algorithms

MDPI

Article

On Hierarchical Text Language-Identification Algorithms

Maimaitiyiming Hasimu [1,2,3,*] and Wushour Silamu [1,2]

[1] School of Information Science and Engineering, Xinjiang University, Urumqi 830046, China; wushour@xju.edu.cn
[2] Key Multi-lingual Laboratory of Xinjiang, Urumqi 830046, China
[3] Department of Computer, Hotan Teachers College, Hotan 848000, China
* Correspondence: mamtimin116@163.com; Tel.: +86-991-858-2762

Received: 7 February 2018; Accepted: 23 March 2018; Published: 27 March 2018

Abstract: Text on the Internet is written in different languages and scripts that can be divided into different language groups. Most of the errors in language identification occur with similar languages. To improve the performance of short-text language identification, we propose four different levels of hierarchical language identification methods and conducted comparative tests in this paper. The efficiency of the algorithms was evaluated on sentences from 97 languages, and its macro-averaged F1-score reached in four-stage language identification was 0.9799. The experimental results verified that, after script identification, language group identification and similar language group identification, the performance of the language identification algorithm improved with each stage. Notably, the language identification accuracy between similar languages improved substantially. We also investigated how foreign content in a language affects language identification.

Keywords: language identification; character N-gram; script identification; language group identification; similar language identification

1. Introduction

Language identification (LI) is generally viewed as a form of text categorization. It is a process that attempts to classify text in a language into a pre-defined set of known languages [1]. LI is the first step in text mining, information retrieval, speech processing and machine translation [2–4]. Although LI is often considered a solved problem, studies have verified that LI accuracy rapidly drops when identifying short text [5–7], and confusion errors often occur between languages in the same family or in similar language groups [3,4,8]. Therefore, considerable room for improvement exists in terms of improving short-text LI performance and similar language identification performance.

Languages are written in different scripts, and each script has a unique defined code range in Unicode. This helps identify different parts of a script within a document [9]. Languages belong to different families, and language families can be divided into similar phylogenetic units [10]. A remarkably similar pattern is exhibited by languages within a phylogenetic unit [11], and this can be leveraged in LI to allow discrimination between different language groups written in the same script to narrow the range of identification.

In our earlier research [12], we presented tree-stage short-text language identification and evaluated the algorithm on 51 languages' sentences belonging to three different scripts. The experimental results verified that, after script identification and language group identification, LI accuracy improved.

Similar languages often reflect a common origin and are members of a dialect continuum. In addition, these languages share similar syntactic structures, so a strong lexical overlap exists between them [13]. Hence, LI confusion errors often occur with similar languages in the same language

group. Some studies have been completed on the identification of similar languages within similar languages and similar language groups within similar language groups. However, the identification of similar language in real LI systems has rarely been studied, and in our earlier research, we also did not analyze similar language identification.

Similar languages often mix with other languages by using the same script or same language group and share same the scripts, vocabulary and character N-grams. Distinguishing similar languages from each other requires a larger feature set than distinguishing languages without similar languages in LI. Similar language groups, because they have a highly similar structure, can be discriminated from other languages within the overall language group or same-script languages, and similar language group members can be identified within the similar language group. For LI systems to better handle similar languages, we propose two different hierarchical language identification algorithms to analyze similar language group identification using same-script languages and language groups and similar language identification within similar language groups.

In this paper, we present three new hierarchical LI algorithms. To analyze the advantages and disadvantages of the different levels of hierarchical LI systems, we completed comparative experiments on five different LI systems. In our earlier research, we did not provide hierarchical LI corpora's annotation process, did not complete a detailed analysis and interpretation of the algorithm principle and workflow and did not compare hierarchical LI to any other open source LI tools. With this paper, we intend to address these shortcomings. We also compare hierarchical methods to open-source LI langid.py in selecting 97 languages initially supported by langid.py.

In our earlier research, we selected character two-gram and three-gram as features and used a term-frequency-based feature selection and a weighting method commonly used in LI. In the subsequent research process, we found that, when increasing the number of languages in LI, the term-frequency-based feature selection and weighting method was not suitable for short-text LI. Moreover, language group identification accuracy also dropped, and character three-gram and two-gram are not suitable for LI for a larger number of languages. In addition, comparative research about feature selection in LI is lacking. For LI systems to identify language groups from a large number of languages and to further improve LI accuracy, we chose more advanced feature-selection methods, as well as a weighting method, and selected eight different types of character N-gram as features. We determined which feature selection, feature weighting and character N-grams are more suitable for language identification.

2. Related Work

Many LI studies' experimental results verified that LI accuracy drops when increasing the number of languages. Baldwin and Lui [7] compared naïve Bayes (NB), support vector machine (SVM) and three-distance-measure-based methods with datasets containing different sizes of documents and different numbers of languages. Their experimental results verified that the LI task becomes significantly more complex for shorter documents, larger numbers of languages, multilingual documents and higher-class skew. Abainia et al. [2] conducted comparative tests using 11 similarity measures combined with several types of character N-grams and proposed five high-frequency approaches. Their datasets included forum documents belonging to 32 languages, and their experimental results showed that LI is more accurate for small datasets containing fewer languages. Sibun and Reynar [14] used relative entropy to discriminate language similarity; their dataset included 27 languages written in the Roman script, and their experimental results verified that LI accuracy was higher when fewer languages were involved. Majlis [15] used five algorithms for LI tests on 30, 60 and 90 languages. The experimental results verified that increasing the number of languages reduced the accuracy of LI systems and increased prediction and training time.

Some researchers studied how to cluster natural languages, but their methods cannot be applied to LI. Gamallo et al. [16] defined two quantitative distances to measure the distances between two languages in a set of 44 languages. They found that groups of languages with short distances between

them tend to form a language family or sub-family. Souter et al. [17] explored the correlation between the frequency of bigrams and trigrams for nine European languages, and their results adhered to the Indo-European family tree predicted by historical linguists. Damashek [18] selected character five-grams, measured similarities among documents in 31 languages and revealed a similarity-based clustering of languages. Damashek's experimental results led to a conclusion that accurate family groups can be identified by grouping languages with similar scores. However, only the impressive performance of their system in language discrimination was verified; no LI-related data were provided. Goldhahn and Quasthoff [19] selected character trigrams and the most frequently-used words to cluster 108 languages and reported that the former yielded better results, showing that the clustering results identified the genealogical relations among languages.

Since 2014, the Discrimination between Similar Languages (DSL) shared task has been organized every year, providing researchers an opportunity to evaluate their LI systems in discriminating between similar languages, varieties and dialects. In the four DSL shared tasks, SVM outperformed the other classification methods in most situations [20–23]. Although some researchers attempted to use deep-learning-based approaches [24–26], most performed poorly compared to traditional classifiers [22,23].

Every DSL shared task provided the DSL dataset, in which the similar language groups were manually selected and implemented similar language identification (SLI) [20–23]. Some researchers studied similar language group identification methods in DSL shared tasks. The proposed similar language group prediction algorithm predicted the similar language group to which a given language belonged and then discriminated among languages within the group. Goutte et al. [27] proposed a language group classifier using character four-grams as features that exhibited excellent performance. However, its prediction time was longer in tests than in training. Porta and Sancho [28] used a simple token-based feature and maximum-entropy classifier to predict language groups. Lui et al., Fabra-Boluda et al. and Ács et al. used two-stage similar language identification approaches, as well [29–31]. However, the above five research efforts did not provide any testing data with which to evaluate the efficiency of their similar language group identification methods. The above-mentioned similar language group identification approaches cannot be directly applied to LI tasks because similar languages are often presented as being mixed with other languages belonging to the same language group. Therefore, we first had to find the similar language group amongst the other languages and then identify languages within similar language group.

Goutte et al. investigated the progress made between the 2014 and 2015 DSL shared tasks, estimated an upper bound on possible performance using ensemble and oracle combination and analyzed the learning curves for both similar language group prediction and similar language identification [32]. Their experimental results verified that similar language group prediction reaches perfect performance using relatively few examples, and almost all of the errors are always within a group of similar languages or variants.

Each script in Unicode has its own code range, and this advantage was previously used [9] to traverse every letter in a text to find the starting and ending code points in a given script, detect different scripts portions in the text and distinguish the language used in each. However, the authors [9] analyzed all the languages in the system when predicting the language rather than those using the same script.

3. Proposed Method

3.1. LI Corpora

We used the Leipzig Corpora Collection, Project Gutenberg, the European legislature datasets and discriminative similar language (DSL) Corpus Collections to create the LI corpora [20–23,33]. Among them, the Leipzig Corpora Collection has the most languages. To evaluate our proposed LI algorithm, we compared it to other LI open-source projects, such as Polyglot [34] and langid.py [35].

During evaluation, we used the same language corpora for both our algorithm and the open-source LI project. Polyglot can identify more than 196 languages, but we were unable to access all the language sources in Polyglot from currently available corpora sets. langid.py was pre-trained on 97 languages, all of which can be accessed in the Leipzig Corpora Collection. langid.py is superior to most other open-source LI tools in terms of short-text LI accuracy [36]. For these reasons, we selected the Leipzig Corpora Collection to create our LI corpora and selected langid.py as the baseline approach.

Another reason for choosing the Leipzig Corpora Collection was that our research objective was to identify short text language. The Leipzig Corpora Collection contains randomly selected sentences in the language of the corpus. The sources were newspapers or text randomly collected from the web and split into sentences [37,38]. In our experiment, we used sentences from the Leipzig Corpora Collection to train, test and evaluate the LI algorithm. The minimum length for most languages' sentences in our test was approximately 20 characters.

We used ISO 639-2/T codes to annotate language [39]. In this coding scheme, each language is represented by a three-letter code. Node that "Code" in the tables is the language ISO code. The 97 languages initially supported by langid.py belong to 25 writing scripts. Among them, 20 scripts have only one member (Table 1). The other five writing scripts—Arabic, Cyrillic, Devanagari, Latin and Eastern Nagari—have many languages (Tables 2–5). According to language family knowledge, the Arabic-, Cyrillic- and Latin-script members in the corpora belong to different language groups (Tables 2–4). Some similar language groups also exist in the Latin-script datasets. In Table 2, the languages in parentheses represent the similar language group within the relevant language group. In our test, we selected the 10K-format dataset from the Leipzig Corpora Collection, in which each language has 10,000 sentences. Of the 97 languages used in langid.py, 93 can be directly downloaded from the Leipzig Corpora Collection's website. For the other four languages, we contacted the paper's [37] authors, who provided the remaining languages.

Table 1. Script datasets with only one member.

Script	Code	Script	Code	Script	Code	Script	Code
Armenian	hye	Japanese Braille	jpn	Ge'ez	amh	Georgian	kat
Greek	ell	Gujarati	guj	Gurmukhi	pan	Hebrew	heb
Kannada	kan	Khmer	khm	Korean	kor	Tamil	tam
Lao	lao	Odia	ori	Sinhala	sin	Telugu	tel
Thai	tha	Simplified Chinese	zho	Tibetan	dzo	Tirhuta	mal

Table 2. Latin script dataset.

