HeLI-OTS, Off-the-shelf Language Identifier for Text

Jauhiainen, Tommi

European Language Resources Association (ELRA)
2022

Jauhiainen , T , Jauhiainen , H & Lindén , K 2022 , HeLI-OTS, Off-the-shelf Language Identifier for Text . in N Calzolari , F Béchet & P Blache, et al. (eds) , Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022) . European Language Resources Association (ELRA) , Paris , pp. 3912-3922 .

http://hdl.handle.net/10138/350001

Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022), pages 3912–3922
Marseille, 20-25 June 2022

HeLI-OTS, Off-the-shelf Language Identifier for Text

Tommi Jauhiainen, Heidi Jauhiainen, Krister Lindén
University of Helsinki
{firstname.lastname}@helsinki.fi

Abstract

This paper introduces HeLI-OTS, an off-the-shelf text language identification tool using the HeLI language identification method. The HeLI-OTS language identifier is equipped with language models for 200 languages and licensed for academic as well as commercial use. We present the HeLI method and its use in our previous research. Then we compare the performance of the HeLI-OTS language identifier with that of fastText on two different data sets, showing that fastText favors the recall of common languages, whereas HeLI-OTS reaches both high recall and high precision for all languages. While introducing existing off-the-shelf language identification tools, we also give a picture of digital humanities-related research that uses such tools. The validity of the results of such research depends on the results given by the language identifier used, and especially for research focusing on the less common languages, the tendency to favor widely used languages might be very detrimental, which Heli-OTS is now able to remedy.

Keywords: language identification, text classification

1. Introduction

The significance of language identification as part of text preprocessing has grown at the same time as natural language processing (NLP) systems have become more common in widely used software applications. For example, to perform machine translation on a piece of text, the machine translation system must know the source text's language. Without automated language identification, the users have to indicate the language of the text manually. Google Translate[1] is an example of a system where language identification has been incorporated (Jauhiainen, 2019).

Language identification is a many-faceted task, and the specific aspect that the system presented in this paper sets out to tackle is the most general case of reliably identifying a large group of languages, most of which are not closely related. The HeLI-OTS, an off-the-shelf language identifier based on the HeLI method (Jauhiainen et al., 2016), includes a repertoire of 200 languages.

In Section 2, we introduce language identification as well as some of the existing off-the-shelf language identification tools. Section 3 presents the HeLI method together with some of the results of the previous language identification performance evaluations where the technique has been featured. In Section 4, we describe the HeLI-OTS tool itself, and in Section 5, we evaluate the performance of the current implementation and compare it with that of the fastText language identifier (Joulin et al., 2016). Section 6 brings to light some known limitations of the tool, Section 7 concludes our findings and Section 8 lists our current improvement targets for future versions of HeLI-OTS.

[1]`https://translate.google.com/`

2. Previous Work

2.1. Language Identification

The problem of identifying the language of digital text has been researched since the 1960s (Jauhiainen et al., 2019d). It can be considered a sub-task of text classification and many, if not most, automatic classification methods have been evaluated concerning their capability of distinguishing between languages.

We have been developing language identification methods for different scientifically motivated purposes for more than a decade, and we have published the source code of some of these systems openly. However, before the tool's publication presented in this paper, we had not issued a ready-to-use language identification system, including pre-compiled language models. We built the previously published systems for specific shared tasks and the needs of our Finno-Ugric Languages and the Internet project (Suki) (Jauhiainen et al., 2015a; Jauhiainen et al., 2021b). Even though the language identifier used in the project as part of the web-harvesting pipeline included language models for over 350 languages, we had selected the language repertoire that gives good results when looking for the low-resourced Uralic languages that were the target of the project. It was clear that the repertoire created problems for many majority languages. For HeLI-OTS, we have curated an assortment of languages that is at the same time quite sizeable but also provides accurate results for the languages concerned.

