bark = 7 \times \ln \left(\frac{Hertz}{650} + \sqrt{1 + \left(\frac{Hertz}{650} \right)^2} \right)$$

Figure 6 is a plot of Bark value against linear Hertz frequency. The plot shows that the Bark scale is linear below 1000Hz, and becomes approximately logarithmic over 1000Hz. This matches the way a human would perceive the frequency of sounds.

Besides Barkfilters, Heeringa (2004) uses cochleagrams to represent sounds (Figure 4 Row 3). A cochleagram is similar to the Barkfilter in that they both use the Bark-scale, but in a cochleagram it returns the loudness of frequencies in time instead of intensity. Loudness is what people actually perceive, and it is related to intensity as well as frequency. Similar to the Barkfilter, Heeringa (2004) uses Praat to obtain the loudness values of sounds. In Praat, the same formula is used to calculate the Bark-scale in cochleagrams.

⁷ Praat is a software for phonetic analysis of speech. More information: <http://www.fon.hum.uva.nl/praat>.

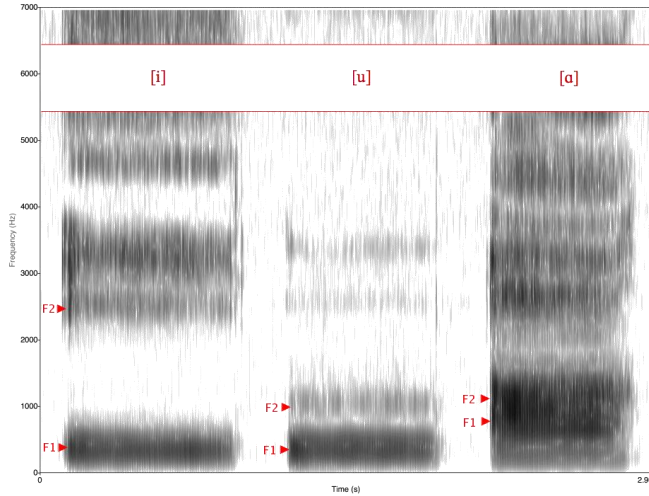


Figure 5. This figure shows the F1 and F2 of three different vowels [i], [u], and [a]. The distance between F1 and F2 of [i] is greater than [u], and [u] is greater than [a] (image created by Wikipedia user ish ishwar in 2005).

Formant track is the third representation of sounds used by Heeringa (2004). In a spectrogram, the use of small analysis windows makes individual harmonic blends and bands appear. The middle frequency of a band at a point of time is a formant, and a formant track is formed by connecting formants at a range of continuous time (Figure 4 Row 4). A formant in the lowest band is called F1, a formant in the second lowest band is called F2, etc. (Figure 5). Heeringa (2004) uses the formants from F1 and F2 to represents sounds since different sounds have their own specific F1 and F2 (Rietveld and Van Heuven, 1997). The Bark-scale is used in this case as well, but the transformation is conducted by the formula of Traunmüller (1990), which is:

$$Bark = \frac{26.81 \times Hertz}{1960 + Hertz} - 0.53$$

The formula is recommended for the phonetic project. The plot of this formula is shown in Figure 6.

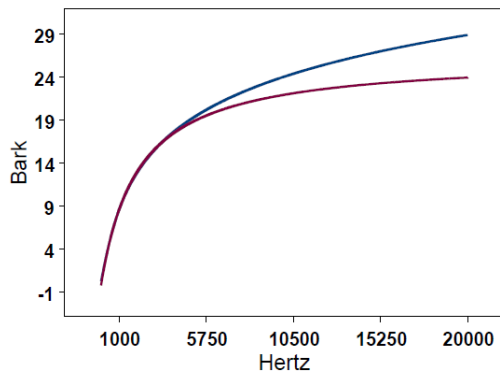


Figure 6. Plot of Schroeder et al. (1979) formula (upper line) and Traunmüller (1990) formula (lower line) (Heeringa, 2004).

4.1.2. Measuring Segment Distances Acoustically

In order to measure segment distances acoustically for all sounds in the IPA alphabet, sound samples are required. A tape called The Sounds of the International Phonetic Alphabet on which all sounds in IPA alphabet are produced by John Wells and Jill House is available and Heeringa (2004) uses it to extract acoustic data for all the sounds. Spectrogram-based segment distances are measured by considering the spectrogram or formant track representation of sounds. Related values, including intensity, loudness, and formants, can be extracted for every sound in the IPA alphabet applying the above mentioned methods. It is notable that the sound sample used by Heeringa (2004) is canonical, and the sounds may be different from the sounds in the Central Asian data collection.

The spectrogram-based representations of sounds are related to the duration of pronouncing a sound. It is important to normalize the duration of sound to ensure the sounds are comparable. Hence, the sound is analyzed in a unit of the time step. A spectrogram divided by units of the time step is called a spectrum, and a formant track divided by units of the time step is called a formant bundle. Here a spectrum or a formant bundle is called a slice. Within a slice, there are numerous intensity, loudness, or formants values depending on the Barkfilter, cochleagram, or formant tracks that are applied. Assume there are two segments, s_1 and s_2 , and they have m and n slices, respectively. The collections of slices are:

$$s_1 = \{s_{11}, s_{12}, \dots, s_{1m}\}$$

$$s_2 = \{s_{21}, s_{22}, \dots, s_{2n}\}$$

In order to normalize the length of duration, s_1 is duplicated n times and s_2 is duplicated m times. As a result, there are $m \times n$ slices in each segment.

$$s_1 = \{s_{111}, s_{112}, \dots, s_{11n}, s_{121}, s_{122}, \dots, s_{12n}, \dots, s_{1m1}, s_{1m2}, \dots, s_{1mn}\}$$

$$s_2 = \{s_{211}, s_{212}, \dots, s_{21m}, s_{221}, s_{222}, \dots, s_{22m}, \dots, s_{2n1}, s_{2n2}, \dots, s_{2nm}\}$$

Eventually, $m \times n$ pairs of slices are generated. For each pair of corresponding slices, the Euclidean distance between them is calculated. Assume there are v values in each slice belonging to a pair (s_{1ij}, s_{2kl}) , the distance between two slices within a pair is:

$$d(s_{1ij}, s_{2kl}) = \sqrt{\sum_{a=1}^v (s_{1ij}^a - s_{2kl}^a)^2} \quad (i \in [1, m], j \in [1, n], k \in [1, n], l \in [1, m])$$

The distance between two segments s_1, s_2 , is equal to the sum of the Euclidean distances between each pair of corresponding slices, divided by the number of slices $M \times N$.

4.1.3 Applying Segments Distance to the Levenshtein Algorithm

The purpose of introducing spectrogram-based segment distances is to provide more sensitive segment distances leading to a more precise representation of sound distance between two phonetic transcriptions. The spectrogram-based segment distance is applied to the operation weights in the Levenshtein algorithm. There are three operations to transform one transcript to another in the Levenshtein algorithm, and all three operations cost one in the default algorithm. Similar to the PMI-based Levenshtein algorithm, determining a suitable alignment between two phonetic transcriptions is important to decide the minimum edit distance between them, and using more sensitive segment distances in operation weights is beneficial in the determination. Considering again the example of ‘sofa’ in English and Chinese, which the potential alignments repeated below:

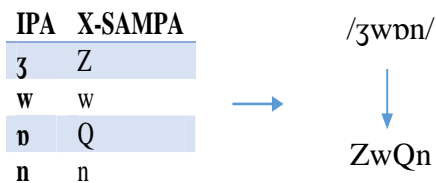
s o u f a	s o u f a	s o u f a
ɣ a f a	ɣ a f a	ɣ a f a
1 1 1 0 0	1 1 1 0 0	1 1 1 0 0
A	B	C

The VC-sensitive algorithm avoids alignment A, which matches a vowel to a consonant, but fails to choose between B and C. Applying spectrogram-based segment distances can determine which is the more suitable alignment between B and C, given that the segment distance between [o] and [a] is different from the distance between [u] and [a]. As long as a suitable alignment is found, the sound distance between two phonetic transcriptions is determined by summing the operation cost. Deletions and insertions are regarded as a segment pair between a normal sound and a silent sound. A silent sound indicates that the values within a spectrum or formant bundle are zero.

4.2 Realizing Spectrogram-based Levenshtein Distance

Heeringa (2004) develops a tool that calculates the spectrogram-based segment distances. The tool is used for measuring pronunciation differences between two dialects, as well as producing the visualization of the result. Utilizing this tool returns the sound distances between a pair of phonetic transcriptions from the Turkic and the Indo-Iranian families with spectrogram-based segment distances. The Turkic family and the Indo-Iranian family are regarded as the “dialects” in the tool analogously. Rather than outputting the sound distances between any two “dialects” (in fact, they are the Turkic and Indo-Iranian families in this case), the tool is modified for the purpose of returning the sound distances of the phonetic transcription pairs. The input phonetic transcription data of the tool is required to be in the form of Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA). X-SAMPA is an extended version of SAMPA, and it is used to transform all IPA phonetic symbols into 7-bit ASCII⁸. Here is an example of transforming /ʒwɒn/ (which means “thick” in Kazakh from the Turkic family) into X-SAMPA:

⁸ See <http://www.phon.ucl.ac.uk/home/sampa/ipasam-x.pdf> for more details about X-SAMPA.



The input of the tool is two files, in which the phonetic transcriptions are stored and compared against each other. Table 14 is the data format used in the tool for outputting spectrogram-based sound distance. Phonetic transcripts per concept per family are separated by a “/” and concepts in a family are separated by a new line tag “\n”. Two data files are generated for Turkic and Indo-Iranian family respectively using Python. The tool requires these two files as input, and the distances are calculated by comparing the phonetic transcriptions from each file in the same line (e.g. the phonetic transcriptions in Line One in Turkic file compares to the ones in Line Two in the Indo-Iranian file). If there are five lines (in other words, five concepts in this case), for instance, of phonetic transcriptions in both files, five distance values (between pronunciations of the same concept) are calculated.