Language Group	ISO Language Code List
Afro-Asiatic/Semitic	Mlt
Austroasiatic/Vietic	vie
Austronesian/Malayo-Polynesian (MP)	(Ind, msa), jav, mlg, tgl
constructed language	vol
International auxiliary language	epo
Indo-European/Albanian	sqi
Indo-European/Balto-Slavic	(bos, hrv), ces, lav, lit, pol, slk, slv
Indo-European/Celtic	bre, cym, gle
Indo-European/Germanic	afr, dan, deu, eng, fao, isl, ltz, nld, nno, (nob, nor), swe
Indo-European/Italic	arg, cat, fra, glg, hat, ita, lat, oci, por, ron, spa, wln
Language isolate (Vasconic)	eus
Niger–Congo/Atlantic–Congo	swa, xho, zul, kin
Quechuan languages/Quechua	que
Altaic/Turkic	azj, tur
Uralic/Finnic	est, fin
Uralic/Finno-Ugric	hun
Uralic/Sami	sme

Table 3. Arabic script dataset.

Language Group	ISO Code List	Language Group	ISO Code
Afro-Asiatic/Semitic	ara	Altaic/Turkic	uig
Indo-European/Indo-Iranian	fas, kur, pus, urd		

Table 4. Cyrillic script dataset.

Language Group	ISO Code
Indo-European/Balto-Slavic	bel, bul, mkd, srp, rus, ukr
Altaic/Mongolic	mon
Altaic/Turkic	kaz, kir

Table 5. Nagari and Devanagari script dataset.

Script	Language Group	ISO Language Code List
Devanagari	Indo-European/Indo-Iranian	hin, mar, nep
Eastern Nagari	Indo-European/Indo-Iranian	asm, ben

3.2. Script Identification

Each script in Unicode has its own defined code range, and this facilitates the detection of different parts of a script in text. We propose a regular-expression matching-based script identification (SI) algorithm (REMSI). Based on the code range of the scripts in Unicode, we created a regular expression for every script. The proposed SI stage consists of the following steps:

Step 1. Identify the character encoding format. If it is not UTF-8, convert it to UTF-8.

Step 2. Remove items that are not alphabets, spaces and characters from the sentence.

Step 3. Use regular expressions to match the sentence to each script's regular expressions to judge whether relevant script contents are in the sentence.

Step 4. Calculate the length of each matching result, and if the length is nonzero, save it in a list. Sort the list by decreasing length.

Step 5. Select the top item on the list as the main script of the text. If a script only has one member, return the language; otherwise, return its script, and further identify its language within languages using the same script.

To effectively use a sentence's script in hierarchical LI, we designed our algorithm to return the language's ISO code when a script has only one member language; otherwise, the script name is returned, and the sentence language is further identified within same-script languages. In the pseudocode of this paper, if the script identification's result length is equal to three, the language has been determined and the language returned.

3.3. Two-Stage LI

Different scripts in Unicode have different code ranges. We can use this advantage to identify a text's script. Although some languages are written by multiple scripts that have different Unicode code ranges, they share common scripts or words with other languages using the same script. Text script is easier to identify than its language. Hence, analyzing all languages in LI is unnecessary when identifying a language. We used this advantage to design two-stage LI. The script of a given text is identified in the first stage. If the SI's result is a language ISO code, return language; otherwise, the language of the text is identified within the same script languages in the second stage. When annotating this method's corpora, we use each sentence's language-relevant ISO code.

The pseudocode of the algorithm is shown in Figure 1. In the pseudocode of this paper, if the script identification's result is a three-letter language ISO code, it indicates that the script has only one member and return language. The eighth line creates a vector based on the LI feature list. Sections 4.6 and 5 discuss how to determine the LI task's feature list. The same design was used in the algorithms, which are introduced below.

```
Two_stage_LI(sentence)
1    script← Script_Identification(sentence)
3    if length(script)==3
4      return language
5    else
7      ngram_list← Extract_N_gram(sentence, LI_in_script's_feature_type)
8      vector← CreateVector (ngram_list, LI_in_script's_feature_list)
9      language← LI_in_Script(vector)
10     return language
```

Figure 1. Pseudocode of two-stage language identification (LI).

3.4. Three-Stage LI-1

The majority of confusion errors in LI occur between similar languages. Similar languages have similar syntactic structures and strong lexical overlap between. This advantage can be used to discriminate similar language groups (SLGs) within the same script languages. Distinguishing similar languages from each other requires a larger feature set than distinguishing languages when no similar languages exist in LI. More features are needed to improve the recognition efficiency of similar languages, but more memory and more time will be required compared with using a small number of features. Additionally, the identification accuracy of other languages is almost unchanged.

To improve similar language identification efficiency, we implemented similar language group dentification (SLGI) in script and designed a three-stage LI-1. Two kinds of three-level hierarchical LI exist: three-stage LI-1 and three-stage LI-2. The script of a given text is identified in the first stage. If the SI's return is a language ISO, return language. Otherwise, the language of the text is identified within the same script languages in the second stage. If the second stage's return value is a language ISO code, it means that the sentence's language is identified, and return language. Otherwise, the return value is a similar language group, and we further identify the language within a similar language group in the third step, and return language. When annotating the second stage's corpora, if the sentence is a member of a similar language group, we used a similar language group name to annotate it. Otherwise, we used its language ISO code. For example, when annotating the Latin script language sentences in Table 2, we use "eng" for the English language sentences and "nor-nob" to annotate the Norwegian and Bokmål language sentences, because Norwegian and Bokmål are highly similar languages. For the third stage's corpora, we used the sentence language ISO code to annotate the sentences. The pseudocode of three-stage LI-1 is shown in Figure 2.

```
Three_stage_LI_1(sentence)
1      script← ScriptIdentification(sentence)
2      if length(script)==3
3        return language
4      else
5        ngram_list← ExtractN_gram(sentence, SLGI's_feature_type)
6        vector← CreateVector (ngram_list, SLGI's_feature_list)
7        SLG= SLGI(vector)
8        if length (SLG)==3
9          return language
10       else
11         vector← CreateVector (ngram_list, LI_in_SLG's_feature_list)
12         Language← LI_in_SLG(vector)
13         return language
```

Figure 2. Pseudocode of three-stage LI-1.

3.5. Three-Stage LI-2

Languages in the same phylogenetic unit are similar in terms of vocabulary and structure. This advantage can be used to discriminate between different language groups (LGs) among languages using the same script. This helps reduce the number of languages in LI and should improve the LI accuracy. In our earlier research, this inspired us to implement language group identification (LGI) in LI, and we designed a three-stage hierarchical LI system called three-stage LI-2. It includes SI and LGI within languages using the same script and language identification within the language group (LI in LG). Second-stage LGI returns the language group name or language ISO code. If the language ISO code is returned, the LI has identified and returned the sentence's language. If it returns the language group name, we further identified the sentence language within the language group. When annotating the LGI corpora, if the LG has only one language, we annotated this language group's sentences with its ISO code. Otherwise, we annotated the sentence with the relevant language group name. For example, when annotating Cyrillic script LGI corpora in Table 4, three targets were found: Balto-Slavic, Common-Turkic and "mon." Because the Altaic/Mongolic language group has only one member in our test, we used its language ISO code "mon" to annotate its language group. For the last stage's corpora, we used the sentence language ISO code to annotate the sentences. The pseudocode of the three-stage LI-2 is shown in Figure 3.

```
Three_stage_LI_1(sentence)
1     script← ScriptIdentification(sentence)
2     if length(script)==3
3       return language
4     else
5       ngram_list← ExtractN_gram(sentence, LGI's_feature_type)
6       vector← CreateVector (ngram_list, SLGI's_feature_list)
7       LG= LGI(vector)
8       if length (LG)==3
9         return language
10      else
11        vector← CreateVector (ngram_list, LI_in_LG's_feature_list)
12        Language← LI_in_LG(vector)
13        return language
```

Figure 3. Pseudocode of three-stage LI-1.

3.6. Four-Stage LI

In three-stage LI-2, if the LG contains a similar language group, a high number of confusion errors still exist between similar languages in LI in LG. This phenomenon inspired us to explore the possibility of first identifying similar language groups within a language group and then identifying similar languages within a similar language group. Therefore, we improved three-stage LI-2 to four-stage LI. The pseudocode of four-stage LI is shown in Figure 4.

Four-stage LI includes SI and LGI within languages using the same script, language identification within the language group (LI in LG) and language identification in similar language groups (LI in SLG). Third-stage LI in LG returns the similar language group (SLG) name or language ISO code. If the language ISO code is returned, the LI has identified and returned the sentence's language. If it returns the SLG name, we further identified the sentence language within the SLG. When annotating this stage's corpora, if the sentence language did not belong to any SLG, we used the language ISO code to annotate it; otherwise, we used the SLG name. For example, when annotating LI in LG corpora for the Austronesian/Malayo-Polynesian (MP) language group in Table 2, four targets were returned: "ind-msa," "jav," "mlg" and "tgl." The first target represents a SLG that includes Indonesian and Malayan languages.

```
Four_stage_LI(sentence)
1    script← ScriptIdentification(sentence)
2    if length(script)==3
3      return language
4    else
5      ngram_list← ExtractN_gram(sentence, LGI_in_script's_feature_type)
6      vector← CreateVector (ngram_list, LGI_in_script's_feature_list)
7      LG= LGI(vector)
8      if length (LG)==3
9        return language
10     else
11       vector← CreateVector (ngram_list, SLGI_in_LG's_feature_list)
12       SLG← SLGI_in_LG(vector)
13       if length(SLG)==3
14         return language
15       else
16         vector← CreateVector(ngram_list, LI_in_SLG's_feature_list)
17         language← LI_in_SLG(vector)
18         return language
```

Figure 4. Pseudocode of four-stage LI.

3.7. Toy Example

To illustrate the hierarchical LI process, we chose a toy example for four-stage LI. Table 6 shows four sentences in four different languages. Sentence A is a Korean sentence, and during the SI process, A's script was identified as "kor". Because the Korean script in our corpora has only one member, the LI system returns A's language as "kor". For sentence B, its script can be identified in the first step as Latin. Because the Latin script in our corpora has many language groups, further identification is needed to determine the language group in the Latin script to which it belongs. In the second step, its language group can be identified as "hun," because the Uralic/Finno-Ugric language group in our experiments has only one member. For C and D, the SI also returns Latin, but the second step returns Germanic because the Germanic language group has many members. In the third step, to further identify the language, whether the language is a certain language or a member of a similar language group is determined. Sentence D can be identified as nor-nob, a group whose members consist of the Norwegian and Bokmål languages. Next, D must be further identified within the similar language group and was identified as nor. For sentence C, in the third step, its language can be identified as "eng" because English does not belong to any similar language groups in our test.

Table 6. Toy example sentences. SLG, similar language group.

ID	Content	Language	Script	LG	SLG
A	저는 유학생입니다	Korean	Korean		
B	Egy nemzetközi diák vagyok.	Hungarian	Latin	Finno-Ugric	
C	I'm an international student.	English	Latin	Germanic	
D	Jeg er en internasjonal student.	Norwegian	Latin	Germanic	nor-nob

4. Experimental Setup

4.1. Preprocessing

Two kinds of preprocessing were used in our experiments, as follows:

Preprocessing in SI: When identifying a sentence script, its encoding format was first identified. If its encoding format was not UTF-8, we changed it to UTF-8. Except for alphabets, spaces and characters, items were then removed from the sentence; multiple spaces were replaced with a single space; and spaces at the beginning and end of the sentence were removed. The return value of this preprocessing step is the input for the next preprocessing step.

Preprocessing in LI in SI, LGI, SLGI, LI in LG and LI in SLG: The preprocessing for each of the above classification tasks was basically the same, addressing languages using the same script. In the preprocessing step, contents containing other scripts were removed from the sentences.