2.2. Existing Off-the-shelf Language Identifiers

TextCat was one of the first downloadable off-the-shelf language identifiers (van Noord, 1997).[2] It uses the language identification method presented by Cavnar and

[2]`https://www.let.rug.nl/vannoord/TextCat/`

Trenkle (1994) and is able to distinguish between 76 languages. It is no longer widely used, but for example, Somboonviwat et al. (2006) used it in detecting the language of web pages for creating a national Web archive for Thailand. One of the still widely used language identifiers is the *langid.py* by Lui and Baldwin (2012) with 97 languages (Lui, 2017).[3] Abu Kwaik et al. (2018) used it in constructing and evaluating a text corpus for Levantine Arabic dialects and Bérard et al. (2019) as well as Li et al. (2020) in data cleaning for machine translation. It was also used by Aulamo and Tiedemann (2019) in parallel corpora creation, by Hovy et al. (2020) in machine translation system evaluation, and by Frey (2020) in language choice research.

Some language identifiers are available as methods that the users can utilize directly in Java programs. The Apache OpenNLP suite[4] includes a "langdetect" package which can be incorporated into Java programs. It comes equipped with 103 languages trained using the corpora found in the Leipzig corpora collection.[5] It was used, for example, by Adams et al. (2019) as part of a privacy protection toolkit. Apache Tika, a content analysis toolkit, contains a language detector for 18 languages.[6] Ali et al. (2019) used it in pre-processing text for cross-lingual ontology enrichment.

Another Java implementation is the Language Detection Library[7] which has been used in NLP, for example, by Stab et al. (2018) for filtering English text from the CommonCrawl[8] Web corpus. It includes profiles for 53 languages. The "optimaize" language-detector comes with 71 built-in languages.[9] It was used by Saggion et al. (2017) and Ferrés et al. (2018) as part of a pipeline for extracting textual content from pdfs, by Glass and Gliozzo (2018) in a pipeline for building a knowledge base from text, by Tebbifakhr et al. (2018) in cleaning data used to create a model for automatic post-editing, by Abromeit and Chiarcos (2019) and Abromeit et al. (2020) in automatically annotating language resources with language metadata, by Mrabet et al. (2020) in creating a system for the TREC 2020 Health Misinformation track, and by Noei et al. (2019) to filter user reviews in issue report prioritization.

Today, one of the most used off-the-shelf identifiers is based on the *fastText* method and is equipped with 178 languages (Joulin et al., 2017; Joulin et al., 2016).[10] The fastText language identifier has recently been used in NLP, for example, by Hiippala et al. (2019) for studying the diversity of virtual linguistics landscape, Godinez et al. (2020) in medical text language identification, Kreutz and Daelemans (2020) for streaming language-specific Twitter data, Wenzek et al. (2020) for extracting monolingual datasets from web crawl data, and Alshaabi et al. (2021) for identifying and exploring the relative daily use of languages on Twitter.

All of the mentioned language identifiers can be used to discriminate between languages in their language repertoire. None of them are perfect: they perform very well for some tasks and for others very badly. How well they perform depends greatly on the task at hand. As language identification is usually an upstream task in NLP or other pipelines, its performance significantly affects the reliability of the whole pipeline. With the more precise off-the-shelf language identifier presented in this article, we aim to improve the usability of language identification in an even wider variety of workflows.

3. HeLI Language Identification Method

The system is based on language identification technology developed by the authors during several research projects since 2010. The HeLI method has proven efficient and has remained competitive in several language identification shared tasks. The HeLI method has been quite successful when used to discriminate between closely related languages (Jauhiainen et al., 2015b; Jauhiainen et al., 2016; Jauhiainen et al., 2017a; Jauhiainen et al., 2018a; Jauhiainen et al., 2018b; Jauhiainen et al., 2018c). In this Section, we describe the method as it is implemented in HeLI-OTS, which uses the same preprocessing scheme and parameters which performed best in our 2017 evaluation (Jauhiainen et al., 2017b). These parameters include, for example, the penalty score and the maximum length of character *n*-grams used.