Concept	Turkic family	Indo-Iranian family
‘one’	1 bIr / 1 brIw / 1 bIrIw / 1 brIw / ...	1 jak / 1 jak / 1 jak / 1 jakta / 1 jak / ...
‘two’	1 jek@\ / 1 jek@\ / 1 jIkjew / 1 jIk@\ / ...	1 s_hE / 1 se / 1 se / 1 setta / 1 s_he / ...
‘three’	1 }s_h / 1 }s_h / 1 }s_hju: / 1 }s / 1 }s / ...	1 tSahOr / 1 tSOOr / 1 tSOOr / 1 tSOorta / ...
‘four’	1 t2rt / 1 t2rt / 1 tw2rt / 1 t_h2rt / 1 t2rt / ...	1 pandZ / 1 panZ / 1 panZ / 1 panZta / ...
‘five’	1 b_hIs / 1 b_hIs / 1 b_hes / 1 b_hIs / ...	1 kalOn / 1 kalOn / 1 kalOn / 1 kalOn / ...
...

Table 14. This table shows part of the pronunciations in X-SAMPA format of five concepts (‘one’, ‘two’, ‘three’, ‘four’, and ‘five’) from the Turkic family and Indo-Iranian family. Each concept is represented in one line, and the pronunciations are separated by “/”. The number “1” in front of each pronunciation functions as a label and it has no influence on the result. The 183 concepts are listed line by line in each file.

As shown in Table 14, there is more than one pronunciation in every line in either file. Applying the tool to this data format generates distances on the concept level (later in this chapter). In order to generate distances on the word level, there should be only one pronunciation in every line. Hence, the data should be rearranged in another format. The duplicated data in a language family per concept is deleted to avoid repeated phonetic transcription pairs. A file containing the phonetic transcriptions in the Turkic family and a file containing the phonetic transcriptions in the Indo-Iranian family are created for each concept. Hence, concept ‘one’, for instance, has two files called ‘one_turkic’ and ‘one_indoiranian’ respectively, and the content of each file is:

'one_turkic'	'one_indoiranian'
1 bIr	1 jak
1 brIw	1 jakta
1 bIrIw	1 jakt@\
1 b@\r	1 jakt6
1 bir	1 jaktQ
1 bIS	1 i:
1 biS	

Table 15. The content of file 'one_turkic' and 'one_indoiranian' for the Spectrogram-based algorithm.

In order to ensure that every phonetic transcription in a family is compared to every phonetic transcription in the other family, the phonetic transcriptions are arranged in a certain way. Assume there are m unique phonetic transcriptions denoted as t_x ($x = 1, 2, \dots, m$) in the Turkic family, and n unique phonetic transcriptions denoted as i_y ($y = 1, 2, \dots, n$) in the Indo-Iranian family for the given concept. Each phonetic transcription in the Turkic family is duplicated n times one by one, while all the phonetic transcriptions in the Indo-Iranian family are duplicated together for m times. In general, the data per concept is arranged as in Table 16. Table 17 is an example showing the data arrangement of concept 'one'.

Turkic	Indo-Iranian	Turkic file after duplication	Indo-Iranian file after duplication	Phonetic transcription pairs
t_1 t_2 t_3 \vdots t_x \vdots t_m	i_1 i_2 i_3 \vdots i_x \vdots i_n	t_1 t_1 t_1 \vdots t_1 t_2 t_2 t_2 \vdots t_2 \vdots t_m t_m t_m \vdots t_m	i_1 i_2 i_3 \vdots i_n i_1 i_2 i_3 \vdots i_n \vdots i_1 i_2 i_3 \vdots i_n	t_1, i_1 t_1, i_2 t_1, i_3 \vdots t_1, i_n t_2, i_1 t_2, i_2 \vdots t_2, i_n \vdots t_n, i_m

Table 16. The data arrangement for using the tool to output spectrogram-based distance.

The two files of a concept are used as the input of the tool. The output is a list of pronunciation distances between phonetic transcription pairs per concept. This procedure is iterated for all the concepts. Eventually, every concept has its list of distance values. Various spectrogram-based segment distances are prepared in advance. Three sets of sound distances using Barkfilter,

cochleagram, and formant tracks as acoustic representations are generated by changing the parameter in the executable file. Similar to the PMI-based method, the diacritics in the transcripts are ignored for the sake of simplification in this case. For the purpose of further evaluation, the distance values, concepts, and phonetic transcription pairs are arranged in tuples as shown in Table 2, as follow:

('animal', 3.6941, 'zanwar', 'hajvɔn')

Turkic file of concept 'one'	Indo-Iranian file of concept 'one'	pairs
1 bIr	1 jak	bIr, jak
1 bIr	1 jakta	bIr, jakta
1 bIr	1 jakt@\	bIr, jakt@\
1 bIr	1 jakt6	bIr, jakt6
1 bIr	1 jaktQ	bIr, jaktQ
1 bIr	1 i:	bIr, i:
1 brIw	1 jak	brIw, jak
1 brIw	1 jakta	brIw, jakta
1 brIw	1 jakt@\	brIw, jakt@\
1 brIw	1 jakt6	brIw, jakt6
1 brIw	1 jaktQ	brIw, jaktQ
1 brIw	1 i:	brIw, i:
...

Table 17. Data arrangement for concept "one" for generating distance based on the spectrogram-based algorithm.

Moreover, it is possible that the loanword is detected on the concept level. In this case, the sound distance is calculated between two lists of phonetic transcriptions, instead of between two phonetic transcriptions. Heeringa (2004) calculates the distance between two lists of phonetic transcriptions by finding natural pairs from two lists, as well. The distance between two lists of phonetic transcriptions is then determined by the distances of natural pairs and the number of phonetic transcriptions. Unlike the method used in Nerbonne & Kleiweg (2003), the sum of the distances of natural pairs is divided by the number of phonetic transcriptions⁹.

4.3 Result and Evaluation

Three lists of tuples (each tuple including concept, distance, phonetic transcription in the Turkic family, and phonetic transcriptions in the Indo-Iranian family) are created by using the three acoustic representation of segment distances in the Levenshtein algorithm. Table 18 is the statistical information of sound distance values generated by each method. The histograms of the three sound distance lists show that the distributions of the sound distance values are similar. Unlike the distance values generated by the PMI-based algorithm (Figure 1), the sound distance values are approximately normally distributed (the p-values of Shapiro-Wilk normality test for these three distances lists are all greater than 0.3). Moreover, the number of potential loanwords (the distance values on the left of the x-axis that are smaller) are more frequent than the extreme high distance values on the right of the x-axis.

⁹ See Heeringa (2004) Chapter 5 for more details.

A word from a phonetic transcription pair is classified as a loanword if the sound distance between these two phonetic transcriptions is smaller than a chosen threshold. An interval between the mean value and the minimum value of a list of sound distance values is divided into evenly spaced values, and these values are the potential thresholds. As a result, a predicted loanword list is generated for each acoustic representation. The predicted loanwords are compared against the gold standard, loanword classifications determined by an expert. Also, a 10-fold cross validation is conducted to explore the performance with independent data.

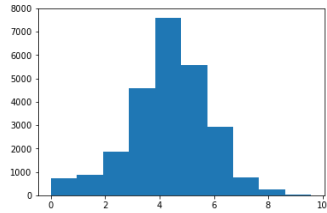
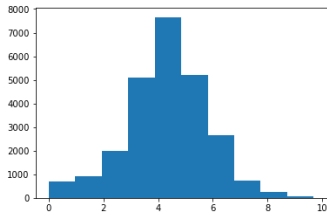
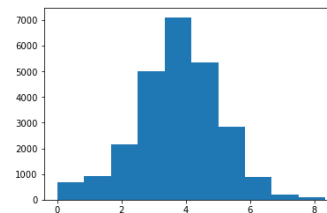
	Barkfilter	Cochleagram	Formant Tracks
Statistical description	Min. :0.000 1st Qu. :3.389 Median :4.254 Mean :4.319 3rd Qu. :5.175 Max. :9.581	Min. :0.000 1st Qu. :3.395 Median :4.264 Mean :4.306 3rd Qu. :5.196 Max. :9.688	Min. :0.000 1st Qu. :2.982 Median :3.732 Mean :3.749 3rd Qu. :4.579 Max. :8.343
Histogram			

Table 18. Statistical description and visualizations of sound distances between phonetic transcriptions of the words from the Turkic and Indo-Iranian families by applying Barkfilter, cochleagram, and formant tracks to measure segment distance.

Table 19 presents selected thresholds achieving the highest F1 scores, and the corresponding precision/recall score. (the explanation of these values is in Chapter 3). Simultaneously, a graph is created for the result of each acoustic representation, which shows the plot of the evaluation results against 200 potential thresholds between mean and minimum.

The results of using the Barkfilter and cochleagram are similar, and the plots of the Barkfilter and cochleagram can be considered as identical. On the other hand, the F1 score of using formant tracks has a tiny disadvantage compared to the ones of the other two. In addition, using the formant track representation provides higher precision since the threshold is significantly lower than the other two representations. Although there is a difference between the plot of formant tracks and the plot of the Barkfilter or cochleagram, the trends of all three plots are similar.

At the low threshold values, the precision is as high as around 0.95 while recall is very low. With the rise of threshold values, precision drops slowly, but there is a significant drop at threshold value of around 1.5 for the Barkfilter and the Cochleagram, while precision keeps smoothly dropping for the formant tracks. A similar development may be seen for recall. For the Barkfilter

and Cochleagram, recall rises sharply but suddenly remains flat at a threshold value of around 1.0, and begins to climb again at a threshold value of around 1.5. But recall keeps rising in a similar slope for formant tracks. After the intersection of the lines, precision drops steadily and recall keeps climbing, reaching a value of almost 1.0. This phenomenon has been explained as the tradeoff between precision and recall. The trend for F1 score line is similar to the trend for recall. The F1 score peaks at 0.7427, 0.7441, and 0.7364 using Barkfilter, cochleagram, and formant track. Although the peak F1 scores are not very different, using formant tracks tends to return higher precision, while the other two tend to favor recall.

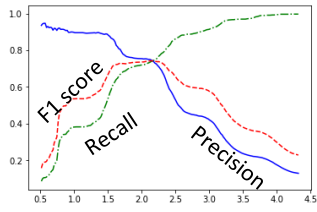
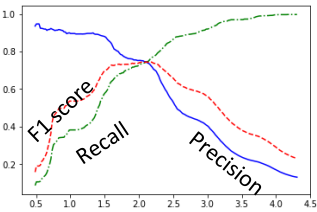
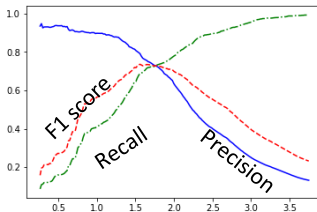
	Barkfilter	Cochleagram	Formant Tracks
Threshold	2.1381	2.1560	1.5585
Precision	0.7467	0.7332	0.7915
Recall	0.7387	0.7553	0.6884
F1 score	0.7427	0.7441	0.7364
Precision, recall, and f1 score against 200 Potential thresholds			

Table 19. Results and evaluation of applying Barkfilter, cochleagram, and formant tracks to detect loanwords.