4.2. Character N-Gram

In LI, statistical models can be generated using the number of words [4] or letters in the given text [2], N-gram statistics [3,8] or a combination of the two [2]. The dominant statistical approach used in the literature is the character-based N-gram model, which is superior to the word-based model for small text fragments and performs equally well on large fragments. It is also tolerant of errors in text, requires no prior linguistic knowledge, is highly accurate and easily creates and computes any given text. Hence, most LI systems use character N-grams [2,3,8]. We therefore restricted the feature sets we used to character N-grams.

An N-gram is a sequence of n consecutive letters. The N-gram-based approach for LI divides the text into character strings of equal size [5]. Some languages are assumed to use certain N-grams more frequently than others. An example of the decomposition of the sentence "good boy" into character N-grams is presented in Table 7. The symbol "-" represents a space, which is used to capture the start and end of words.

Table 7. Example of the decomposition of a sentence into character N-grams.

N-Gram Type	N-Gram
1-gram	g, o, o, d, b, o, y
2-gram	-g, go, oo, od, d-, -b, bo, oy, y-
3-gram	-go, goo, ood, od-, d-b, -bo, -boy, oy-
4-gram	-goo, good, ood-, od-b, d-bo, -boy, boy-

4.3. Feature Selection

In this paper, four feature-selection methods were applied. The brief preliminary notations are outlined in Table 8. The feature-selection methods are as follows:

Chi-squared (*CHI*) is a well-known feature-selection method used in pattern recognition and measures the correlation between the term t_i and the category C_j [40], expressed as:

$$\chi^2(t_i, C_j) = \frac{N(ad - bc)}{(a+b)(a+c)(b+d)(c+d)}, \quad (1)$$

$$CHI(t_i) = \max_{j \in (1,M)} \chi^2(t_i, C_j), \quad (2)$$

where a, b, c, d and M are explained in detail in Table 8.

The distinguishing feature selector (*DFS*) is a novel filter-based probabilistic feature-selection method [41]. It measures whether a term frequently occurs in only one class of the highest importance in discriminating different classes. *DFS* also determines whether a term frequently occurs in some classes, is relatively discriminative and considers the term irrelevant in other situations. It can be defined as:

$$DFS(t_i) = \sum_{i=1}^{M} \frac{P(C_j|t_i)}{P(\overline{t_i}|C_i) + P(t_i|C_j) + 1}, \quad (3)$$

where $P(C_j|t_i)$, $P(\overline{t_i}|C_i)$, $P(t_i|C_j)$ and M are explained in detail in Table 8.

The normalized different measure (*NDM*) is a modified balanced accuracy measure (ACC2) [42]. *NDM*'s feature selection concept is that an important term that is in the document's frequency in the

positive class (tp) or negative class (fp) should be closer to zero, along with a high $|tpr - fpr|$ value, which is relatively more important. It can be expressed as:

$$NDM = \frac{|tpr - fpr|}{min(tpr, fpr)}, \tag{4}$$

where tpr and fpr are explained in detail in Table 8.

The odds ratio (OR) reflects the odds of the item occurring in the positive class normalized by that of the negative class [43]. OR is defined as:

$$OR(t_i, C_j) = \frac{P(t_i|C_j)\left(1 - P(t_i|\overline{C_j})\right)}{\left(1 - P(t_i|C_j)\right)P(t_i|\overline{C_j})} \tag{5}$$

$$Feature(t_i) = \max_{j \in (1,M)} OR(t_i, C_j) \tag{6}$$

where $P(t_i|C_j)$ and $P(t_i|\overline{C_j})$ are explained in detail in Table 8.

Table 8. Preliminary notations.

Notation	Value	Meaning
M		M is the number of classes
a	$count(t_i, C_j)$	Number of documents belonging to class C_j and containing term t_i.
b	$count(t_i, \overline{C_j})$	Number of documents not belonging to class C_j and containing term t_i.
c	$count(\overline{t_i}, C_j)$	Number of documents belonging to class C_j and not containing term t_i.
d	$count(\overline{t_i}, \overline{C_j})$	Number of documents not belonging to class C_j and not containing term t_i.
N	$(a + b + c + d)$	Total number of documents in the training corpora.
$P(C_i\|t)$	$a/(a + b)$	The probability of class C_j when word t_i is present.
$P(\overline{t}\|C_i)$	$c/(a + c)$	The probability of other items $\overline{t_i}$ when class C_j is present.
$P(t\|\overline{C_j})$ *or* fpr	$b/(b + d)$	The probability of item t_i when other class $\overline{C_j}$ is present.
$p(t_i\|C_J)$ *or* tpr	$a/(a + c)$	The probability of item t_i when other class C_j is present.

4.4. Document Representation

After selecting the feature subsets, all sentences were represented by a feature vector with the term frequency-inverse document frequency ($TFIDF$) weighting function [44]. The weight of the term t_i in sentence d_j is calculated by the following formula:

$$TFIDF(t_i, d_j) = \frac{tf(t_i, d_j) log \frac{N}{n(t_i)}}{\sqrt{\sum_{k=1}^{M} \left\{ tf(t_k, d_j) log \frac{N}{n(t_i)} \right\}^2}}, \tag{7}$$

where $tf(t_i, d_j)$ denotes the number of times t_i occurs in sentence d_j, $n(t_i)$ is the number of sentences in which t_i occurs at least once, N is the total number of sentences in the training corpus and M is the size of the feature subset.

4.5. Method Classification and Evaluation

We used two classifiers for the experiments: SVM and Bayes NB [45]. The reason for this selection is that the SVM outperformed the other classifiers in four DSL shared tasks, and our baseline approach, langid.py, was implemented in NB. In our experiments, for NB, we selected scikit-learn's multinomial NB and used Lidstone smoothing $\alpha = 0.05$. For the SVM, we selected a linear kernel and used parameters C = 1 and class_weighted = balanced.

The standard success measure method, macro-averaged F1-score, was used in this study to measure the performance of the TLI tasks. In macro-averaging, the macro F-measure is computed for

each class within the dataset, and then, the average for all classes is obtained [46]. Using this method, equal weight is assigned to each class regardless of class frequency. Macro-F1 can be formulated as:

$$Macro - F1 = \frac{\sum_{k=1}^{C} F_k}{C}, F_k = \frac{2P_k R_k}{P_k + R_k} \quad (8)$$

$$P_k = \frac{TP_k}{TP_k + FN_k}, \; R_k = \frac{TP_k}{TP_k + FP_k} \quad (9)$$

where C is the number of existing classes, TP_k is the number of correctly classified documents for class C_k, FP_k is the number of incorrectly classified documents for class C_k, and FN_k is the number of incorrectly classified documents relative to other classes.

4.6. Comparative Experiment Design

To evaluate our proposed algorithm's efficiency, we performed seven group tests: (1) regular-expression matching-based SI versus SI proposed in Hanif et al. [9], (2) one-stage LI, (3) two-stage LI, (4) three-stage LI-1, (5) three-stage LI-2, (6) four-stage LI and (7) the effect of foreign-language content in the LI corpus on the LI.

During the tests, we found that some languages, especially the Xhosa language, include a significant amount of foreign language. This creates noise during the feature selection and classification processes, and its sentences are often misclassified into other languages that do not belong to the same LG. Therefore, to investigate the effects of noisy language content on LI performance, we performed comparative tests using 96 languages (Sections 5.3–5.7). We then completed comparative tests using 97 languages (Section 5.8).

We split our corpora into training, testing and evaluation. Among these, the testing corpora included 1000 sentences for every language, and 10-fold cross-validation was used to evaluate the performance of the different levels of hierarchical LI. In each classification task, different feature selection methods were used to select the feature sets in different N-gram feature types, and each feature set used a different feature range for training and testing classifiers. Finally, we compared the different classifiers' macro F1-scores in different feature sizes and selected the optimal classification model (OCM) for each task. The OCM is a classification model whose accuracy does not improve or minimally improves when more features are added, even when the feature size increases considerably. The OCM was selected to perform comparative tests for different levels of hierarchical LI and LI in langid.py tests.

5. Results and Discussion

5.1. Performance Evaluation of Script-Identification Algorithm

To evaluate our proposed algorithm, we used REMSI and SI to identify each sentence's script in our evaluation corpora, for a total of 96,250 sentences in 97 languages. The macro-averaged F1-scores for both methods were the same, and the identification accuracy was very high. Most errors originated from Latin-script languages, especially English, which is used in other scripts' sentences or other scripts' contents in Latin-script sentences. REMSI is superior in execution time to SI. The results for SI are provided in Table 9. In this paper, the time unit format is hour:minute:second.millisecond.

Table 9. Script identification results.

REMSI	Score	Time	SI	Macro F1	Time
F1 score	0.9986	0.978	0.00:58.26	0.9981	0.03:53.50

5.2. N-Gram Distribution in Different Corpora

To evaluate the effect of the number of languages on LI, we calculated different N-gram numbers in different LI corpora. The results are given in Table 10, and we verified that fewer different N-gram numbers were found in shorter N-gram feature types than in longer N-gram feature types. From the results, we observed that more different N-grams exist in a corpus containing more languages than in a corpus containing fewer languages. Thus, narrowing the LI identification range can be beneficial for feature selection, noise reduction and training, thus improving LI accuracy.

Table 10. Different N-gram numbers in different LI corpora.

ID	In All 97 Languages	In Latin Script	In Germanic LG	In Nor-Nob SLG
2-gram	256,629	7389	3203	1706
3-gram	973,319	78,320	31,516	13,781
4-gram	2,342,861	438,945	164,640	57,012
5-gram	4,174,524	1,391,955	445,814	114,227

5.3. One-Stage LI

One-stage LI is an original LI. SI, LGI and SLGI do not occur during the LI process. The text's language is directly identified from all languages in LI. The results for one-stage LI, when a *CHI* feature-selection method was used, are shown in Table 11, and the confusion matrix and F1-scores related to the similar languages are shown in Table 12. Note that the number after the classification name is the relevant feature type used in the classification and ''FS'' in the tables is the feature size. From the results, we conclude that a mixture of N-grams is suitable for one-stage LI, and SVM's accuracy was higher than NB. SVM was more accurate than NB for most LI tasks in our experiment. Due to space limitations, we only provide LI results related to SVM. The selected OCM for one-stage LI is when use SVM, use a mixture of three-grams and two-grams and feature size is 15,000. We used the selected OCM to perform one-stage LI testing. The results are given in Table 24. From the confusion errors in Table 12, we determined that most errors in LI occur between similar languages.

Table 11. Results of LI in one-stage LI. FS, feature size.

FS	1000	3000	5000	10,000	15,000	20,000	25,000	30,000
NB-2	0.440	0.870	0.920	0.942	0.952	0.952	0.952	0.957
SVM-2	0.423	0.879	0.933	0.955	0.964	0.964	0.964	0.964
NB-3	0.590	0.763	0.874	0.916	0.932	0.935	0.936	0.950
SVM-3	0.588	0.768	0.886	0.932	0.949	0.952	0.952	0.963
NB-4	0.446	0.667	0.776	0.864	0.878	0.899	0.905	0.914
SVM-4	0.448	0.672	0.781	0.878	0.892	0.913	0.919	0.928
NB-5	0.334	0.560	0.670	0.775	0.800	0.822	0.840	0.851
SVM-5	0.339	0.566	0.676	0.786	0.813	0.836	0.855	0.866
NB-30	0.690	0.834	0.920	0.945	0.950	0.952	0.952	0.963
SVM-30	0.697	0.845	0.937	0.953	0.968	0.970	0.970	0.970
NB-40	0.692	0.815	0.899	0.937	0.947	0.949	0.949	0.961
SVM-40	0.698	0.829	0.918	0.957	0.966	0.968	0.968	0.969

Table 12. Confusion matrix and F1-scores related to similar languages in one-stage LI.