3.1. Language Models and Features

The language models used in the HeLI-OTS consist of single words and character *n*-grams. These character *n*-grams of length from one to six characters have been generated from word segments, including the beginning and end of the words manifested by a space character. All characters were lowercased when compiling the models, and only the characters present in the alphabets, logograms, or syllabaries of languages were retained. In addition to those characters defined to belong to the Unicode letter characters by Java,[11] including, for example, Chinese and Hieroglyphic signs, we considered some additional characters as integral parts of words.[12] All other characters were transformed into space characters and used as word-delimiters.

[3]`https://github.com/saffsd/langid.py`

[4]https://opennlp.apache.org

[5]`https://corpora.uni-leipzig.de/en`

[6]`http://tika.apache.org`

[7]`https://github.com/shuyo/language-detection`

[8]`http://commoncrawl.org/`

[9]`https://github.com/optimaize/language-detector`

[10]`https://fasttext.cc/docs/en/language-identification.html`

[11]We used the regex tokens \p{L} and \p{M}.

[12]The list of these additional characters can be seen in the *identifyLanguage* function of the HeLI.java file at `http://urn.fi/urn:nbn:fi:lb-2022011702`.

3.2. Method

When identifying the language of a text, the program first lowercases all characters and converts non-letter characters into white spaces in a similar manner as was done when compiling the language models. The text is then divided into words. Each word is given a score for each language. The scores for words are calculated using the highest order language models possible. Words are considered to be of the highest order, followed by character *n*-grams from the longest to the shortest. If the word being scored is found in at least one language, the word is scored using the word models. The score is the negative logarithm of the relative frequency of the word in each language. If the word is missing from some language, a fixed penalty score of `7.0` is used. If the word is not found in any of the word models, the method backs off to using the character 6-grams. The word is divided into overlapping character *n*-grams, and the word gets a score equal to the average of the scores of these *n*-grams. The same fixed penalty value is used in case an *n*-gram is missing from some language. If none of the 6-grams generated from the word are found in any of the models, the method backs off to using character 5-grams and continues to as short *n*-grams as needed. Finally, for the whole sentence, each language gets a score equal to the average of the scores of the words.

The HeLI method works very well for languages using clear boundaries between words as it can give equal opportunity to each word to influence the outcome. For example, the word "the" is very characteristic of English and is given the same opportunity as the word "international". If only character *n*-grams were used, "international" would have roughly four times the weight of "the". HeLI can also discriminate between languages that do not use white spaces as word boundary markers but does not perform as well as with languages using them. For example, in Chinese, complete sentences are considered one "word". Furthermore, in the case of code-switching, the method gives disproportionate value to short sequences of Latin characters amid, for example, a long continuous line of Chinese characters. The program has a built-in fix for this latter problem so that when a text has more than 50% of its characters from the CJK character set, only Japanese, Mandarin, or Korean can be given as a result.

3.3. Performance

In 2016, we achieved the shared first position in the Discriminating Between Similar languages shared task using the HeLI method (Jauhiainen et al., 2016; Malmasi et al., 2016). In that shared task, the aim was to distinguish between 12 languages or language variants, some of which were very similar to each other, such as the Argentine, Castilian, and Mexican variants of written Spanish. In 2018, we equipped the HeLI method with adaptive language models and won both the German Dialect Identification (GDI) and the Indo-Aryan Language Identification (ILI) shared tasks (Jauhiainen et al., 2018b; Jauhiainen et al., 2018c; Zampieri et al., 2018).