4.4 Cross Validation

A 10-fold cross validation is applied to assess this model as well. There are ten values of precision, recall, and F1 score for the Barkfilter, cochleagram, and formant tracks representations. Table 20 shows the mean value of the evaluation results.

	Barkfilter	Cochleagram	Formant tracks
Precision	0.7370	0.7024	0.6835
Recall	0.7646	0.7482	0.7721
F1 score	0.7379	0.7127	0.7068

Table 20. Mean value of each evaluation results of a 10-fold cross validation.

4.5 Discussion

As argued in Chapter 3, the F1 score is the standard of deciding the “good performance” of an algorithm in loanword detection. The performance of formant track is not as good as the other two representations. Since using the Barkfilter and the cochleagram return nearly identical F1 scores, either Barkfilter or cochleagram is an appropriate representative of the spectrogram-based Levenshtein algorithm in loanword detection. Besides, the results also show that there is almost

no difference between using Barkfilter and Cochleagram. This means that considering human perception in the representation of sound distances has little effect on loanword detection. On the other hand, the results in cross validation show that the Barkfilter representation has the highest F1 score with 0.7379, which means the Barkfilter performs better than the other two representations when dealing with independent data.

5. SCA SOUND ALGORITHM FOR MEASURING SOUND DISTANCE

Both the PMI-based and the spectrogram-based Levenshtein algorithms are modifications of Levenshtein distance, in which they provide alternative approaches to calculate the substitution cost. Besides calculating distance values between two sequences, these methods are capable of providing corresponding sequence alignments (this is the original purpose of proposing PMI-based algorithm). Actually, alignment analysis is important in sequence comparison.

Refined algorithms can output a sound distance between phonetic transcriptions, leading to a more appropriate sequence alignment simultaneously. Conversely, an appropriate sequence alignment can assist in determining a distance value reflecting pronunciation distance more accurately. List (2012) proposes a modified sound class alignment (SCA), which is used for phonetic alignments based on sound classes. With the improved phonetic alignment proposed by List (2012), a refined distance between phonetic transcriptions is calculated giving the formula of Downey et al. (2008). Downey et al. (2008)'s formula, derived from the ALINE algorithm (Kondrak, 2000), is used to calculate the distance between two phonetic transcriptions when the alignment of the two transcriptions are given. In sum, phonetic alignment analysis and distance calculation are two results of SCA-based sound distance algorithm. The algorithm is realized by LingPy, a python library for historical linguistics (List and Forkel, 2016). Further in this section, the algorithm of SCA is introduced, as well as the formula based on the ALINE algorithm. Furthermore, the implementation of SCA and phonetic distance calculation is explained. Finally, the result of predicting loanwords and the corresponding evaluation of these algorithms are presented as well as the comparison to similar methods previously applied in this dissertation.

5.1 Introduction to the SCA Sound Distance Algorithm

The SCA sound distance algorithm consists of two components: phonetic alignment and distance calculation. The phonetic alignment algorithm proposed by List (2012) is the core of his modification, in which sound classes are applied to a basic pairwise sequence alignment (PSA) model and its extensions. In addition, scoring functions are required to reflect the weights of transitions between sound classes. Also, prosodic features are considered to determine gap penalties and substitution costs. In conclusion, sound classes, the corresponding scoring functions, and prosodic profiles constitute a sequence model in SCA. List (2012) summarizes that phonetic alignment in SCA is conducted in four stages:

- 1) Tokenization: phonetic transcription is tokenized into phonetic segments.
- 2) Class conversion: phonetic segments are converted to be represented by sound classes and prosodic profiles through an SCA sequence model.
- 3) Alignment analysis: List (2012) applies DIALIGN, an extension of PSA, for alignment analysis.
- 4) IPA conversion: The sound classes are converted back to phonetic segments.

After two phonetic transcriptions are aligned, the distance between them is calculated. The algorithm used to calculate the distance is proposed by Downey et al. (2008), and the algorithm is a modification of ALINE algorithm.

5.1.1 Alignment- introduction to PSA

-Sequence modeling in SCA

The sequence model in SCA consists of sound classes, scoring function, and prosodic profiles¹⁰. Sound classes are used to represent phonetic segments in PSA due to the disadvantages of using phonetic sequences directly. Although using phonetic segments in PSA is intuitive, the disadvantage is that different languages might have completely different pronunciation systems, which makes the alignment language dependent. The concept of sound class is proposed by Dolgopolsky (1964), and the main idea is that sounds are divided into different classes according to their phonetic correspondences. In other words, sounds in a given class regularly appear in phonetic correspondence. The sound class model of Dolgopolsky has been used for automatic cognate identification, by investigating the sound classes of the first two consonants of the words (Turchin et al., 2010). Originally, Dolgopolsky introduces ten sound classes, while List (2012) applied an extension Dolgopolsky model with 28 sound classes (Table 21).

No.	CI.	Description	Examples	No.	CI.	Description	Examples
1	A	unrounded back vowels	a, ɑ	15	P	labial plosives	p, b
2	B	labial fricative	f, β	16	R	trills, taps, flaps	r
3	C	dental / alveolar affricates	ts, dz, tʃ, dʒ	17	S	sibilant fricatives	s, z, ʃ, ʒ
4	D	dental fricatives	θ	18	T	dental / alveolar plosives	t, d
5	E	unrounded mid vowels	e, ε	19	U	rounded mid vowels	ɔ, o
6	G	velar and uvular fricatives	χ, x	20	W	labial approx. / fricative	v, w
7	H	laryngeals	h, ʔ	21	Y	rounded front vowels	u, ʊ, y
8	I	unrounded close vowels	i, ɪ	22	0	low even tones	11, 22
9	J	palatal approximant	j	23	1	rising tones	13, 35
10	K	velar and uvular plosives	k, g	24	2	falling tones	51, 53
11	L	lateral approximants	l	25	3	mid even tones	33
12	M	labial nasal	m	26	4	high even tones	44, 55
13	N	nasal	n, ŋ	27	5	short tones	1, 2
14	O	rounded back vowels	œ, ɒ	28	6	complex tones	214

Table 21. An extension of Dolgopolsky model with 28 sound classes used in List (2012).

Dolgopolsky's approach forbids the transitions between sound classes despite the fact that they occur. Scoring functions are designed to represent the probabilities of transition from one sound

¹⁰ Secondary sequence structure is also part of sequence modelling in SCA. It segments a word into syllables rather than sound units. It is useful in monosyllabic languages like Chinese.

class to another. List (2012) applies a theoretical approach to derive the scoring functions¹¹. It considers the directionality of sound changes. The directionality of sound changes means changing from class A to class B is observed while the reverse direction (from B to A) is hardly ever seen. For instance, velar plosives are easily palatalized to affricates, and then to sibilants ([k] or [g] is palatalized to [tʃ], [ts], [dʒ], [dʒ], then to [ʃ], [ʒ], [z], [s]). But the opposite direction (changing from [ʃ], [ʒ], or [tʃ], [ts] to [k], [g]) is rare. A directed weighted graph is used to derive scoring function. Generally, any two closely connected sound classes are connected by a directed edge with a weight, and the weight reflects the probability of changing between those two sound classes. A smaller weight means a higher changing probability. Figure 7 illustrates the direction of changing dentals to affricates weights as 2 while changing from dentals to fricatives weights 4. Hence, the probability of changing dentals to affricates is higher than the probability of changing dentals to fricatives.

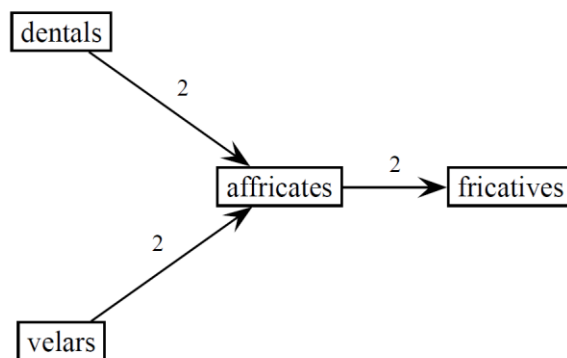


Figure 7. An example presents weights of sound changing (List, 2012).

Prosodic profile, introduced by List (2012b), is a vector representation of a sequence. A score is assigned to each segment according to its sonority decided by the sonority hierarchy (Geisler, 1992), and each segment is assigned to various prosodic environments which are ordered by a hierarchy of strength. The hierarchy of strength leads to relative weights reflecting the penalty of introducing gaps and a bonus for matching environments (Table 22). Each sequence is represented by sound classes and prosodic profiles through the above-defined sequence model, along with scoring functions deciding the probability of transitions between sound classes. Table 22 illustrates the correspondences of phonetic segments, sound classes, and relative weight for a Bulgarian word *jabǎlko* /jabǎlka/ ('apple').

In conclusion, the sequence model used in SCA model of List (2012) is based on the extension of Dolgopolsky model containing 28 sound classes. The scoring function is theoretically derived according to directionality and probability of sound changing. The prosodic profile contributes to determining relative weights, which is used for modifying scores of gap penalty and substitution.

¹¹ Scoring function can be derived by an empirical approach, in which the probabilities are derived according to the sound correspondence frequencies in the language of the world (Brown et al., 2013).

Phonetic transcriptions	j	a	b	ə	l	k	a
Sound classes	J	A	P	E	L	K	A
Prosodic profile	6	7	1	7	5	1	7
Prosodic environment	#	v	C	v	c	C	w
Relative weight	7	3	5	3	4	5	1
Relative scores for Sonority (Prosodic profile): plosives (1), affricates (2), fricatives (3), nasals (4), liquids (5), glides (6), and vowels (7)							
Prosodic environment and hierarchy (Relative weight value in the bracket): # (7, word-initial consonant) > V (6, word-initial vowel) > C (5, ascending sonority) > c (4, descending sonority) > v (3, sonority peak) > \$ (2, word-final consonant) > w (1, word-final vowel)							

Table 22. An example of converting phonetic transcriptions to sound classes, and assigning relative weights. The bottom two rows are sonority hierarchy and prosodic environment hierarchy respectively.