ID	Bos	Hrv	ID	Ind	Msa	ID	Nob	Nor
bos	768	207	ind	714	262	nob	555	322
hrv	205	710	msa	229	756	nor	407	428
F1-score	0.771	0.735		0.772	0.744		0.544	0.468

5.4. Two-Stage LI

The languages in our corpus belong to 25 different scripts. Among them, 20 scripts only have one member. Thus, the language can be identified during the SI process. The other five scripts have many members. After script identification, their members are identified within the same script languages.

The pseudocode is shown in Figure 1. The classification results for LI in the same script languages are shown in Figures 5–10 and Table 13. After examining the results in the figures, we found that the shorter character N-gram requires fewer features to reach its optimal value than longer character N-grams, and combinations of different types of N-grams are more efficient than single-length N-grams. Two types of mixed N-grams exist in our tests: a mixture of three-grams and two-grams and a mixture of two-grams, three-grams and four-grams. The two types have nearly the same efficiency, which is better than that of the other single-type N-grams in some LI tasks. This is because when using a hybrid N-gram model, we can use different lengths of prefixes, suffixes and roots to create the basic unit of words. The above situation also occurs with the other LI tasks.

Table 13. Results of LI in non-Latin script languages when using SVM, distinguishing feature *DFS* and 2-gram.

FS	300	500	700	1000	1500	2000
Arabic	0.997	0.998	0.998	0.998	0.998	0.998
Cyrillic	0.978	0.988	0.990	0.991	0.992	0.992
Devanagari	0.992	0.992	0.993	0.993	0.993	0.993
Nagari	0.999	0.999	0.999	0.999	0.999	0.999

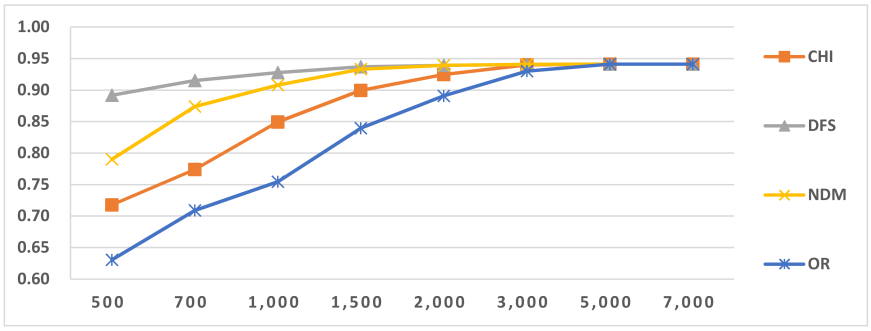

Figure 5. Results of LI in Latin script languages when selecting SVM and two-gram.

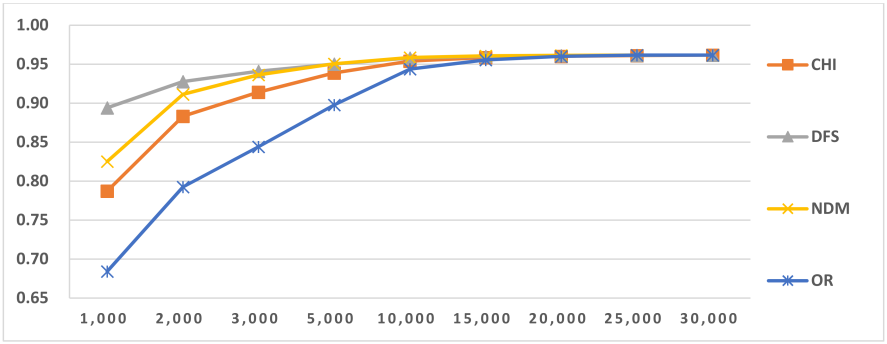

Figure 6. Results of LI in Latin script languages when selecting SVM and three-gram.

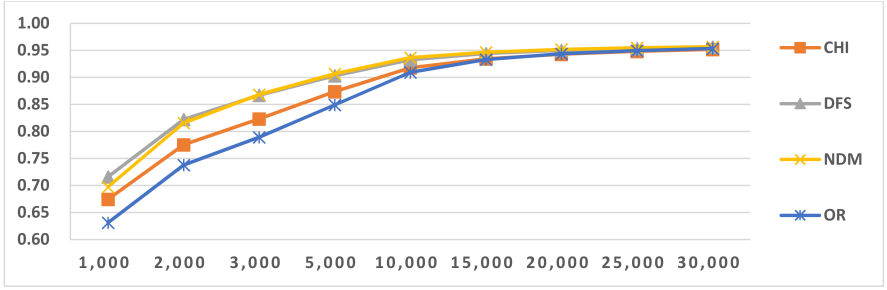

Figure 7. Results of LI in Latin script languages when selecting SVM and four-gram.

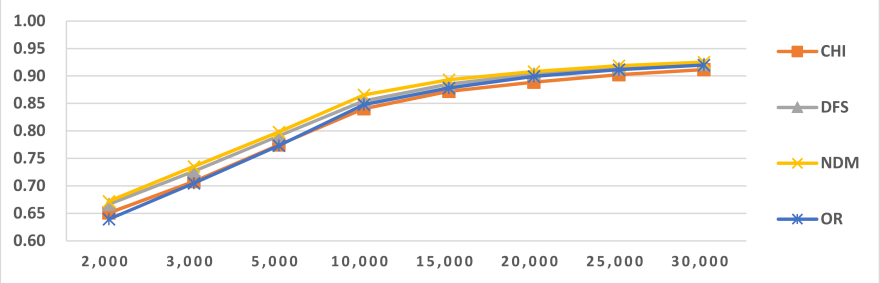

Figure 8. Results of LI in Latin script languages when select SVM and five-gram.

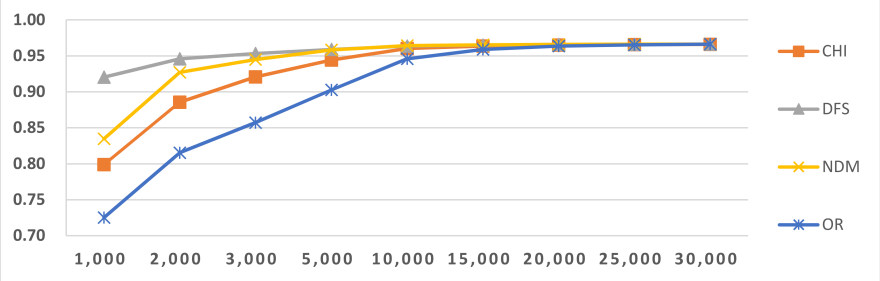

Figure 9. Results of LI in Latin script languages when selecting SVM and mix-gram (2–3).

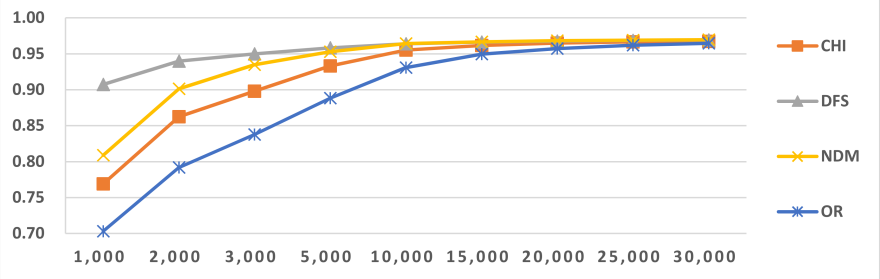

Figure 10. Results of LI in Latin mix-gram (2–4) Latin script languages.

For longer feature types, such as four-gram and five-gram, accuracy continued to increase, as shown in Figures 5–10 and Table 11. This reveals that if longer feature types are selected for LI, more features are needed to reach optimal accuracy than for shorter N-gram feature types. This is because when longer N-gram feature types are selected, a larger number of different N-gram numbers exist than in shorter feature types. Thus, longer N-gram feature types require longer feature sizes to cover enough distinct features. The above situation also occurred in the other LI tasks.

The LI results in Table 13 reveal that only using character two-grams to reach high classification accuracy is possible with fewer languages and no similar languages within the same-script language group, for example, in LI of Arabic, Cyrillic, Devanagari and Nagari scripts. A similar situation occurred in other tasks in the remaining experiments. The reason for this result is that several hundreds or thousands of two-grams that can cover the majority of the two-grams in the language group contain few languages. Moreover, little lexical overlap exists between the languages because no similar languages were in the group. In this situation, there are enough distinct features to identify the language. Therefore, after SI, we did not need a large number of features for language groups that contain fewer languages and do not contain similar languages.

In this paper, we used four feature-selection methods, *CHI*, *NDM*, *DFS* and OR, to perform comparative tests. From the results in Figures 5–10, we found that the efficiencies of *DFS* and *NDM* are better than the other two feature selection methods. The reason for this result is that they prefer to

select the features frequently occurring in single classes and do not select features that frequently occur in multiple classes [38,39]. Their advantages mean these methods are suitable for LI. We also concluded that for classifiers that use the same feature type, but with different feature selection methods, as the number of features increases to a certain value, their LI values are the same and close to the maximum value. The same situation appeared in other LI tasks. Due to space limitation, the following analysis only provides the LI results when using *DFS*.

The LI classification in Latin script was lower than for other scripts. Because it has more languages than the others and some highly similar language groups are found within Latin-script languages, confusion errors occur between similar languages. After examining the classification results, we selected the OCM for two-stage LI (Table 14). We used the selected OCMs to perform two-stage LI testing. The results are given in Table 24. From the comparative results, we concluded that the accuracy of the two-stage LI is higher than that of one-stage LI and that the two-stage LI execution time is shorter than that of one-stage LI. Because the use of SI can narrow the LI algorithm's range, it helps to reduce noisy features, feature selection and training. The confusion matrix related to similar languages is shown in Table 15. From the confusion matrix results in Tables 12 and 15, we concluded that SI similar language identification accuracy significantly increased.

Table 14. Optimal classification method (OCM) for LI in different scripts in two-stage LI.

ID	Model	FS	F1-Score	ID	Model	FS	F1-Score
Arabic	SVM-2	500	0.998	Cyrillic	SVM-2	1500	0.992
Devanagari	SVM-2	700	0.993	Latin	SVM-(2–3)	10,000	0.965
Nagari	SVM-2	300	0.999				

Table 15. Confusion matrix and F1-scores related to similar languages in two-stage LI.

ID	Bos	Hrv	ID	Ind	Msa	ID	Nob	Nor
bos	758	236	ind	770	218	nob	528	414
hrv	52	940	msa	167	828	nor	330	588
F1-score	0.836	0.842		0.784	0.801		0.562	0.578

5.5. Three-Stage LI-1

Three-stage LI-1 includes three sub-tasks: SI, LI in script and LI in similar language group (SLG); the pseudocode is shown in Figure 2. In second-stage LI in script, the process text language is identified as either a specific language or a member of a similar language group. If the second stage's return is a similar language group, the text's language is further identified within the similar language group. In our experiments, three highly similar language groups were placed in the Latin-script languages (Table 2), and no highly similar languages in other scripts' languages were in our test. Their LI in script is the same as two-stage's LI in script. Their results are shown in Table 13, and the selected OCM is given in Table 14. The LI in Latin script results are shown in Figure 11. For LI in Latin script, the result's macro F1-score was 0.989. This reveals that similar language groups can be accurately identified from the other languages using the same script, and similar language group identification does not affect other language identification.