In 2017, we evaluated several language identification methods (Cavnar and Trenkle, 1994; Vogel and Tresner-Kirsch, 2012; King and Dehdari, 2008; Brown, 2013; Vatanen et al., 2010) in a test setting combining several challenging contexts: the training and testing sets being from different domains (mostly Wikipedia for training and Universal Declaration of Human Rights for testing), the amount of training data for some languages being meager (starting from 2,710 words), the texts to be identified being very short (starting from 5 characters), and the number of possible languages being reasonably large (285 languages in the repertoire of the language identifier) (Jauhiainen et al., 2017b). In our evaluation, when using fewer than 20 characters, the HeLI method was outperformed by the absolute discounting method used by Vatanen et al. (2010) but attained the best F1-score in all the tests with more than 20 characters and having, for example, eight times fewer recall errors and five times fewer precision errors with test texts of 100 characters, than the second-best evaluated method.

In 2020, we organized the Uralic Language Identification (ULI) shared task with three separate tracks (Gaman et al., 2020; Jauhiainen et al., 2020b). The first two tracks focused on the problematic issues in finding rare languages among many more common languages, but the third track was the first general-purpose language identification shared task to date to feature a large number of languages. The ULI shared task was continued in the VarDial evaluation campaign 2021 as an open leaderboard (Chakravarthi et al., 2021) and the best language identification method to date in the third track is still the HeLI method which we used as a baseline. The HeLI method was used with identical parameters to the 2017 evaluation, and no parameters were optimized for the ULI task. Anyone wanting to evaluate additional language identification methods is welcome to submit new results, which will be added to the leaderboard for as long as the leaderboard is active. The information on how to get the training and testing data can be found on the leaderboard page.[13]

The continued competitive performance of the HeLI method and the possibilities given to us by our ongoing "Language Identification of Speech and Text" project[14] drove us to publish a general-purpose off-the-shelf language identification tool.

[13]`http://urn.fi/urn:nbn:fi:lb-2020102201`

[14]`https://www.vaikuttavuussaatio.fi/en/funded-projects/`

4. HeLI-OTS

The HeLI-OTS 1.2 is described in the Metashare service[15] of the Language Bank of Finland[16], and it is downloadable from Zenodo (Jauhiainen and Jauhiainen, 2022).[17] The software is available under both CC-BY and Apache 2 licenses. The complete download package includes the Java source code of the identifier and language models for 200 languages. If the user is interested in only using the language identifier, it is sufficient to download just the pre-compiled `HeLI.jar` file (44.1MB), which can be used as a standalone language identifier in systems equipped with the Java 16 runtime environment.[18] It can identify *c.* two thousand sentences per second using one computing core (3.2 GHz) and around 3 gigabytes of memory on a 2021 laptop computer.

We have included two Python files to provide a simple example of how `HeLI.jar` can be used in a Python environment.

In the `LanguageModels` directory, there are seven files per language containing the features mentioned earlier and their frequencies in the material used for training. Each file contains a maximum of 10,000 features.

The frequencies for words and character *n*-grams in different languages have been calculated using different corpora. The languages that were part of the repertoire of the ULI shared task (Jauhiainen et al., 2020b) all used the same corpora: Wanca 2016 corpora[19] (Jauhiainen and Jauhiainen, 2020) for rare Uralic languages (Jauhiainen et al., 2019a) and corpora from the Leipzig corpora collection[20] (Goldhahn et al., 2012) for others (Biemann et al., 2007). For the rest of the languages, the training sources are mostly the same as in the 2017 evaluation[21] (Jauhiainen et al., 2017b). In 2021, the HeLI-OTS 1.1 was tested by Lingsoft[22] and the Language Bank of Finland, and they encountered practical problems with dialectal Finnish, which tended to be identified as one of the close languages, such as Livvi-Karelian (olo). In order to remedy this problem, which we had already encountered earlier when crawling the Finnish internet during the Suki project, ten additional models for dialectal Finnish were added using the sources gathered during the project.