-PSA and its extension

The first PSA algorithm is known as Needleman-Wunsch alignment algorithm (Needleman & Wunsch, 1970). Similar to the Levenshtein algorithm (Table 3), the basic PSA algorithm proceeds by creating a matrix in which scores are filled by comparing two sequences per segment. The path leading to the final score is found by backtracking. In the Levenshtein algorithm, three operations of transforming a sequence to another are introduced and various scores (cost) of these operations are determined. PSA applies a different method. Basic PSA defines that the score is -1 if two segments fail to match or one of them is null (empty), while the score is 1 if two segments match. Therefore, the basic PSA alignment algorithm is also a Levenshtein distance. The difference is that PSA alignment algorithm calculates the similarity between sequences rather than distance.

Modifications are made for the various alignment problems. One of the modifications is the structural extensions¹². Structural extensions allow alignment between two sequences considering the full sequences (global), the partial sequences (local), or a combination of these global and local (DIALIGN). A global alignment tends to treat every segment in a sequence equally, and a local alignment tends to only align partial sequences because of the fact that only part of the sequence is comparable to the other in many cases. DIALIGN conducts alignment analysis globally by considering the whole sequence, as well as analyzing local alignment.

5.1.2 Distance- ALINE Algorithm and its Modification

As long as two phonetic transcriptions are aligned applying the methods introduced above, the distance between the pair of phonetic transcriptions is calculated. In order to obtain a distance value accurately representing the degree of dissimilarity of two phonetic transcriptions, phonetic features ought to be included. List (2012) applied the formula proposed by Downey et al. (2008) to calculate the sound distances between two phonetic transcriptions after the alignment analysis is conducted. The formula of Downey et al. (2008) is a modification of the ALINE algorithm from Kondrak (2000). The ALINE algorithm is proposed specifically for phonetic transcriptions, and it generates similarity scores between phonetic transcriptions for phonetic alignment (Kondrak, 2000). In the ALINE algorithm, each segment is represented as a vector including features in

¹² Another modification is substantial extensions, and it aims to modify the scores by considering phonetic features linguistically.

various categories, such as vowel length, phonation etc., and these features are applied to generate similarity scores between phonetic transcriptions¹³. One of the main disadvantages of ALINE algorithm is that the raw score is not appropriate for comparison between pairs. For instance, the similarity score between /pu/ and /pu/ (50) is different from the similarity score between /tausebasai/ and /tausebasai/ (230), despite the fact that each pair has identical pronunciations. Normalization is required in order to overcome the above-mentioned disadvantage of ALINE algorithm. Downey et al. (2008) improve the algorithm via normalizing the score by the average score of word self-comparisons. Assume two phonetic transcriptions p_1 and p_2 , and the similarity score between them is s . The similarity score between p_1 (or p_2) and itself is s_1 (or s_2).

The normalized similarity score s_{norm} between p_1 and p_2 is:

$$s_{norm} = \frac{2s}{s_1 + s_2}$$

The formula is changed to represent distance (dissimilarity) rather than similarity:

$$d = 1 - \frac{2s}{s_1 + s_2}$$

The distance ranges from 0 to 1 while converging to 1 implies that the two phonetic transcriptions become more similar. The normalization considers the lengths of phonetic transcriptions and penalty of mismatching.

5.2 Realizing SCA Sound Distance Algorithm

It is convenient to apply the SCA sound distance algorithm using the python library LingPy (List and Forkel, 2016). LingPy is a series of open-source Python modules which is developed for quantitative analysis in historical linguistics. It integrates the various methods allowing users to conduct data analysis, such as sequence alignments, tokenization, searching cognates, and distance calculation. The SCA sound distance algorithm by List (201) is realized in LingPy by using appropriate functions and parameters.

Code 5 is the python code for calculating the distance between two phonetic transcriptions. Firstly, the phonetic transcriptions are tokenized into phonetic segments with function:

```
ipa2tokens(ipa)
```

After that the pair of phonetic transcriptions is paired:

```
align_pair_ipa=align.pairwise.Pairwise(tokenized_ipa1,tokenized_ipa2)
```

The pair is aligned and the parameters indicate that the distance value is outputted and the SCA method is used:

```
align_pair_ipa.align(distance=True, method='sca')
```

¹³ Detail is in Kondrak (2000).

As noted above, the SCA model allows three extensions of PSA which are global, local, and DIALIGN respectively. In this experiment, “global” is applied, which is the common method to treat every segment equally like the other distance algorithms used in this dissertation.

The distance is extracted by

```
distance=align_pair_ipa.alignments[0][2]
```

```
from lingpy import *

phonetic_pairs=[ipa1,ipa2]

## Tokenizes the phonetic transcriptions into phonetic segments.
for ipa in phonetic_pairs:
    tokenized_ipa_pairs.append(ipa2tokens(ipa))

## Pairs the two tokenized phonetic transcriptions.
align_pair_ipa=align.pairwise.Pairwise(tokenized_ipa_pairs[0],to-
kenized_ipa_pairs[1])

## Aligns the pair of tokenized phonetic transcriptions.
## The two parameters specify that the distance will be outputted and the
## method used is SCA.
align_pair_ipa.align(distance=True,method='sca')

## Output aligned phonetic transcriptions and distance between them.
aligned_ipa1,aligned_ipa2,distance=align_pair_ipa.alignments[0]
```

Code 5. Python code for determining the distance between two phonetic transcriptions using the SCA algorithm.

It is notable that distance between phonetic transcriptions generated by LingPy is possibly higher than 1, which contradicts the formula of Downey et al. (2008). The reason is that the VC sensitive algorithm (introduced in section 2.2) is applied by LingPy, and the SCA algorithm is realized upon the VC sensitive algorithm. It means that substituting a vowel for a vowel or consonant affects the distance derived by the SCA-based algorithm because VC sensitive algorithm avoids the substitution between consonant and vowel. This causes the possibility of obtaining a negative similarity value. In other words, the distance value may be more than 1 according to the distance formula.

It is straightforward to build an appropriate data structure to store the phonetic transcription pairs so that the data structure is applied in Code 5. Each pair is represented as a three-element tuple including concept and the two phonetic transcriptions. More than 25 thousands pairs are generated and they are put in a list, as:

```
[('one', 'bir', 'jak'),
 ('one', 'bir', 'jakta'),
 ('one', 'bir', 'jaktə'),
 ('one', 'bir', 'jaktə'),
 ...
 ('other', 'bofqa', 'diga'),
 ('other', 'bofqa', 'ani'),
 ('other', 'bofqa', 'axti')]
```

Eventually, a list of tuples is generated. Each tuple consists of the concept, a pair of phonetic transcriptions, where one transcription comes from a Turkic family language and the other from an Indo-Iranian family language, and the distance value between them. Each generated pair consists of one phonetic transcription in one language family, and one from the other per concept (as mentioned in Section 2.1). There are more than 25 thousands tuples in the result list, and each tuple is in the format of:

('animal', 0.60769, 'zanwar', 'hajvön')

5.3 Result and Evaluation

A list of tuples containing the concept, distance values, and a pair of phonetic transcriptions are generated applying SCA distance algorithm implemented by LingPy library. The distance values are the sound distances between phonetic transcriptions from the Turkic family and the Indo-Iranian family. Table 23 is the statistical description presenting related facts of the distance values generated from all the word pairs from all the concepts.

Min.	0.0000
1st Qu.	0.5789
Median	0.7077
Mean	0.6642
3rd Qu.	0.8142
Max.	1.1670

Table 23. Statistical description of the sound distances derived from the SCA-based algorithm.

Figure 8 is the histogram presenting the distribution of distance values. In general, the distribution is slightly skewed to the right. The majority of the values are between 0.6 and 0.9. Both extremely low distance values and extremely high values are not common. This histogram is similar to the one generated by the PMI-based algorithm (Figure 1). The notable feature is that there is a small “peak” at the values below 0.1, and those values are regarded as potential loanwords. This fact might lead the algorithm to detecting more loanwords compared to the MI-based algorithm.

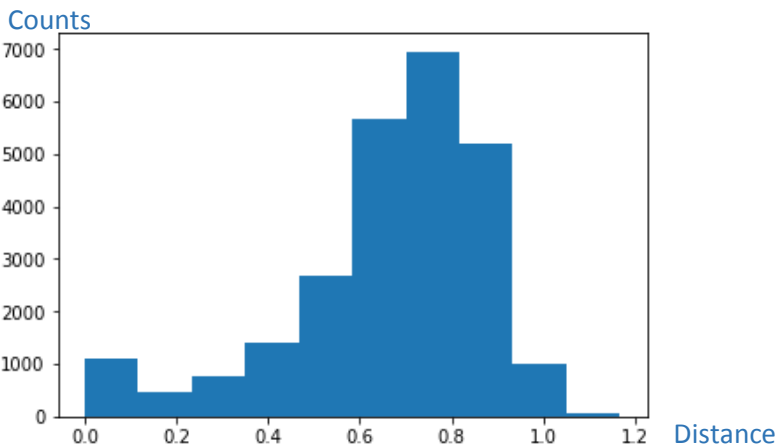


Figure 8. Histogram of sound distances between phonetic transcriptions from the Turkic and Indo-Iranian family using SCA distance algorithm.

5.3.1 Determining Threshold

The word represented by one of the phonetic transcriptions in a pair is classified as loanword if the sound distance of the pair is smaller than a selected threshold. Using the methods above, 200 potential threshold values are generated by evenly dividing an interval between smallest distance values other than zero (0.1667) and the mean of the distance values (0.6642) into 200 pieces. The threshold leading to the highest F1 score value is the most suitable threshold. F1 score and other evaluation values are calculated by comparing detected loanwords to the gold standard. Figure 8 is a plot presenting the precision, recall, and F1 scores under the 200 potential thresholds. Similar to the plots generated by PMI-based and spectrogram-based methods, there is a tradeoff between precision and recall values. Precision drops down from the high value at small threshold values, while recall climbs up from the low value to high along with the increasing of thresholds. It is notable that recall increases rapidly under threshold value of 0.1, while precision drops in a stable manner. This fact is reflected by the gathering of low distance values under 0.1 in the histogram (Figure 7). Recall rises more moderately after threshold is higher than 0.1, but the absolute slope value is still greater than the one of precision dropping. The intersection is at the threshold value of around 0.31. Precision keeps declining to a low level, and recall remains rising to almost 1 with the rising of threshold values. The F1 score is used to reflect the performance of the algorithm. It climbs up in a similar way as recall climbs before the intersection, and drops down afterward. The highest F1 score is at the threshold of 0.297. The evaluation values are summarized below:

Threshold	0.297
F1 score	0.8512
Precision	0.8834
Recall	0.8213

Table 24. Summary of the performance using SCA-based algorithm.