The results for LI in SLG are shown in Figures 12–14. Notably, the last feature numbers in these three figures are the total feature numbers in the task in our experiment. The results verify that, to obtain high LI accuracy for similar languages, more features are needed than other languages in the same script. The reason for this is the high similarity in structure and high lexical overlap in similar languages. If we increase the feature size in two-stage LI, more storage and time were needed to predict languages. However, LI accuracy was minimally improved or not at all. Therefore, to improve similar language LI accuracy, we first identified the similar language group and then identified the languages within a similar language group. From the LI in SLG results, we also found that when feature size increased up to a certain point, LI's accuracy reached an optimal value, and after that,

accuracy decreased. This implies that there is still room for improvement in feature selection and weighting for similar language identification. We also found that the performance of NB in LI in SLG was better than SVM.

The selected OCM for LI in Latin script and SLI in three SLG in three-stage LI-1 are shown Table 16. We used the selected OCM to perform three-stage LI-1 testing. From the results given in Table 24, we found that three-stage LI-1's accuracy was higher than that of two-stage LI. The reason for this result is that we accurately identified similar language groups from other languages, and their identification did not affect other language identification. In addition, similar languages obtained enough features in LI in a similar language group. However, the time efficiency of this method was slightly lower than two-stage LI. This is because identifying similar languages requires more features than other languages, and similar language group identification also consumes time. The confusion matrix related to similar languages is shown in Table 17. Compared with the confusion matrix results in Tables 12, 15 and 17, we concluded that, after SI and similar language group identification (SLGI), the similar language identification accuracy significantly increased.

Table 16. Selected OCM for LI in Latin script and SLI in three SLG in three-stage LI-1.

ID	Model	FS	F1-Score	ID	Model	FS	F1-Score
LI in Latin	SVM-(2–3)	10,000	0.989	LI in bos-hrv	NB-5	30,000	0.936
LI in ind-msa	NB-5	20,000	0.930	LI in nob-nor	NB-5	20,000	0.794

Table 17. Confusion matrix and F1-score related to similar languages in three-stage LI-1.

ID	Bos	Hrv	ID	Ind	Msa	ID	Nob	Nor
bos	914	81	ind	889	101	nob	595	367
hrv	145	837	msa	120	877	nor	407	549
F1 score	0.886	0.867		0.876	0.886		0.583	0.562

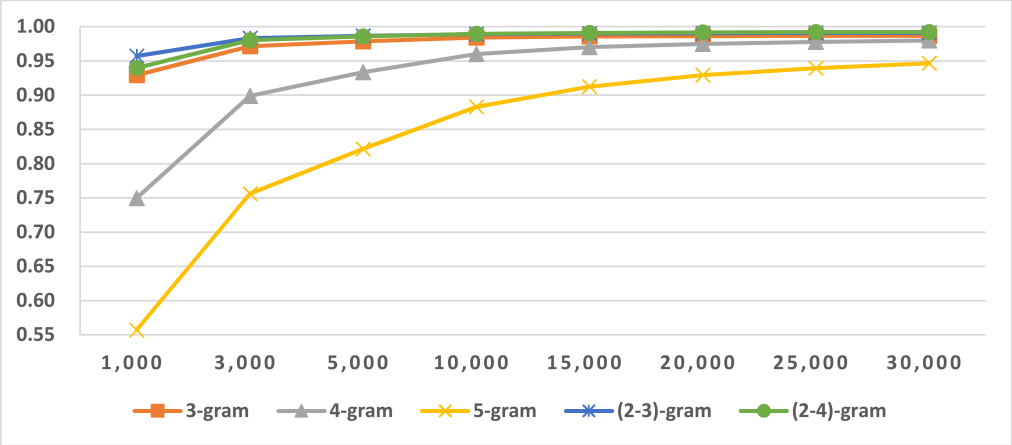

Figure 11. Results of LI (SLGI) in Latin script languages.

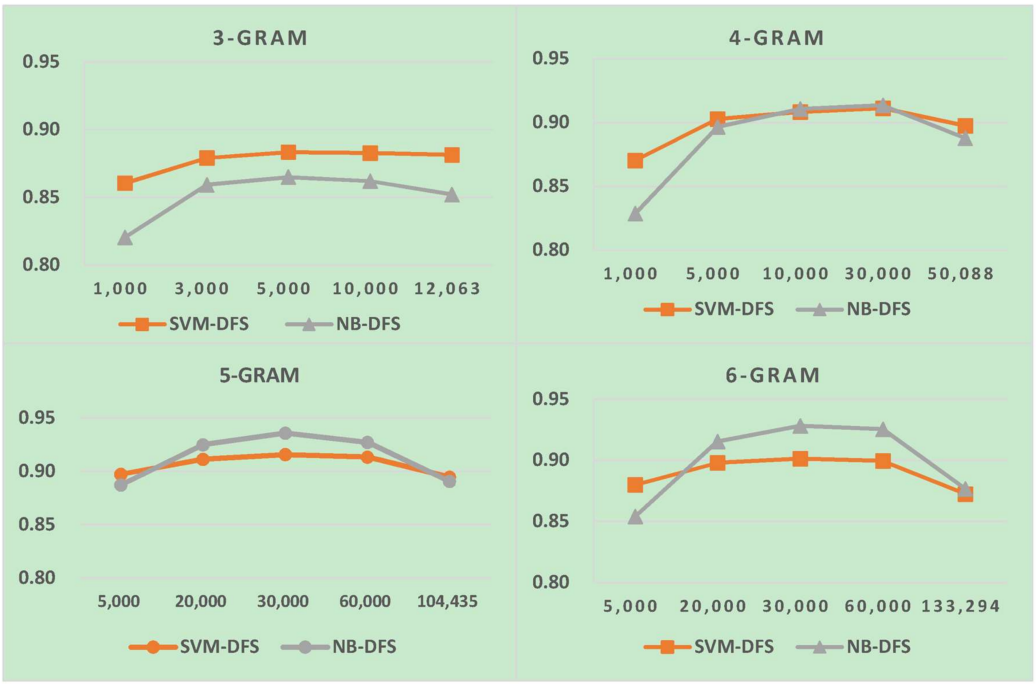

Figure 12. LI in similar language groups (SLG) including bos and hrv.

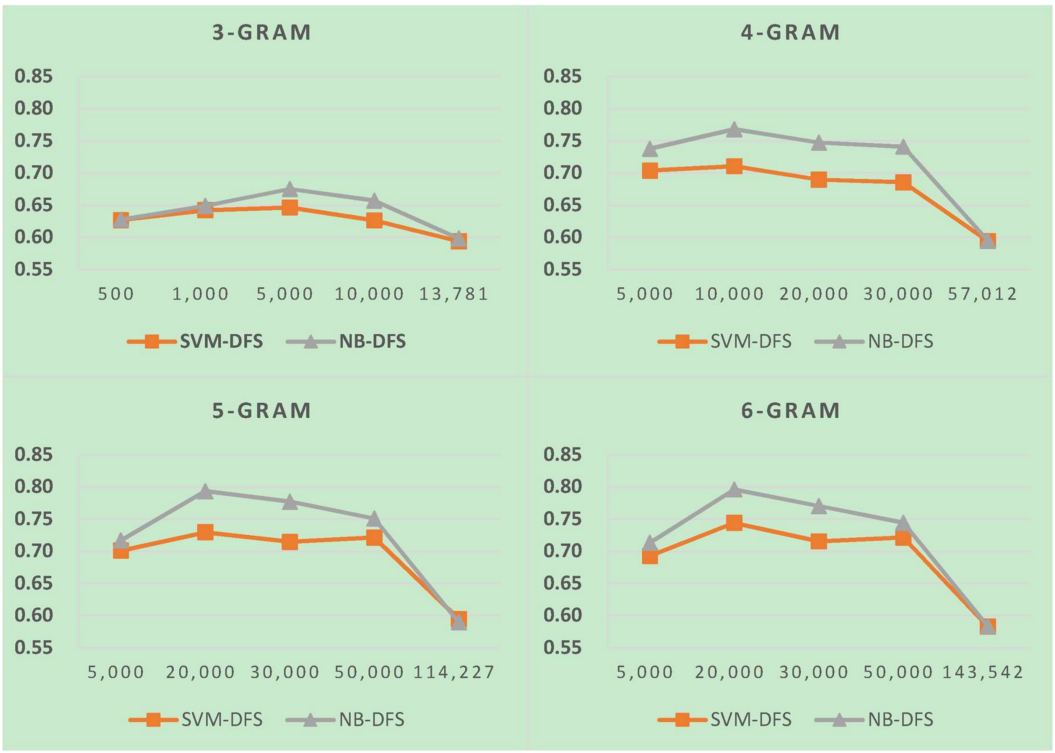

Figure 13. LI in SLG consisting of ind and msa.

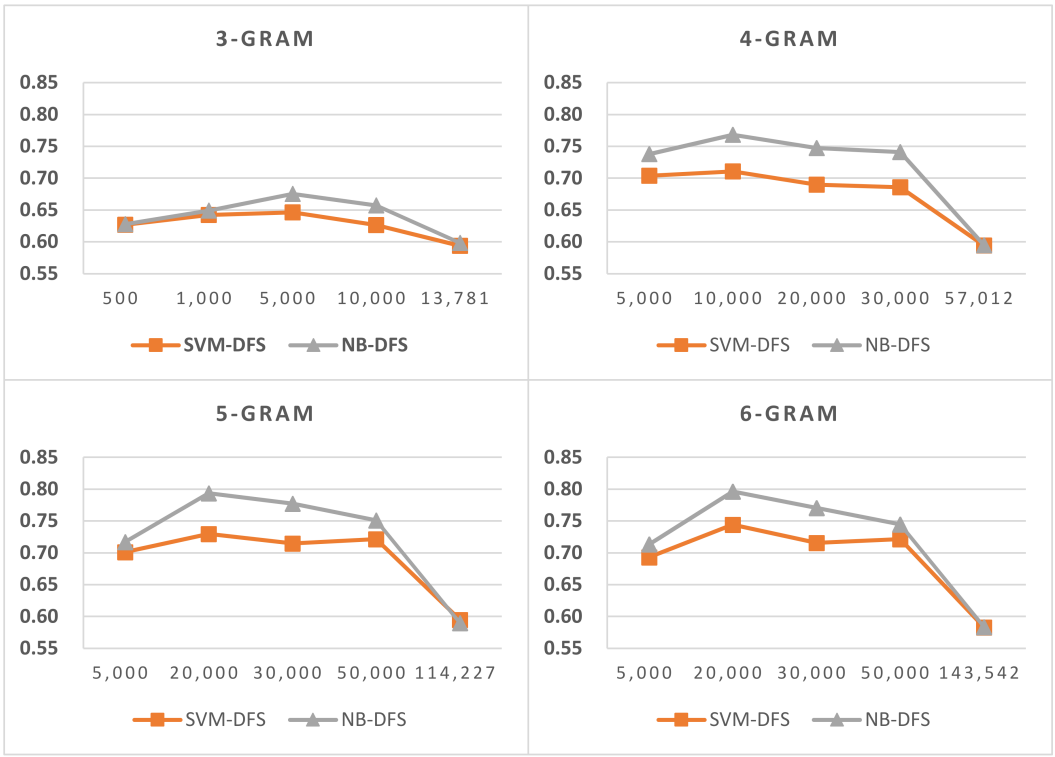

Figure 14. LI in SLG consisting of nob and nor.