[15]`http://urn.fi/urn:nbn:fi:lb-2022011701`

[16]`https://www.kielipankki.fi/language-bank/`

[17]`http://urn.fi/urn:nbn:fi:lb-2022011702`

[18]To date the identifier has been tested on Linux and Mac OS environments.

[19]`http://urn.fi/urn:nbn:fi:lb-2020022901`

[20]`https://corpora.uni-leipzig.de/en`

[21]`http://suki.ling.helsinki.fi/LILanguages.html`

[22]`https://www.lingsoft.fi/en`

```
[ido],3.5029755
[lmo],3.5215342
[mwl],3.5261147
[epo],3.5410736
[spa],3.5475852

[lmo],2.7412848
[ina],3.0175257
[cat],3.0623221
[spa],3.089319
[epo],3.111251
```

Table 1: Example output using "-t 5" option.

4.1. Usage

The identifier can read the text to be identified from either the standard input or from an existing file. If the text is read from standard input, each line entered will be identified separately, and the result will be printed to standard output as an ISO 639-3 three-letter language identifier. The program will continue to receive text to be identified until the EOF character is sent to the standard input.

If an input file is specified using the `-r` option, the program will identify each line found from the file and print the results to standard output or a file defined by the `-w` option.

If the `-t` option is used followed by number *n*, the identifier will print the *n* best languages followed by their language scores (lower is better) for each mystery text. The list of top scores for each text is printed one per line, and an empty line indicates moving to the following mystery text. Table 1 gives an example of such an output.

The `-l` option can restrict the language repertoire loaded when the program commences. The program still reads in the list of languages from the *language-list* file, but the models are loaded for only those languages in which identifiers begin with one of the identifiers listed after the option. For example with `-l hsb,slv` the models for *hbsbos*, *hbshrv*, *hbssrp*, and *slv* are loaded.

By default, the HeLI-OTS supposes that the first and the last character sequences in a text to be identified are complete words. This might not be the case in some situations, and if the `-p` option is used, the supposition is not made for the last sequence.

See Table 2 for command examples.

4.2. Confidence Scores

Using the `-c` option causes the program to print a confidence score after the ISO 639-3 identifier. The confidence score is the absolute difference between the language scores of the predicted language and the second-best language. The higher the score is, the more confident the identification. This relatively simple way of calculating a confidence score is the same we use in our language model adaptation method (Jauhiainen et al.,

Command	Function
`java -jar HeLI.jar`	reads the stdin line by line and writes ISO 639-3 identifiers to stdout until EOF
`java -jar HeLI.jar -r <infile> -w <outfile>`	reads the `<infile>` and writes ISO 639-3 identifiers to `<outfile>`
`-t <number>`	prints the top `<number>` languages for each line
`-l <xxx>,<yyy>,...`	restricts the language repertoire to listed languages
`-p`	considers the last token of each mystery text to be only partial word
`-h`	prints usage instructions
`-c`	prints confidence scores after ISO 639-3 identifiers

Table 2: Example commands for HeLI-OTS.

2019c) for deciding the order of adding new material to the language models. It has proven to be very efficient (Jauhiainen et al., 2019b; Jauhiainen et al., 2021a).

This confidence score could be used to detect unseen languages using thresholding, but setting a global confidence threshold for the identifier is practically impossible as the score differences depend significantly on the languages the user is targeting. If the user is interested in a language with no structural relatives among the repertoire of the language identifier, the threshold can be much higher than for those languages with close relatives.

4.3. Creating Additional Language Models

The HeLI-OTS 1.2 language identifier is accompanied by a `createmodels.java` program which can create additional or customized language models. The current version of the program reads in text files named `<ISO_639-3_identifier>.train` from a sub-directory `Training` and writes seven files beginning with the language identifier to `Models` subdirectory. In order to use these additional models, the `<ISO_639-3_identifier>` has to be added to the `languagelist` file, and the corresponding model files moved to the `LanguageModels` directory. The newly created model files may contain considerably more features than the 10,000 shipped with the current implementation, so it might be necessary to trim them if there are issues with memory or storage space. The three first characters of the `<ISO_639-3_identifier>` are used as the language to which the HeLI-OTS will map the results. If such a language already exists in the repertoire, the best score is used, which is the case, for example, with the additional dialectal Finnish models. The additional Finnish models have `<ISO_639-3_identifier>`s such as `fini`, `finj`, and `fink` and they are all mapped to `fin` when producing the results.