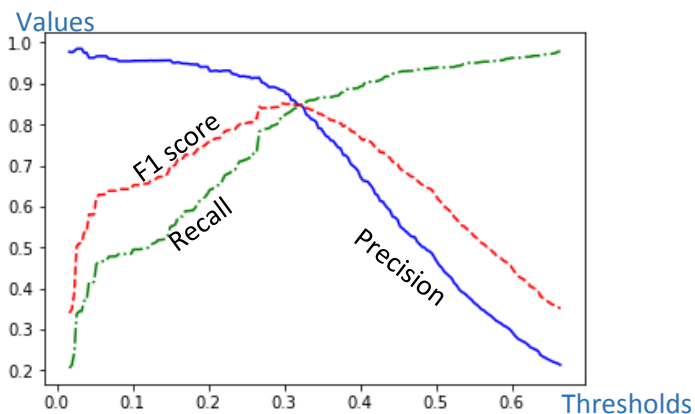


Figure 9. Precision, recall, and F1 score against the 200 potential thresholds.

5.3.2 Cross Validation

A 10-fold cross validation is conducted in order to examine the performance of the algorithm when independent data is used. The method is illustrated in Figure 3. The average precision, recall, and F1 score of the cross validation values are summarized as follows:

Precision	0.8346
Recall	0.8019
F1 score	0.8128

Table 25. Mean scores in 10-fold cross validation for the PMI-based Levenshtein algorithm.

5.4 Discussion

The SCA distance algorithm outperforms PMI-based Levenshtein algorithm and spectrogram-based Levenshtein algorithm when comparing F1 score. While applying LingPy to tokenize phonetic transcriptions, certain segments fail to be segmented probably. For instance, /pf/ is probably /p^hf/ and LingPy is capable of segmenting /p^hf/ as a sound unit, but LingPy treats /pf/ as two separated units. This might slightly affect the distance values. Since /pf/ is used in the other algorithms, the side effect caused by this pitfall is negligible.

6. COMPARISON OF THE THREE ALGORITHMS

So far, a model is designed to detect the pairs of the words containing loanwords. The core of the model is to discover the best value as a threshold to classify loanwords, and the F1 score is used to represent the performance of an algorithm at a given threshold. Thus, the model is able to evaluate the performances of the three refined edit distance algorithms in loanword detection. The performance of the three algorithms is compared from various perspectives. First of all, the performances of the algorithms in loanword detection are compared by directly comparing the evaluation values derived from the models. Meanwhile, the performance of the algorithms dealing with independent data is compared as well (cross validation). Secondly, the distributions of the distance values derived from these algorithms are compared in order to explore the distribution differences. Thirdly, the results of detected loanwords are compared. There is a list of predicted loanwords generated by using each algorithm to detect loanwords. Every list contains correctly detected loanwords, incorrectly detected loanwords, loanwords that are not detected, etc. The characteristics of these words in each algorithm are investigated and compared to other algorithms. Fourthly, an alternative method, detecting outliers, is used to classify loanwords rather than discovering the threshold outputting the highest F1 score. Comparison between these two methods is related to the practical application of applying this model to detect loanwords.

6.1 Comparing Performances

6.1.1 Thresholds and Performances

A threshold is required to classify loanwords. If the distance value of a pair of words is lower than a threshold, this pair is classified as containing a loanword. Otherwise, there is no loanword in the pair. The “best threshold” is obtained by examining 200 potential thresholds from the interval between the minimum distance value and the mean value of the distances of each algorithm. The F1 score is used to evaluate the performance of an algorithm at a given threshold.

The graphs in the right column of Table 26 show the values of precision, recall, and the F1 score of the selected 200 potential thresholds per algorithm. The graphs present the change of the three

evaluation values along with the increase of threshold values. In general, precision is positively correlated to the threshold, while recall is negatively correlated to the threshold. The graphs of each algorithm in the right column of Table 26 show that precision is high (almost 1.0) at small threshold values, and it keeps declining while the threshold increases. In contrast, recall is low at the smaller threshold values, and it keeps climbing while the threshold increases. The F1 score is the result of considering both precision and recall. The F1 score climbs along with the increasing of thresholds, peaks at a certain point (the best threshold), and keeps dropping after the peak. Although the shape of the lines varies in each algorithm, the trends are identical as described above.

Evidently, the SCA-based edit distance algorithm outperforms the other two. The F1 score of applying the SCA-based algorithm to detect loanwords is more than 0.85, which is significantly higher than the F1 score derived from PMI-based algorithms (0.7847) or Barkfilter algorithm (0.7427). Meanwhile, both corresponding precision and recall values of the SCA-based algorithm are higher than the other two as well (Table 27).

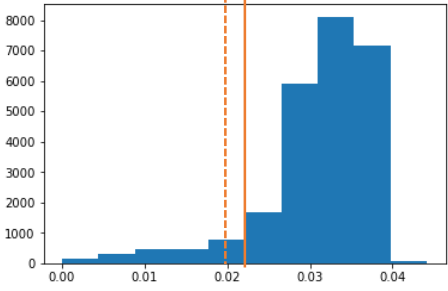
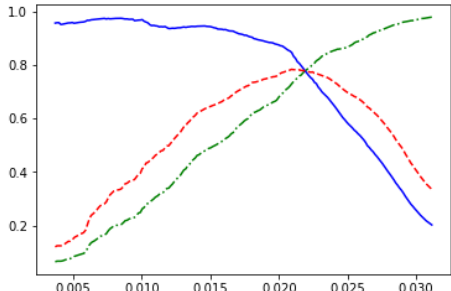
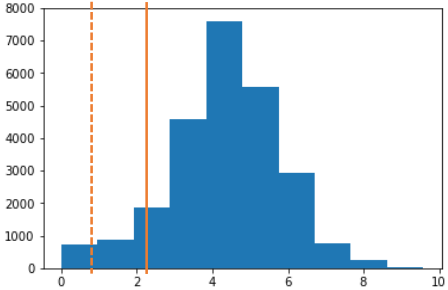
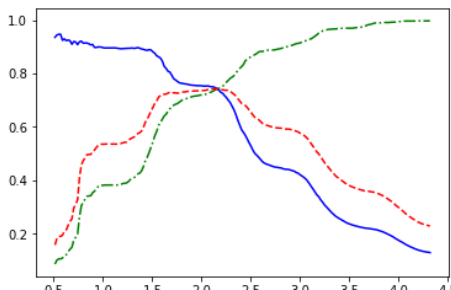
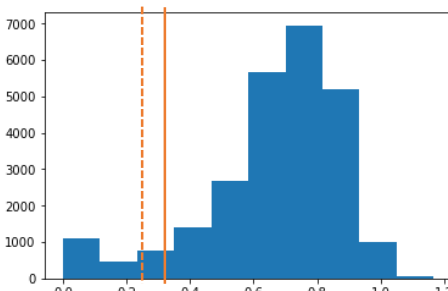
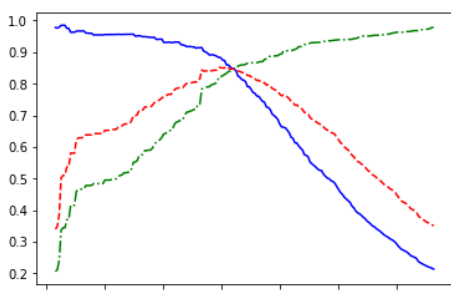
	<p>Distribution (Solid line: the position of best threshold. Dash line: the position of lower outlier boundary.)</p>	<p>P/R, F1 score against 200 potential thresholds (Blue solid line: precision; Red dash line: F1 score; Green mixed solid and dash line: recall.)</p>
PMI	 <p>Lower Outlier boundary: 0.01934 Best threshold: 0.021</p>	
Spectrogram (Barkfilter)	 <p>Lower Outlier boundary: 0.71 Best threshold: 2.1381</p>	
SCA	 <p>Lower Outlier boundary: 0.22595 Best threshold: 0.297</p>	

Table 26. Comparison of the three algorithms. The first column is the histogram presenting the distribution of distance values derived by each algorithm. The second column is the plots of evaluation values against 200 potential thresholds of each algorithm.

	PMI	Spectrogram	SCA
F1 score	0.7847	0.7427	0.8513
Precision	0.8405	0.7467	0.8831
Recall	0.7357	0.7387	0.8218

Table 27. F1 score, precision, and recall of the three algorithms when the best threshold is used. In other words, the highest F1 score of each algorithm and their corresponding precision and recall.

6.1.2 Determining Threshold Using Lower Outlier Boundary

An alternative method to extract an appropriate threshold is to calculate the lower outlier boundary of the list of distance values. In a list of the observation points, outliers are the points that are distant from other points. Mathematically, it is defined as the points outside the range of:

$$[Q_1 - k(Q_3 - Q_1), Q_3 + k(Q_3 - Q_1)]$$

Where:

Q_1 is the first quartile of a list;

Q_3 is the third quartile of a list;

k is a positive constant.

Tukey (1977) proposed that $k=1.5$ indicates an “outlier”. The lower outlier boundary is the boundary separating the outliers that are smaller than the other points from the rest. In other words, lower outlier boundary is

$$Q_1 - 1.5(Q_3 - Q_1)$$

If a distance value of a pair of word is lower than the defined lower outlier boundary, the distance value is exceptionally smaller than the other distance values. The pair of words classified as an outlier is exceptional to other pairs as well. Hence, it is intuitive to classify a pair of words containing a loanword if the distance value of the pair is lower than the lower outlier boundary.