5.6. Three-Stage LI-2

Three-stage LI-2 includes three sub-tasks: SI, LGI in using same script languages and LI in LG; the pseudocode is shown in Figure 3. In the LGI process, the language group to which the text's language belongs is determined using the same script languages. In the third stage, the language within the language group is identified. In our experiments, Arabic-, Cyrillic- and Latin-script languages were divided into several language groups, as shown in Tables 2–4. The Devanagari and Nagari scripts also have many members (Table 5), but their respective members only belong to one language group. Identifying their members after SI is equivalent to the LGI process; the results are shown in Table 13, and their selected OCM is shown in Table 14.

The results of LGI on Arabic, Cyrillic and Latin scripts are shown in Figure 15 and Table 18. From the results, we found that if there are fewer members in the script groups, using only two-gram features is sufficient to reach the optimal LGI score. If more languages are used in the scripts, three-gram or mixtures of N-grams are more suitable for LGI. LGI's accuracy is very high, revealing that we can discriminate between different LG in the same script languages. The results for LI in LG are shown in Figures 16–19 and Table 19. Comparing the LI in LG results to the LI in script results, we concluded that the majority of the LI in LG's feature size, which reached optimal accuracy, is lower than the LI in script's feature size. It also revealed that narrowing the identification range is beneficial to language identification. From the results in Table 19, we also concluded that if there are fewer members or no similar language group in LG, using only two-gram features is sufficient to reach the optimal LI score. The boldface or underscore numbers in Table 19 highlights the LG that includes similar languages or has more members. Therefore, during identification, their members need more features and longer feature types than other LGs.

Table 18. LGI in different script languages when using *DFS* and two-gram.

FS	300	500	700	1000	2000	3000	5000	7000
Arabic	0.999	0.999	0.999	0.999	0.999	0.999		
Cyrillic	0.998	0.999	0.999	0.999	0.999			
Latin	0.960	0.980	0.986	0.989	0.991	0.992	0.992	0.992

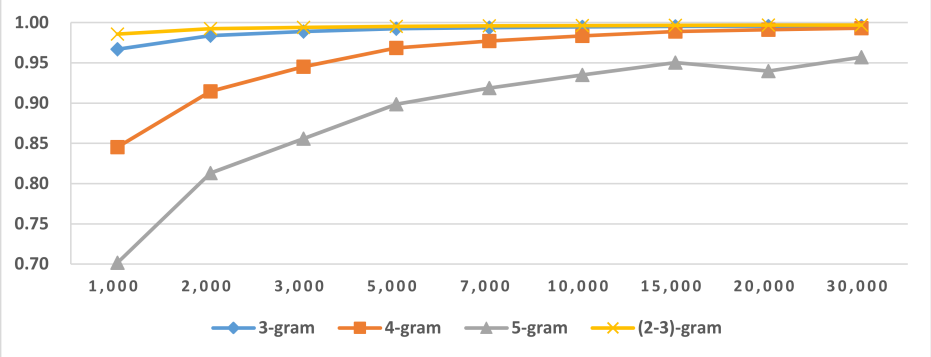

Figure 15. LGI in Latin script languages.

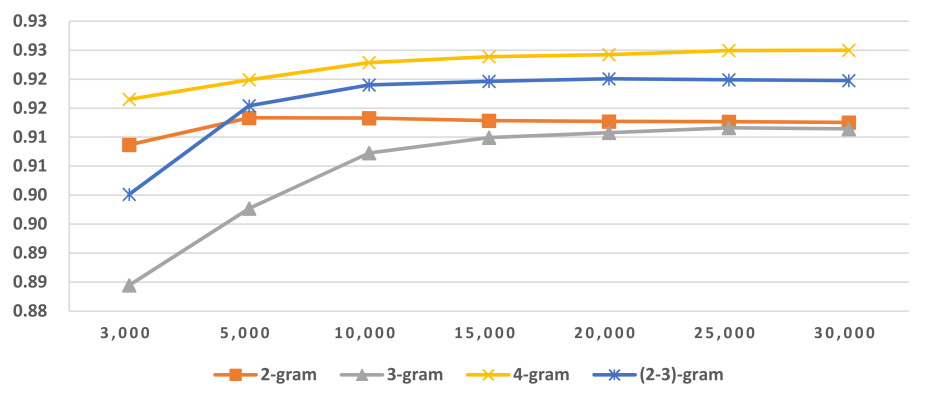

Figure 16. LI in Indo-European/Germanic in Latin script languages.

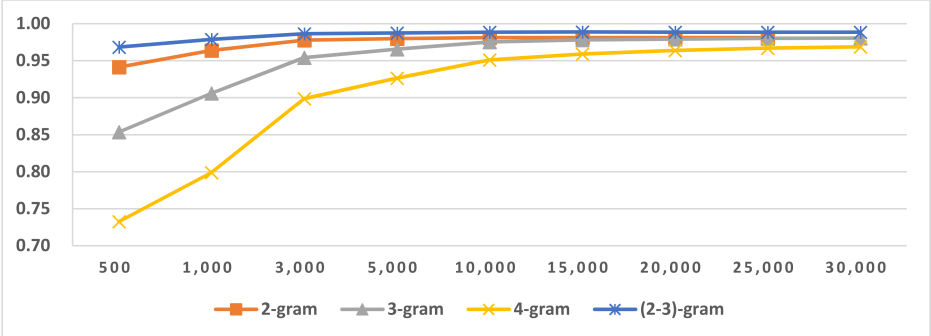

Figure 17. LI in Indo-European/Italic in Latin script languages.

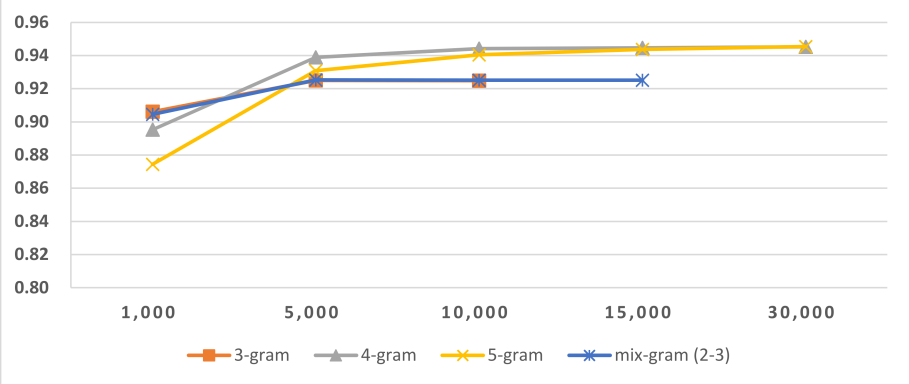

Figure 18. LI in Austronesian/Malayo-Polynesian (MP) in Latin script languages.

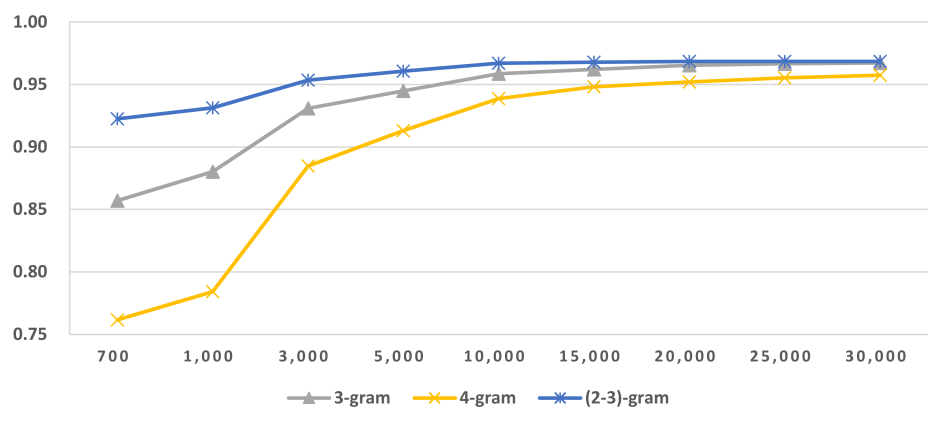

Figure 19. LI in Indo-European/Balto-Slavic in Latin script languages.

The selected OCM for LGI is given in Table 20. The three-stage LI-2 result is shown in Table 24, and from the results, we concluded that the three-stage LI-2's macro F1-score is higher than the of two-stage LI. The reason for this result is that we accurately identified language groups by using similar script languages and identification language within a language group. Narrowing the identification range is beneficial to LI, improving the LI's accuracy. The confusion matrix related to similar languages is shown in Table 21. Comparing the results in Tables 15 and 21, similar language identification accuracy in three-stage LI-2 is higher than that of two-stage LI. The reason for this finding is that, after LGI narrowing of the identification range, there are more than enough features to discriminate similar languages.

Although the accuracy of three-stage LI-2's is higher than that of two-stage LI, the majority of confusion errors still occur between similar languages. The prediction time was slower than two-stage LI's because the LGI process also consumes time.

Table 19. LI in LG when using two-gram.

	FS	100	300	500	700	1000	3000
Arabic	Indo-European/Indo-Iranian	0.993	0.997	0.998	0.998	0.998	
Cyrillic	Altaic/Turkic	0.993	0.997	0.998	0.998	0.998	
	Indo-European/Balto-Slavic	0.946	0.982	0.986	0.989	0.989	
Latin	Niger-Congo/Atlantic-Congo	0.996	0.999	0.999	0.999	0.999	
	Indo-European/Celtic	0.997	0.999	0.999	0.999	0.999	
	Altaic/Turkic	0.977	0.981	0.982	0.981	0.981	
	Uralic/Finnic	0.996	0.997	0.998	0.998	0.998	
	Indo-European/Germanic	**0.778**	**0.849**	**0.872**	**0.878**	**0.883**	**0.882**
	Indo-European/Italic	0.883	0.953	0.965	0.969	0.971	0.971
	Austronesian/Malayo-Polynesian	**0.832**	**0.865**	**0.871**	**0.872**	**0.872**	
	Indo-European/Balto-Slavic	**0.795**	**0.888**	**0.912**	**0.920**	**0.927**	**0.935**

Table 20. Selected OCM for LGI and LI in LG in three-stage LI-2.

ID	Model	FS	F1-score
LGI in Arabic script languages	SVM-2	300	0.999
LGI in Cyrillic script languages	SVM-2	500	0.999
LGI in Latin-script languages	SVM-(2–3)	5000	0.995
LI in Indo-European/Indo-Iranian in Arabic script	SVM-2	500	0.998
LI in Indo-European/Balto-Slavic in Cyrillic script	SVM-(2–3)	5000	0.995
LI in Turkic-Common/Turkic in Cyrillic in Cyrillic	SVM-2	300	0.999
LI in Indo-European/Germanic in Latin	SVM-(2–3)	15,000	0.924
LI in Indo-European/Italic in Latin	SVM-(2–3)	10,000	0.989
LI in Niger-Congo/Atlantic-Congo in Latin	SVM-2	300	0.999
LI in Austronesian/Malayo-Polynesian (MP) in Latin	SVM-(2–3)	10,000	0.945
LI Indo-European/Celtic in Latin	SVM-2	300	0.999
LI in Turkic-Common/Turkic in Latin	SVM-3	3000	0.983
LI in Indo-European/Balto-Slavic in Latin	SVM-(2–3)	10,000	0.966
LI in Uralic/Finnic Latin	SVM-3	500	0.999

Table 21. Confusion matrix and F1-score related to similar languages in three-stage LI-1.