With this program, creating language models for all the 214 languages or variants took 4,5 hours using one core from a laptop computer. Unpacked, these language models take 115 megabytes of storage space, so about half a megabyte per language. When adding new languages, it is not necessary to re-train the existing language models, as HeLI is a generative classifier and the models are independent of each other.

5. Evaluation

Comparing language identification systems is a complicated issue as the evaluation setup can greatly affect the evaluation outcome. A recent evaluation, in which *fastText* was superior to the *langid.py* language identifier, was carried out by Toftrup et al. (2021). Another evaluation with similar results was carried out earlier by Hiippala et al. (2019). In light of these evaluations, we decided to compare HeLI-OTS especially with the fastText implementation, which Toftrup et al. (2021) referred to as a *de facto* standard in language identification.

5.1. Initial Experiments

Hiippala et al. (2019) evaluated the performance of the *fastText*, *langid.py*, and *CLD2* in identifying the language of Instagram posts. Their test set included 1,755 sentences.[23] We rerun *langid.py* on the sentences and received a micro F_1 score of 0.786 and a macro F-score of 0.628. For HeLI-OTS, our initial test gave a micro F_1 score of 0.863 and a macro F_1 score of 0.685. We thought especially the macro F-score to be rather low when compared with, for example, the macro F_1 score of 0.925 reached using the HeLI method at the ULI-178 shared task, so we manually inspected some of the results to see what kind of errors our identifier made.

The test set includes some sentences, the language of which could not be identified with certainty, such as just the placename “Helsinki” written in Cyrillic, which is labeled as Russian even though it could equally well be one of the other languages written in Cyrillic. Additionally, we noticed that some of our training data (such as those for Ewe “ewe”, Igbo “ibo”, and Tagalog “tgl”) included lots of text written in English. Text in English and other languages using Latin characters were also found in the training corpus for Chinese (cmn), Japanese (jpn), and Korean (kor), as

[23]The amount is different from the 1,688 reported in their paper, but we checked with the authors, and 1,755 is the correct number for sentences.

	afr	cmn	eng	ewe	fin	fry	ibo	jpn	ltz	lus	myv	pam	roh	tgl	others
cmn		7						2							
eng	3	3	694	10	6	4	31	6	8	3	19	3	3	13	32
fin					417				1		3	2	2		16
jpn		1						26							

Table 3: Languages confused with English when using Hiippala et al. test set before corpora cleaning.

Program	**#languages**	**Testing data**	**Macro F_1**	**Micro F_1**	**Micro Recall**	**Micro Precision**
HeLI-OTS 1.2	200	ULI 110	0.699	0.993	0.993	0.993
fastText	176	ULI 110	0.536	0.822	0.822	0.822

Table 4: Results of the evaluations using ULI 110 test set.

well as in corpora of languages using Cyrillic characters. Table 3 is a confusion table showing the languages to which English text was identified more than twice.

We ended up cleaning the training set by manually inspecting several training texts. We used word frequency lists generated from the training texts to find frequent foreign words for some languages. We removed some of the foreign language incursions we found, most of which were in English. Furthermore, Latin characters were automatically filtered from corpora such as Japanese and Mandarin. After this, the micro F_1 score rose to 0.876, and the macro F_1 score to 0.693. The improvement was clear but not substantial, and, since we had inspected the errors made by HeLI-OTS, the results were no more comparable with those attained without learning from the test set.