Generally, the value of the lower outlier boundary is lower than the best threshold per algorithm (Table 28). Since the lower outlier boundary values differ from the best threshold, the F1 scores derived from the lower outlier boundary values are probably lower than those derived from the best thresholds. The difference of F1 scores is very small in case of PMI-based algorithm, while the difference is slightly more obvious in case of SCA-based algorithm. However, the difference of F1 score in case of Barkfilter is huge. The F1 score is only 0.2971 when the lower outlier boundary is used as threshold. The reason is that the lower outlier boundary value is much smaller than the best threshold. The more standard normality of the distribution of distance values from Barkfilter-based algorithm leads to fewer outliers compared to the outliers in case of the other two algorithms. Using a low value as threshold possibly causes low recall value. Actually, the recall is 0.1771 when using the lower outlier boundary as the threshold in case of Barkfilter algorithm. Using the lower outlier boundary value as threshold tend to obtain higher precision (using Barkfilter algorithm is considered an extreme situation). In practice, this method reduces computational cost dramatically, and the bias to precision can be ignored, (sometimes it may be preferred) unless it causes extreme low recall.

	PMI	Barkfilter	SCA
Lower Outlier boundary value/ its corresponding F1 score	0.01934/ 0.7516	0.71/ 0.2971	0.22595/ 0.7867
Best threshold value/ its corresponding F1 score	0.021/ 0.7847	2.1381/ 0.7427	0.297/ 0.8513

Table 28. Comparison of performances between using lower outlier boundary as the threshold and using best threshold value as the threshold.

6.1.3 Cross Validation

Cross validation is used to evaluate the performance of a model when dealing with independent data. The data of pairs of words are split into a training set and a testing set.

In this dissertation, applying one of the three algorithms to the model to detect loanwords with training data outputs a best threshold for each algorithm, and the best threshold is applied to the testing data so that the performance of the model is evaluated. In order to reduce the computational cost, there are only ten (instead of 200) potential thresholds selected by evenly dividing the interval between the minimum distance value and the mean of distance values into ten parts. The cross validation result shows that the SCA-based algorithm outperforms the PMI-based algorithm and the Barkfilter-based algorithm, and the PMI-based algorithm outperforms the Barkfilter-based algorithm. This performance ranks are identical to those obtained by using the best thresholds from 200 candidates. In order to explore the performance of dealing with independent data, the cross validation result ought to be compared to the situation that cross validation is not conducted. To ensure comparability, the evaluation result generated by determining the best threshold from ten candidates (instead of 200) is compared to the cross validation because ten potential thresholds are used in each iteration in the cross evaluation. It turns out there is no huge difference between those two evaluation results in case of all three algorithms (Table 29). In other words, the respective loanwords detection models generated by the three algorithms is capable to deal with independent data.

	PMI	Spectrogram	SCA
Cross validation			
Precision	0.760	0.7370	0.8346
Recall	0.795	0.7646	0.8019
F1 score	0.768	0.7379	0.8128
10 thresholds			
Precision	0.8808	0.7273	0.8752
Recall	0.6628	0.7488	0.8293
F1 score	0.7564	0.7379	0.8516

Table 29. Cross validation results of the three algorithms, and performances of obtaining best threshold from “ten threshold candidates”.

It should be noticed that the cross validation result of the PMI-based algorithm may not accurately reflect the actual ability to deal with independent data. The reason is that the PMI value between segments should be calculated based on the data in training set rather than all data. Hence, simply using distance values calculated according to the PMI values derived from the whole data set may affect the result of cross validation in this case. There is a possible solution. In each iteration, PMI values between all segment pairs are calculated based on the word pairs in training set, and the

distance values of the word pairs in training set and testing set should be calculated based on that PMI values. Considering the size of this dissertation, this is suggested in future works.

6.2 Comparing Distribution of Distance Values

There are approximately 25K distance values between two phonetic transcriptions generated per algorithm. The number of distance values is not the same per algorithm because certain phonetic transcriptions fail to be compared to the others in an algorithm. The difference of distance values quantities among the algorithms are small compared to the total number of the distance values (Table 30), the difference will not affect the comparison of the three algorithms.

Algorithm	Number of distance values
PMI	25103
Barkfilter	25237
SCA	25290

Table 30. The number of distance values generated by each algorithm.

Table 31 provides a statistical summary of the distance values derived by the three algorithms. The statistical descriptions include features of distance values, such as mean, median, etc., and these features offer an overview of the data distribution with concrete digits. Since the mechanisms of calculating distance are different from each other, it is meaningless to compare the statistical description among the three algorithms quantitatively. In order to compare the distribution of the distance values, using histograms is more appropriate.

	PMI	Barkfilter	SCA
Min.	0	0	0
1st Qu.	0.02909	3.389	0.5789
Median	0.03325	4.254	0.7077
Mean	0.03113	4.319	0.6642
3rd Qu.	0.03559	5.175	0.8142
Max.	0.04420	9.581	1.1670

Table 31. Statistical descriptions of sound distances derived from each algorithm.

A histogram is commonly used to visualize the distribution of data. Hence, observing the histogram contributes to interpreting the distance values generated from an algorithm. Meanwhile, it is intuitive to realize the approximate percentage of loanwords focusing on the low distance values. The first column of Table 26 shows the histogram of each algorithm. The distributions of the distance values extracted from the PMI algorithm and the SCA algorithm are similar in that the two histograms are skewed to the right. Hence, the low distance values are not common in these two situations, and the majority of the distance values gather in a narrow interval covering relatively high values. On the other hand, the distance values derived from spectrogram-based algorithm are normally distributed, where both low and high distance values are infrequent. The majority of the distance values cluster around the mean of the distance values. In the case of SCA, it is notable that there is a small peak at the position of extreme small distance values in its histogram, while the number of distance values increases along with the increase of distance values until they peak at a certain point in the other two cases.

The position of the best threshold of each algorithm is marked as a solid line in the respective histogram (Table 26 first column). In the histogram generated from the PMI-based algorithm, the best threshold is at the middle of the interval between the minimum value and the mean value. In the case of the other two algorithms, the best thresholds are closer to the smallest distance value. Obviously, the best threshold affects the number of detected loanwords per algorithm (Table 32). It is intuitive to estimate the number of detected loanwords derived from each algorithm by observing the distribution before the best threshold line in the respective histogram. In the case of the PMI-based algorithm, although the best threshold is more to the middle of the histogram than the SCA-based algorithm, there are fewer word pairs classified as containing loanwords. Meanwhile, the number of word pairs classified as containing loanwords in the case of the Barkfilter-based algorithm is less than the one in the case of the SCA-based algorithm in spite of the similar positions of their best threshold.

Algorithm	Number of detected loanwords
PMI	1815
Barkfilter	1661
SCA	1984

Table 32. The number of predicted loanwords generated by each algorithm using their respective best threshold.

6.3 Comparing Results of Detected Loanwords

Using the best threshold in the model with each algorithm results in a list of word pairs which are predicted to involve a loan. There are three predicted loanword lists corresponding to each algorithm. Some expert-classified loanwords are not found in a predicted loanword list using one of the algorithms to detect loanwords, or not found in any predicted loanword list (false negative). On the other hand, some pairs are not classified as containing loanwords by experts, but they appear in the predicted loanword lists (false positive). The third list consists of genuine words that were classified correctly as such. Figure 10 is a Venn diagram presenting the relations of the predicted loanword lists of each algorithm and gold standard. The rectangle contains all the pairs of words in the data. The circle in the light color represents the pairs containing loanwords in gold standard. The three circles represent the pairs detected as containing loanwords by the three algorithms respectively. The intersections of the different sets represent possible situations about loanwords across each predicted loanwords list of an algorithm. For instance, some pairs are detected by one or some of the algorithms, and they may be in or outside the gold standard.

6.3.1 Result Analysis

When a word is borrowed from a donor language, adjustments are required in order to fit the linguistic features of the recipient language. Previous studies show that there are phonological adjustments when loanword are adapted into the recipient language because of the phonological features and constraints of the recipient language (Yip, 1993; Tsuchida, 1995; Miao, 2005). By sketchily studying the gold standard, the patterns of phonological adaptation between the Turkic family and the Indo-Iranian family may be discerned. The results explain the reason for the superior performances of the SCA-based algorithm as we will explain shortly.

The study of the gold standard shows that epenthesis (or deletion) and substitution appear in the phonological adaption of loanwords between the Turkic family and Indo-Iranian family.

Epenthesis or deletion means that a segment is added or deleted when a word is borrowed by recipient language (Table 33), while substitution means a segment is substituted with one or more segments. Notably, the major operation is the substitution.

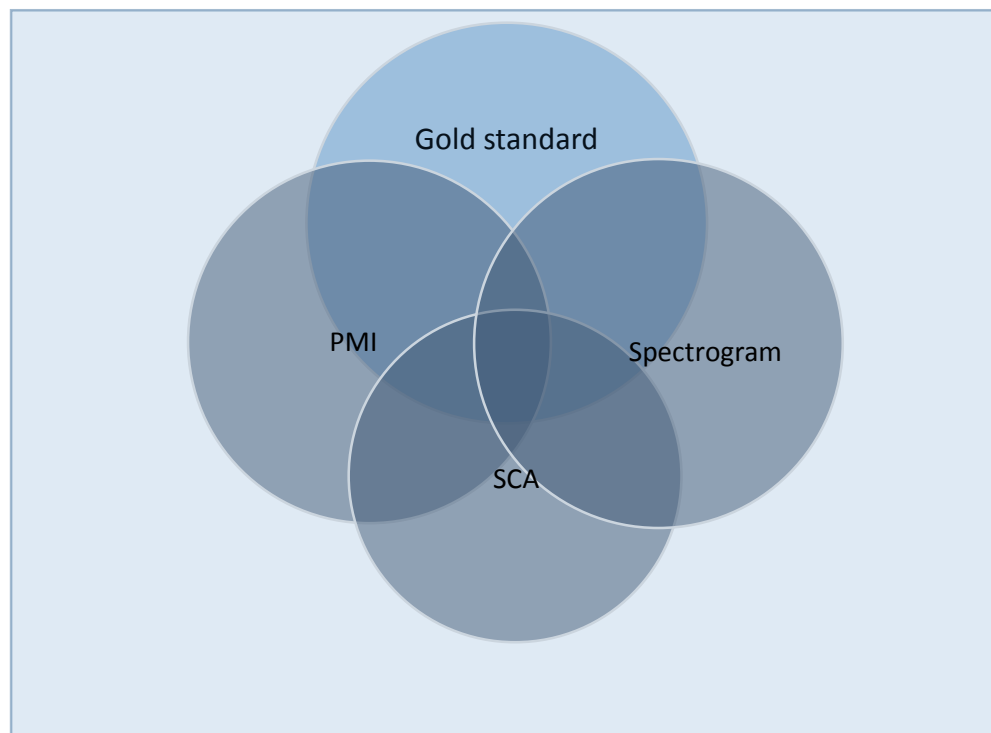


Figure 10. Venn diagram showing schematically the overlap between the genuine loanwords and those predicted by the algorithms. For instance, the intersection between the set of gold standard and any one of the algorithm set represents the correctly predicted loanwords of each algorithm. The part of the intersection of the three algorithm sets outside gold standard represents the loanwords that incorrectly detected by all the three algorithms.