ID	Bos	Hrv	ID	Ind	Msa	ID	Nob	Nor
bos	739	227	ind	852	144	nob	572	404
hrv	35	957	msa	131	865	nor	380	598
F1-score	0.851	0.863		0.850	0.857		0.580	0.589

5.7. Four-Stage LI

Four-stage LI includes four sub-tasks: SI, LGI in same script languages, LI in LG and LI in SLG; the pseudocode is shown in Figure 4. In third-stage LI in LG, a sentence language identifies whether a certain language or a member of a similar language group is in an LG. A member of a similar language group further identifies its language within a similar language group. In our experiments, three highly similar language groups exist in three language groups in Latin-script languages (Table 2). The SLGLI in LG results are shown in Figures 20–22. Other scripts' language groups do not contain similar language groups; their LI in LG result is the same as that of three-stage LI-2. Four-stage LI's second-stage result is the same as that of three-stage, and its fourth-stage result is the same as three-stage LI-1's third-stage result.

From the SLGLI in LG results in Figures 20–22, we found that a similar language group can be accurately identified within a language group. The selected OCM is shown in Table 22. We used the selected OCM for four-stage LI and performed four-stage LI testing. The result is shown in Table 24, which shows that its macro F1-score is the same as that of three-stage LI-2, and the prediction time is slower than that of three-stage LI-2. This is because if a language belongs to a similar language group, we need an LI in the SLG process, which also consumes time. The confusion matrix related to the similar language is shown in Table 23. From the confusion-matrix results in Tables 21 and 23, we concluded that, after LGI and LI in SLG, similar language identification accuracy also improved.

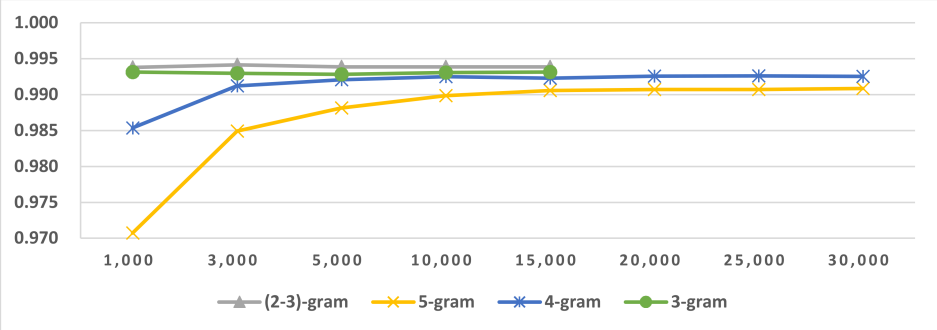

Figure 20. LI in Austronesian-Malayo/Polynesian (MP) in Latin.

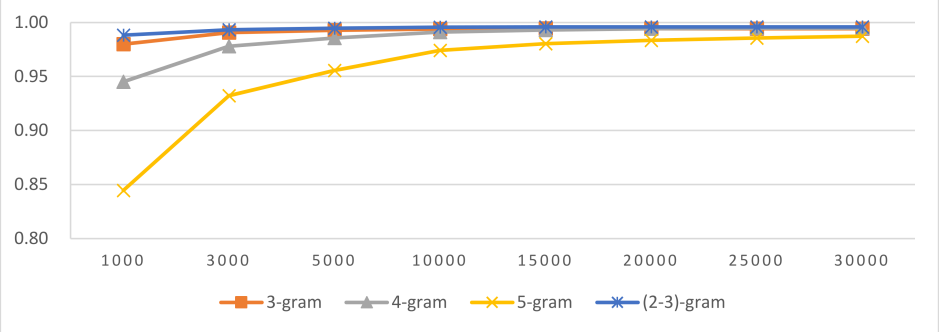

Figure 21. LI in Indo-European/Balto-Slavic in Latin.

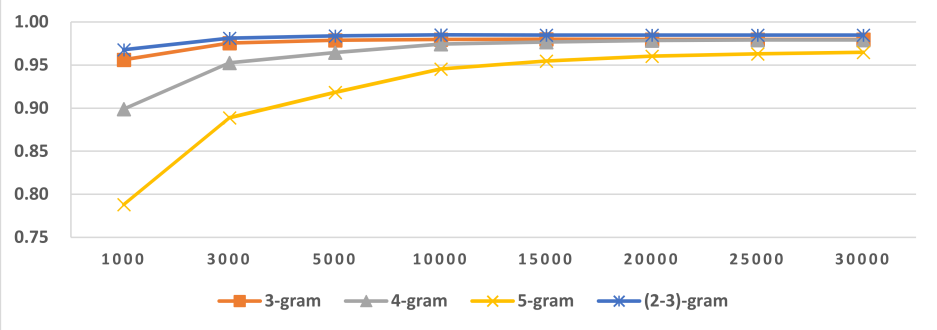

Figure 22. LI in Indo-European/Germanic in Latin.

To evaluate our proposed methods' efficiency, we compared it to the open source LI tool langid.py. The comparison results are shown in Table 24. From the results, we concluded that our proposed hierarchical LI's efficiency is higher than that of langid.py, demonstrating that our proposed method is suitable for short text LI.

To evaluate how different levels of hierarchical language identification affect LI, we used Indo-European/Germanic group languages' F1-scores with different LI methods in (Table 25). Comparing the different level hierarchical LI results in Tables 24 and 25, we concluded that, after using SI, LGI or SLGI to narrow the LI's identification range, LI's accuracy improved each stage. The improvement in similar language identification accuracy was especially significant.

Table 22. Selected OCM for SLGI in three LG in four-stage LI.

ID	Model	FS	F1-Score
LI in Austronesian-Malayo/Polynesian (MP) in Latin	SVM-(2–3)	1000	0.994
LI in Indo-European/Balto-Slavic in Latin	SVM-(2–3)	5000	0.995
LI in Indo-European/Germanic in Latin	SVM-(2–3)	5000	0.984

Table 23. Confusion matrix between similar languages in LI in LG.

ID	Bos	Hrv	ID	Ind	Msa	ID	Nob	Nor
bos	915	81	ind	894	102	nob	609	376
hrv	145	847	msa	120	874	nor	414	568
F1 score	0.885	0.877		0.879	0.883		0.594	0.576

Table 24. Comparison results for different levels of hierarchical LI.

Type	One-Stage	Two-Stage	Three-Stage 1	Three-Stage 2	Four-Stage	Langid.py
Macro-F1	0.9659	0.9757	0.9792	0.9797	0.9799	0.8766
Time	0:00:44.41	0:00:42.70	0:00:47.64	0:01:15.81	0:01:18.97	0:01:18.97

Table 25. F1-score for Indo-European/Germanic languages in Latin for different levels of hierarchical LI.

ID	Afr	Dan	Deu	Eng	Fao	Isl	Ltz	Nld	Nno	Nob	Nor
One-stage	0.989	0.908	0.983	0.962	0.969	0.973	0.980	0.978	0.872	0.543	0.468
Two-stage	0.995	0.965	0.989	0.980	0.988	0.989	0.982	0.991	0.934	0.562	0.577
Three-stage 1	0.994	0.968	0.990	0.981	0.990	0.992	0.986	0.990	0.946	0.583	0.561
Three-stage 2	0.993	0.983	0.992	0.977	0.992	0.996	0.886	0.994	0.973	0.579	0.588
Four-stage	0.994	0.980	0.991	0.977	0.994	0.997	0.985	0.995	0.974	0.594	0.576
langid.py	0.893	0.868	0.916	0.900	0.693	0.811	0.926	0.904	0.811	0.217	0.552

5.8. Foreign Language Content in LI Corpora Effect on LI

Foreign language items within the LI corpora affect the LI training and prediction processes and reduce accuracy. During the experiment, we found that the Xhosa (xho) language, which belongs to the Niger-Congo/Atlantic-Congo group, was often misclassified into other languages. A similar situation also occurred in the langid.py test. In our research, Xhosa corpora has 10,000 sentences. According to the experimental test, seven of the sentences belonged to other scripts. Sentence length statistics are given in Table 26. We added this language to perform comparative tests. The results are presented in Table 27. The confusion matrix related to Xhosa in two-stage LI is given in Table 28.

From the confusion matrix, we see that Xhosa sentences are often predicted as Afrikaans (afr), which belongs to the Germanic language group and not to the Atlantic-Congo group. Theoretically, the probability of Xhosa's sentences being misclassified into the same language group's language is greater than being misclassified into another language group's language. However, in our tests, 90 Xhosa sentences were misclassified into Afrikaans, which belongs to another language group, and 25 sentences were misclassified into Zulu (zul), which belongs to the same language group. A total of 996 of the 1000 Afrikaans sentences were correctly classified as Afrikaans when LI did not include Xhosa. Additionally, 948 of the 1000 Afrikaans sentences were correctly classified to Afrikaans when LI included Xhosa. The results verified that when foreign content is included in the corpora, LI accuracy drops because foreign content creates noise during the feature selection and classification processes.

Table 26. Xhosa corpora statistics.

Length Unit	Max	Min	Average
Character	260	19	112.14
Word	61	2	15.78

Table 27. Comparative results of the effect of noisy content on LI.

Test Type	One-Stage	Two-Stage	Three-Stage 1	Three-Stage 2	Four-Stage	Langid.py
Macro-F1	0.9620	0.9726	0.9748	0.9753	0.9754	0.8736
Time	0:00:45.167	0:00:44.537	0:00:44.376	0:01:19.102	0:01:20.765	0:01:20.960

Table 28. Confusion matrix related to Xhosa.

Including Xhosa						Not Including Xhosa				
	afr	kin	swa	xho	zul		afr	kin	swa	zul
afr	948			47		afr	996			
kin		996		2		kin		998		
swa			996			swa			998	
xho	90			839	25	zul				1000
zul				16	984					
Macro-F1	0.927	0.997	0.996	0.861	0.977		0.995	0.998	0.998	0.999

6. Conclusions

Languages are written using different scripts and belong to different language groups. Languages in the same language group are similar in terms of vocabulary and structure, and the majority of errors occur between similar languages in LI. Considering these facts, we presented three different hierarchical LI algorithms in this paper. To evaluate the proposed algorithms, we performed comparative tests on different levels of hierarchical LI algorithms and the open source LI tool, langid.py. The experimental results verified that, after SI, LGI and SLGI, the accuracy improved with each stage. This proves that narrowing the LI range can improve accuracy, especially between similar languages. Our proposed method's efficiency is superior to the open source LI tool langid.py.

We used different-length N-gram feature types to perform comparative LI tests. The experimental results verified that if there are fewer languages or no similar language groups, using only a shorter N-gram feature type and a relatively small feature size can result in optimal LI accuracy. When mixing the different N-gram feature types to select a feature, the accuracy was superior to that of single N-gram feature types for most of the LI tasks. Further research is required to determine how to extract different length character sequences that can represent prefixes, suffixes, roots and stems in a language and how to apply them to the field of LI.