('correct', 'tu:ra', 'tøʁi')
('forest', 'wormon', 'urmøn')
('meat', 'gwʊft', 'guft')
('lake', 'kwøɫ', 'kul')
('old', 'kwønie', 'kujna')

Table 33. Examples of epenthesis (or deletion) in phonological adaptation when borrowing words.

The substitution of some segments for others is regular. For instance, [a] is commonly substituted for (or by) [e], [ɔ], and [ɒ], but it is rarely substituted for (or by) [u]. Table 34 is the summary of the substitution patterns. The segments that are interchangeable share similar pronunciations (at least their pronunciations are similar according to human perception). More importantly, Table 34 shows that the sound segments are divided into groups to some degree. It implies that the lower distance value between any two segments within a group can raise the probability of detecting loanwords correctly.

	Commonly seen Segments that are interchanged	Examples (a,b,c) from gold standard a: concept b: phonetic transcription in Turkic family c: phonetic transcription in Indo-Iranian family
Vowels	[a], [e], [ɔ], [ɐ]	('animal', 'hajβan', 'hajβɔn') (('short', 'k'elte', 'kalta') (('fruit', 'miβe', 'meva') (('breast', 'køkriek', 'kukrak') (('wing', 'qanat', 'qanɔt') (('wind', 'ʃamɒl', 'samɔl')
	[i], [i̯], [ɪ], [ə]	('smooth', 'silləq', 'sillɪq') (('smooth', 'tekis', 'tekkəz') (('correct', 'tu:rə', 'tuyri') (('root', 'təm̩ir', 'təm̩ər') (('fruit', 'm̩iva', 'miva') (('straight', 'təri', 'tøyri') (('louse', 'bit', 'bit') (('breast', 'kukrik', 'qoqrak')
	[ə], [u], [ɜ], [ʊ], [ø], [ʊ], [ɔ]	('egg', 'tuxum', 'txəm') (('egg', 'tqum', 'txəm') (('dog', 'kəʃək', 'kuʃuk') (('louse', 'ʃubəʃ', 'ʃopuʃ') (('forest', 'wɔrman', 'urmɔn') (('flower', 'gul', 'gɔl') (('forest', 'wɔrman', 'ør̩mɔn') (('breast', 'køk̩rək', 'kukrak')
Consonants	[k], [q], [g]	('leaf', 'parak', 'barg') (('breast', 'køk̩rək', 'kukraq')
	[q], [x], [χ]	('blood', 'qan', 'xun') (('back', 'ɔrqa', 'arχa')
	[t], [d]	('tree', 't̩erək', 'daraxt') (('sea', 't̩en̩iz̩', 'den̩iz̩')
	[β], [v], [w], [f]	('animal', 'ajβan', 'hajvɔn') (('animal', 'hajβan', 'hajwon') (('to dig', 'kaβl̩em', 'kɔftan') (('fruit', 'miwe', 'm̩eva')
	[ʃ], [ʃ̩], [dʒ], [j], [s]	('dust', 'ʃan', 't̩ʃan') (('to live', 'd̩ʒaʃa', 'jaʃam') (('star', 'd̩ʒildiz̩', 'jildiz̩') (('bird', 'qus', 'kuʃ') (('star', 'ʃuldiz̩', 'jildiz̩')

Table 34. Summary of segment substitution between loanwords in the gold standard from the data of Central Asian languages.

The importance of regular segment substitution is reflected in the PMI-based algorithm, the spectrogram-based algorithm, and the SCA-based algorithm. The segment distance in the PMI-based algorithm is based on the probability of two-segment alignment. On the other hand, the

segments in the same group in Table 34 may have similar acoustic features due to the similar pronunciations of the segments in the same group. In the case of the SCA-based algorithm, the division is similar to the division generated by sound classes (Table 21). Since all these three algorithms reflect the features of segment substitution in loanwords, why does the SCA-based algorithm outperform the other two? The reason may be related to two aspects.

The first reason is that the SCA-based algorithm uses DIALIGN alignment (considering full and partial sequence) between two sequences before calculating the distance, while the sequence alignments of the other two algorithms are based exclusively on segment distance. Hence, alignment and distance calculation are more independent in the SCA-based algorithm.

The second reason is the method of distance calculation. In the SCA-algorithm, the distance between two segments within a sound class is zero, and the distance between two segments from different sound classes has a fixed distance according to which two sound classes are compared. This is the reason for the small peak at the low value in the histogram of SCA-algorithm result (Figure 8) probably. Also, it explains the frequent appearance of identical sound distance values. Moreover, the distance between segments from two varied sound classes is larger than the two from the same sound class. In the case of the other two algorithms, the distance between any two segments is unique and more concrete in general. Unlike the SCA-based algorithm, the distance between two segments within a sound class is almost never zero. Nevertheless, the distance between any two segments within a group is smaller than two from two different groups.

In summary, although PMI-based algorithm and spectrogram-based algorithm provide more concrete sound distances between two segments than what SCA-based algorithm does theoretically, the distance calculation scheme of the SCA-based algorithm is superior in detecting loanwords. Assigning “zero distance” to any two segments within a sound class is more appropriate given the fact that substitution within a sound class models the phonological adaptation of loanwords between Turkic language family and Indo-Iranian language family. The smaller difference between two segments within a sound class raises the probability of classifying two sequences involving those two segments as containing loanword.

7. CONCLUSION

There are various refined edit distance algorithms which are sensitive in measuring the sound distance between two words, and loanwords can be detected by comparing distances between pairs of words. This dissertation attempts to compare the performance of applying three refined edit distance algorithms to measure sound distance, and the derived distance values are applied to detect loanwords. The algorithms used in this dissertation are the PMI-based edit distance algorithm, the spectrogram-based edit distance algorithm, and the SCA edit distance algorithm. The performance evaluations of the three algorithms in loanword detection show that the SCA-based algorithm outperforms the PMI-based algorithm and the spectrogram-based algorithm, and the PMI-based algorithm is slightly better than the spectrogram-based algorithm. Moreover, realizing SCA-based algorithm is more convenient than the other two algorithms. The tools used to calculate the PMI-based distance and spectrogram-based distance require the tedious preparation of pre-processing the data, and it is complicated to use those tools is complicated since they are based on a non-popular platform. In contrast, SCA-based distance is released by a Python library, which is efficient, and the pre-processing of data is straightforward. The convenience of

calculating SCA-based distance is beneficial to the development of an application for loanwords detection. In conclusion, the result shows that it is feasible to apply Levenshtein distance to represent the sound distance between two words written in phonetic transcriptions, and it is effective to apply the sound distance between words for loanword detection between two unrelated languages. Amongst the algorithms investigated in this dissertation, applying the SCA-based algorithm is appropriate to detect loanwords, and it outperforms algorithms and methods (such as ancestral state reconstruction (Köllner & Dellert, 2016) and string similarity (Mi et al., 2014)) used in previous studies as well.

Each pair of words is assigned a distance value representing the sound distance between the words in a pair. The model introduced in this dissertation classifies the loanwords by an empirically determined threshold. In other words, if the distance value of a pair is lower than the threshold, the pair is classified as containing loanword, otherwise, it is not. Determining an appropriate threshold is core to classifying loanwords. Using the “200 potential thresholds” method we are able to find the best threshold which leads to the highest F1 score. The pitfall is that the computational expense is high since the model needs to be run for 200 times.

In practice, the search for the best thresholds from the potential thresholds can be terminated as soon as the F1 score begins to decline because there is only one extremum in the line of the F1 score. “Only one extremum” implies that the highest value before value declines is the highest value globally. Obviously, early termination of the investigation of potential thresholds reduces the cost of computation. An alternative method is simply to use lower outlier boundary as a threshold. Although the best performance may not be reached, the computational expense is reduced significantly. It is notable that the F1 score is not the only indicator of reflecting the performance because the highest F1 score is not always desired. High precision may be desired if incorrect detection is not tolerated, while high recall may be desired if it is required to detect more loanwords. Therefore, using F1 score as an indicator of performance is not always appropriate.

It is possible that the pronunciation of word w_1 from language A is similar to the pronunciation of word w_2 from language B, but neither w_1 nor w_2 is a loanword. For instance, the comparison section above shows that all the pairs of some words are expertly classified as “not loanword”, but they are predicted as loanwords whatever algorithm is used. The concept “I” is mainly pronounced as /mʲen/ in Turkic family, and /man/ in the Indo-Iranian family. These two pronunciations are not cognates. However, their pronunciation is similar enough that the models developed by all the three algorithms classify this pair containing a loanword (Table 10).

	Distance	Lower outlier boundary	Best threshold
PMI	0.0106	0.0193	0.021
Barkfilter	0.7628	0.71	2.1381
SCA	0.04	0.22595	0.297

Table 35. The distance between /mʲen/ (Turkic family) and /man/ (Indo-Iranian family) generated from the three algorithms. Comparing with the respective lower outlier boundary and best threshold, the distances are lower or a little higher than lower outlier boundary. Hence, /mʲen/ and /man/ have similar pronunciation.)

If a pair of words are not cognates but share similar pronunciation and meaning, they are called false cognates. Examples of false cognates are ‘saint’ (/sɛ̃/) in French and ‘聖 (sheng)’ (/ʃəŋ/) in Mandarin (meaning “saint”), ‘斬る (kiru)’ /kʲiru/ in Japanese and ‘kill’ /kɪl/ in English. The words representing the concept “I” in Turkic family and Indo-Iranian family are probably false cognates.

The existence of false cognates may influence the principal hypothesis of this dissertation. The result of applying the model introduced in this dissertation shows that there are only 48 pairs of words (accounting for less than 0.2% of the total pairs of words) are incorrectly detected as containing loanwords no matter which algorithm is used. Although detecting loanwords via the sound similarities between words is not always reliable, false cognates has little effect to the performance of the model due to their low proportion.

A sensitive distance value is important to accurately represent the sound difference between words, and this is one of the reasons for applying the refined algorithm to calculate sound distance. However, the algorithms used in this experiment are not perfect. For instance, the sound samples used in the spectrogram-based algorithm for extracting acoustic features may not be representative, meaning that the data might fail to reflect the acoustic features of pronunciation in at least some languages.

As for PMI-based algorithm, the PMI of two segments is influenced by the dataset itself because the calculation of PMI depends on the available data. The performance of the algorithm in loanword detection may be worse than the cross validation result. The pitfall of the algorithms may influence the performance of the algorithm in loanwords detection.

For the purpose of loanword detection, these algorithms can be improved in future study. For instance, the sound samples used in the spectrogram-based algorithm might be collected by other people with different mother tongue so that the bias to certain accents is diminished.

It is possible that the performance of these algorithms in loanword detection could vary if evaluated on similar data of other languages instead of languages in Central Asia. Applying the model to other languages would be useful in order to examine the universality of the model. Besides, the evaluation of the performance of the refined edit distance algorithms in loanword detection would be more convincing. Moreover, an application can be developed properly for loanword detection between two unrelated languages, for instance an application with user-friendly interface. Such an application would be useful for the studies in historical linguistics.

BIBLIOGRAPHY

- Brown, Cecil., Holman, Eric., & Wichmann, Søren. (2013). Sound correspondences in the world's languages. *Language*, 89(1), 4-29.
- Church, Kenneth., & Hanks, Patrick. (1990). Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1), 22-29.
- Delz, Marisa. (2013). A theoretical approach to automatic loanword detection (Master thesis, Eberhard-Karls-Universität Tübingen).
- Dolgopolsky, Aharon. (1964). Gipoteza drevnejšego rodstva jazykovych semej Severnoj Evrazii s verojatnostej točki zrenija [A probabilistic hypothesis concerning the oldest relationships among the language families of Northern Eurasia]. *Voprosy jazykoznanija*, 2, 53-63.
- Downey, Sean., Hallmark, Brian., Cox, Murray., Norquest, Peter., & Lansing, Stephen. (2008). Computational feature-sensitive reconstruction of language relationships: Developing the

- ALINE distance for comparative historical linguistic reconstruction. *Journal of Quantitative Linguistics*, 15(4), 340-369.
- Geisler, Hans. (1992). *Akzent und Lautwandel in der Romania* (Vol. 38). Gunter Narr Verlag.
- Haspelmath, Martin., & Tadmor, Uri. (2009). *Loanwords in the world's languages: a comparative handbook*. Walter de Gruyter.
- Heeringa, Wilbert. (2004). *Measuring Dialect Pronunciation Differences using Levenshtein Distance* (Doctoral dissertation, University of Groningen)
- Köllner, Marisa., & Dellert, Johannes. (2016). *Ancestral state reconstruction and loanword detection*. Universitätsbibliothek Tübingen.
- Kondrak, Grzegorz. (2000). A new algorithm for the alignment of phonetic sequences. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference* (pp. 288-295). Association for Computational Linguistics.
- Kondrak, Grzegorz., & Hirst, Graeme. (2002). Algorithms for language reconstruction (Vol. 63, p. 5934). Toronto: University of Toronto.
- Levenshtein, Vladimir. (1965). Binary codes capable of correcting deletions, insertions and reversals. *Doklady Akademii Nauk sssr* 163: 845–848. In Russian.
- List, Johann-Mattis (2012) SCA: phonetic alignment based on sound classes. In: *New Directions in Logic, Language and Computation. Lecture Notes in Computer Science* 7415: 32-51. Springer: Berlin Heidelberg.
- List, Johann-Mattis. (2012b). Multiple sequence alignment in historical linguistics. *Proceedings of ConSOLE XIX*, 241, 260.
- List, Johann-Mattis and Forkel, Robert (2016): LingPy. A Python library for historical linguistics. Version 2.5. URL: <http://lingpy.org>, DOI: <https://zenodo.org/badge/latestdoi/5137/lingpy/lingpy>. With contributions by Steven Moran, Peter Bouda, Johannes Dellert, Taraka Rama, Frank Nagel, and Simon Greenhill. Jena: Max Planck Institute for the Science of Human History.
- Menecier, Philippe., Nerbonne, John., Heyer, Evelyne., & Manni, Franz. (2016). A Central Asian Language Survey. Collecting data, measuring relatedness and detecting loans. *Language Dynamics and Change*, 6(1), 57-98.
- Mi, Chenggang., Yang, Yating., Wang, Lei., Li, Xiao., & Dalielihan, Kamali. (2014). Detection of Loan Words in Uyghur Texts. In *Natural Language Processing and Chinese Computing* (pp. 103-112). Springer, Berlin, Heidelberg.
- Miao, Ruiqin. (2005). Loanword adaptation in Mandarin Chinese: Perceptual, phonological and sociolinguistic factors (Doctoral dissertation, Strony Brook University).
- Needleman, Saul., & Wunsch, Christian. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3), 443-453.

- Nerbonne, John., Heeringa, Wilbert., & Kleiweg, Peter. (1999). Edit distance and dialect proximity. *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison*, 15.
- Nerbonne, John., Heeringa, Wilbert., Van den Hout, Erick., Van der Kooi, Peter., Otten, Simone., & Van de Vis, Willem. (1996). Phonetic distance between Dutch dialects. In *CLIN VI: Proceedings of the sixth CLIN meeting* (pp. 185-202).
- Nerbonne, John., & Kleiweg, Peter. (2003). Lexical distance in LAMSAS. *Computers and the Humanities*, 37(3), 339-357.
- Nerbonne, John., & Kretzschmar, William. (2003). Introducing computational techniques in dialectometry. *Computers and the Humanities*, 37(3), 245-255.
- Paradis, Carole., & LaCharité, Darlene. (1997). Preservation and minimality in loanword adaptation. *Journal of linguistics*, 33(2), 379-430.
- Peperkamp, Sharon., & Dupoux, Emmanuel. (2003). Reinterpreting loanword adaptations: the role of perception. In *Proceedings of the 15th international congress of phonetic sciences* (Vol. 367, p. 370).
- Reetz, Henning., & Jongman, Allard. (2011). *Phonetics: Transcription, production, acoustics, and perception* (Vol. 34). John Wiley & Sons.
- Rietveld, Toni. and Van Heuven, Vincent. (1997). *Algemene fonetiek*. Coutinho, Bussum.
- Schroeder, Manfred., Atal, Bishnu., and Hall, J. L. (1979). Optimizing digital speech coders by exploiting masking properties of the human ear. *Journal of the Acoustical Society of America*, 66(6), 1647–1652.
- Swadesh, M. (1955). Towards greater accuracy in lexicostatistic dating. *International journal of American linguistics*, 21(2), 121-137.
- Swadesh, Morris. 1972. What is glottochronology? In Morris Swadesh (ed.), *The Origin and Diversification of Languages*, 271–284. London: Routledge & Kegan Paul.
- Trautmüller, Hartmut. (1990). Analytical expressions for the tonotopic sensory scale. *The Journal of the Acoustical Society of America*, 88(1), 97-100.
- Tsuchida, Ayako. (1995). English loans in Japanese: Constraints in loanword phonology. *Working Papers of the Cornell Phonetics Laboratory*, 10, 145-164.
- Tukey, John. (1977). *Exploratory data analysis*.
- Turchin, Peter., Peiros, Ilia., & Gell-Mann, Murray. (2010). Analyzing genetic connections between languages by matching consonant classes. *Journal of Language Relationship*, 3, 117-126.
- Wieling, Martijn, Bloem, Jelke, Mignella, Kaitlin, Timmermeister, Mona, & Nerbonne, John (2014) Measuring foreign accent strength in English. *Language Dynamics and Change*, 4(2), 253-269.
- Wieling, Martijn., & Nerbonne, John. (2015) Advances in dialectometry. *Annual Review of Linguistics* 1, 243–264.

- Wieling, Martijn, Prokić, Jelena, & Nerbonne, John (2009) Evaluating the pairwise string alignment of pronunciations. *Proceedings of the EACL 2009 workshop on language technology and resources for cultural heritage, social sciences, humanities, and education*. 26-34. Association for Computational Linguistics.
- Van Der Ark, René., Mennecier, Philippe., Nerbonne, John., & Manni, Franz. (2007). Preliminary identification of language groups and loan words in Central Asia. In *Proceedings of the RANLP Workshop on Computational Phonology* (pp. 12-20).
- Yip, Moira. (1993). Cantonese loanword phonology and Optimality Theory. *Journal of East Asian Linguistics*, 2(3), 261-291.

Appendix I

List of concepts from the extended Swadesh list in the data used in this dissertation.

one	rope	to think	rain	round
two	skin	to smell	river	sharp
three	meat	to fear	lake	dull
four	blood	to sleep	sea	smooth
five	bone	to live	salt	wet
big	fat	to die	stone	dry
long	egg	to kill	sand	correct
wide	horn	to fight	dust	near
thick	tail	to hunt	earth	far
heavy	feather	to hit	cloud	right
small	hair	to cut	fog	left
short	head	to pull	sky	name
narrow	ear	to scratch	wind	I
thin	eye	to dig	snow	you
woman	nose	to swim	ice	he
man	mouth	to fly	smoke	we
person	tooth	to walk	fire	you
child	tongue	to come	ashes	they
wife	nail	to lie	to burn	who ?
husband	leg	to sit	road	what ?
mother	knee	to stand	mountain	where ?
father	hand	to turn	red	when ?
animal	wing	to fall	green	how ?
fish	belly	to give	yellow	not
bird	neck	to hold	white	other
dog	back	to squeeze	black	
louse	breast	to wash	night	
butterfly	heart	to wipe	day	
snake	liver	to pull	year	
worm	to drink	to throw	autumn	
tree	to eat	to tie	warm	
forest	to bite	to sew	cold	
stick	to suck	to say	full	
fruit	to spit	to sing	new	
seed	to blow	to play	old	
leaf	to breathe	to freeze	good	
root	to laugh	sun	bad	
bark	to see	moon	rotten	
flower	to hear	star	dirty	
grass	to know	water	straight	