In our experiments, we selected NB and SVM classifiers to perform the LI tasks. From the experimental results, we concluded that for the SLI in SLG task, SVM's classification accuracy was higher than NB. We also used the four feature-selection methods, *CHI*, *NDM*, *DFS* and OR, to perform comparative tests. From the experimental results, we concluded that *DFS* and *NDM*'s efficiencies are better than the other two feature selection methods. The experimental results verified that the feature-selection and feature-weighting methods used are suitable for short-text language identification.

LGI and SLGI are beneficial for narrowing the LI range and increasing LI's accuracy. However, LGI and SLGI data are highly imbalanced data, which negatively impact feature selection. Thus, LGI and SLGI efficiency can still be improved. If we improve the LGI and SLGI efficiency, we can select relatively small features for discriminating different LG or SLG when there are more languages in the same script, which can further improve LI efficiency.

We also investigated how foreign language content in a particular language corpus affects the LI. The experimental results showed that accuracy was reduced. A method for automatically removing foreign language content from the LI corpora requires further investigation.

Acknowledgments: This work was partially supported by the National Program on Key Basic Research Projects of China (Grant No. 2014CB340506) and the Key Project of the National Natural Science Foundation of China (Grant No. U1603262). The authors thank Thomas Eckart and Uwe Quasthoff at Leipzig University, who provided the language corpora.

Author Contributions: Maimaitiyiming Hasimu designed the algorithm, analyzed the data and wrote the paper. Wushour Silamu provided guidance for the algorithm design and writing.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Choong, C.Y.; Mikami, Y.; Marasinghe, C.A.; Nandasara, S.T. Optimizing n-gram order of an n-gram based language identification algorithm for 68 written languages. *Int. J. Adv. ICT Emerg. Reg.* **2009**, *2*, 21–28. [CrossRef]
2. Abainia, K.; Ouamour, S.; Sayoud, H. Effective language identification of forum texts based on statistical approaches. *Inf. Process. Manag. Int. J.* **2016**, *52*, 491–512. [CrossRef]
3. Botha, G.R.; Barnard, E. Factors that affect the accuracy of text-based language identification. *Comput. Speech Lang.* **2012**, *26*, 307–320. [CrossRef]
4. Selamat, A.; Akosu, N. Word-length algorithm for language identification of under-resourced languages. *J. King Saud Univ.-Comput. Inf. Sci.* **2015**, *28*, 457–469. [CrossRef]
5. Cavnar, W.B.; Trenkle, J.M. N-Gram-Based Text Categorization. In Proceedings of the 3rd Annual Symposium on Document Analysis and Information Retrieval (SDAIR-94), Las Vegas, NV, USA, 11–13 April 1994; pp. 161–175.
6. Singh, A.K. Study Some Distance Measures for Language and Encoding Identification. In Proceedings of the Workshop on Linguistic Distance, Sydney, Australia, 23 July 2006; pp. 63–72.
7. Baldwin, T.; Lui, M. Language Identfication: The Long and the Short of the Matter. In Proceedings of the Human Language Technologies: The 2010 Annual Conference of the North American of the ACL, Los Angeles, CA, USA, 2–4 June 2010; pp. 229–237.
8. Brown, R.D. Finding and identifying text in 900+ languages. *Digit. Investig.* **2012**, *9*, 534–543. [CrossRef]
9. Hanif, F.; Latif, F.; Khiyal, M.S.H. Unicode Aided Language Identification across Multiple Scripts and Heterogeneous Data. *Inf. Technol. J.* **2007**, *6*, 534–540.
10. Rowe, B.M.; Levine, D.P. *A Concise Introduction to Linguistics*; Retrieved 26 January 2017; Routledge: Abingdon, UK; pp. 340–341. ISBN 1317349288.
11. Henn, B.M.; Cavalli-Sforza, L.L.; Feldman, M.W. The great human expansion. *Proc. Natl. Acad. Sci. USA* **2017**, *109*, 17758–17764. [CrossRef] [PubMed]
12. Hasimu, M.; Silamu, W. Tree-stage Short Text Language Identification Algorithm. *J. Digit. Lang. Identif.* **2017**, *15*, 354–371.
13. Željko, A. Slovene-Croatian Treebank Transfer Using Bilingual Lexicon Improves Croatian Dependency Parsing. In Proceedings of the 15th International Multiconference Information Society, Ljubljana, Slovenija, 8–12 October 2012; pp. 5–9.
14. Sibun, P.; Reynar, J.C. Languge identification: Examining the issues. In Proceedings of the 5th Symposium on Document Analysis and Information Retrieval, Las Vegas, NV, USA, 15–17 April 1996; pp. 125–135.
15. Majlis, M. Yet another language identifier. In Proceedings of the Student Research Workshop at the Conference of the European Chapter of the Association for Computational Linguistics, Avignon, France, 25–27 April 2012; pp. 46–54.
16. Gamallo, P.; Pichel, J.R.; Alegria, I. From language identification to language distance. *Phys. Stat. Mech. Appl.* **2017**, *484*, 152–162. [CrossRef]
17. Souter, C.; Churcher, G.; Hayes, J.; Hughes, J.; Johnson, S. Natural language identification using corpus-based models. *Hermes J. Linguist.* **1994**, *13*, 183–203.
18. Damashek, M. Gauging similarity with n-grams: Language-independent categorization of text. *Science* **1995**, *267*, 843–848. [CrossRef] [PubMed]

19. Goldhahn, D.; Quasthoff, U. Vocabulary-Based Language Similarity using Web Corpora. In Proceedings of the Ninth International Conference on Language Resource and Evaluation, Reykjavik, Iceland, 26–31 May 2014.
20. Zampieri, M.; Tan, L.; Ljubešić, N.; Tiedemann, J. A Report on the DSL Shared Task 2014. In Proceedings of the Workshop on Applying NlP TOOLS to Similar Languages, Dublin, Ireland, 23 August 2014; pp. 58–67.
21. Zampieri, M.; Tan, L.; Ljubešić, N.; Tiedemann, J.; Nakov, P. Overview of the DSL Shared Task 2015. In Proceedings of the LT4VarDial Workshop, Hissar, Bulgaria, 10 September 2015.
22. Zampieri, M.; Malmasi, S.; Ljubešić, N.; Nakov, P.; Ali, A.; Tiedemann, J. Discriminating Between Similar Languages and Arabic Dialect Identification: A Report on the Third DSL Shared Task. In Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects, Osaka, Japan, 12 December 2016; pp. 1–14.
23. Zampieri, M.; Malmasi, S.; Ljubešić, N.; Nakov, P.; Ali, A.; Tiedemann, J.; Scherrer, Y.; Aepli, N. Findings of the VarDial Evaluation Campaign. In Proceedings of the VarDial Workshop, Valencia, Spain, 3 April 2017.
24. Criscuolo, M.; Aluisio, S. Discriminating between similar languages with word level convolutional neural networks. In Proceedings of the VarDial Workshop, Valencia, Spain, 3 April 2017; pp. 124–130.
25. Belinkov, Y.; Glass, G. A Character-level Convolutional Neural Network for Distinguishing Similar Languages and Dialects. *arXiv* **2016**, arXiv:1609.07568.
26. Bjerva, B. Byte-based Language Identification with deep Convolutional Networks. *arXiv* **2016**, arXiv:1609.09004.
27. Goutte, C.; Léger, S.; Carpuat, M. The NRC System for Discriminating Similar Languages. In Proceedings of the First Workshop on Applying NLP Tools to Similar languages, Varieties and Dialects, Dublin, Ireland, 23 August 2014; pp. 139–145.
28. Porta, J.; Sancho, J.L. Using Maximum Entropy Models to Discriminate between Similar Languages and Varieties. In Proceedings of the First Workshop on Applying NLP Tools to Similar languages, Varieties and Dialects, Dublin, Ireland, 23 August 2014; pp. 120–128.
29. Lui, M.; Letcher, N.; Adams, O.; Long, D.; Cook, P.; Baldwin, T. Exploring Methods and Resources for Discriminating similar languages. In Proceedings of the First Workshop on Applying NLP Tools to Similar Languages, Varieties and Dialects, Dublin, Ireland, 23 August 2014; pp. 129–138.
30. Fabra-Boluda, R.; Rangel, F.; Rosso, P. NLEL UPV authoritas participation at Discrimination between Similar Language (DSL) 2015 shared task. In Proceedings of the LT4VarDial Workshop, Hissar, Bulgaria, 10 September 2015.
31. Ács, J.; Grad-Gyenge, L.; de Rezende Oliveira, T.B.R. A two-level classifier for discriminating similar languages. In Proceedings of the Joint Workshop on Language Technology for Closely Related Languages, Varieties and Dialects, Hissar, Bulgaria, 10 September 2015; pp. 73–77.
32. Goutte, C.; Léger, S.; Malmasi, S.; Zampieri, M. Discriminating Similar Languages: Evaluations and Explorations. *arXiv* **2016**, arXiv:1610.00031.
33. Hofmann, M.; Ralf, K. *Rapid Miner Data Mining Use Cases and Business Analytics Applications*; CRC Press: Boca Raton, FL, USA, 2014.
34. Polyglot. Available online: https://github.com/saffsd/polyglot (accessed on 7 March 2018).
35. Langid.py. Available online: https://github.com/saffsd/langid.py (accessed on 7 March 2018).
36. Lui, M.; BaldWin, T. Langid.py: An off the shelf Language Identification Tool. In Proceedings of the 50th Annual of the Association for Computational Linguistics, Jeju Island, Korea, 8–14 July 2012; pp. 25–30.
37. Goldhahn, D.; Eckart, T.; Quasthoff, U. Building Large Monolingual Dictionaries at the Leipzig Corpora Collection: From 100 to 200 languages. In Proceedings of the 8th International Language Resources and Evaluation (LREC'12), Istanbul, Turkey, 23–25 May 2012.
38. About the Leipzig Corpora Collection. Available online: http://asvdoku.informatik.uni-leipzig.de/corpora/ (accessed on 7 March 2018).
39. Codes for the Representation of Names of Languages. Available online: http://www.loc.gov/standards/iso639-2/php/code_list.php (accessed on 7 March 2018).
40. Forman, G. An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.* **2003**, *3*, 1289–1305.
41. Uysal, A.K.; Gunal, S. A Novel Probabilistic Feature Selection Method for Text Classification. *Knowl. Based Syst.* **2012**, *36*, 226–235. [CrossRef]
42. Rehman, R.; Javed, K.; Babri, A.B. Feature Selection Based on a Normalized Difference Measure for Text Classification. *Inf. Process. Manag.* **2017**, *53*, 473–489. [CrossRef]

43. Brank, J.; Grobelnik, M.; Milic-Frayling, N.; Mladenic, D. Interaction of feature selection methods and linear classification models. In Proceedings of the Workshop on Text Learning held at International Conference on Machine Learning (LCML), Sydney, Australia, 8–12 July 2002.
44. Sebastitiani, F. Machine learning, in automated text categorization. *ACM Comput. Surv.* **2002**, *34*, 1–47. [CrossRef]
45. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *JMLR* **2011**, *12*, 2825–2830.
46. Joachims, T. *Learning to Classify Using Support Vector Machines*; Dissertation; Kluwer: Alphen aan den Rijn, The Netherlands, 2002.