5.2. ULI 110

In order to have some indication of the effectiveness of HeLI-OTS compared with the existing off-the-shelf language identifiers, we built ULI 110, a new test set with 110 languages. This new test set contains 1,074,939 sentences in languages in which fastText and HeLI claim to be proficient. It is a subset of the ULI test set (178 languages) (Jauhiainen et al., 2020b) with only the shared languages from the repertoires of the fastText (176 languages) (Joulin et al., 2017; Joulin et al., 2016) and the HeLI-OTS (200 languages) language identifiers.

Including the time to read in the language models, it took less than 10 minutes for HeLI-OTS 1.2 to process the test sentences using one laptop computing core, so *c*. 1,900 sentences per second. The results of the experiments can be seen in Table 4. With HeLI-OTS, the macro F_1 score was 0.699, and the micro F_1 score was 0.993. This macro F_1 was quite similar to the one attained when using the Instagram captions test set, but the micro F_1 score was extremely high in comparison. With fastText, the macro F_1 score was 0.536 and the micro F_1 score 0.822. Roughly put, this means that in this kind of test setting, fastText makes an error with every fifth sentence and HeLI-OTS just once in a hundred sentences. The averaged micro recall, micro-precision, and micro F_1 scores are very close because 107 of the 110 languages had the same number of test sentences, in which case a recall error in one language means an equal precision error in another. We assumed the difference in performance between the two identifiers was magnified by the careful curation of the HeLI-OTS language repertoire, so we took a closer look at the results of those tested languages in which fastText had fared the worst in order to see whether the training and the testing languages were, in fact, the same.

The worst performing language was Palatine German (pfl), with a recall of 0.0001 and precision of 0.1667. The testing data was from pfl Wikipedia from Leipzig Corpora Collection (LCC). pfl is on the list of languages supported by fastText;[24] however, a mystery text was identified as pfl only six times, and only one of those texts was actually in pfl. The second worst performing language was Central Bikol (bcl), with an F-score of 0.0169. The testing data was again from the corresponding Wikipedia collection at LCC. Although bcl is on the fastText list of languages, most bcl texts were identified as Tagalog. The third worst performing language was Sardinian (srd), with a recall of 0.0158. The test material again originated from Wikipedia. Of the 10,000 Sardinian mystery texts, 2,931 were identified as Italian, 1,887 Spanish, 1,581 as Catalan, and 1,043 as French. These three examples show that fastText performs poorly for some of the rarer languages with close relatives in the test set. Partly due to some of these rare languages having a low recall, the F-score of some of the more common languages remained low. Table 5 shows how fastText has a slightly higher recall than HeLI-OTS for German, English, and Italian, but it comes with a high price in precision.

5.3. OpenSubtitles

In order to test the performance in a setting not defined by ourselves, we used the evaluation setup described by Toftrup et al. (2021) as an inspiration. We took a subset of the 10,000 short texts from the OpenSubtitles (Lison and Tiedemann, 2016) for each of the 20 languages evaluated by Toftrup et al. (2021). Like Toftrup et al. (2021), we removed all special characters, made sure that the texts started with a word, and cut the texts

[24] `https://fasttext.cc/docs/en/language-identification.html`

Program	language	F_1	Recall	Precision
HeLI-OTS 1.2	German	0.9894	0.9851	0.9937
fastText	German	0.5539	0.9989	0.3832
HeLI-OTS 1.2	English	0.9643	0.9974	0.9334
fastText	English	0.4941	0.9982	0.3283
HeLI-OTS 1.2	Italian	0.9451	0.9257	0.9654
fastText	Italian	0.4057	0.9588	0.2573

Table 5: Results for German, English, and Italian with fastText and HeLI-OTS using ULI 110.

Program	#languages	Testing data	Macro F_1	Micro F_1
HeLI-OTS 1.2	20	OpenSubtitles 20	0.824	0.824
HeLI-OTS 1.2	200	OpenSubtitles 20	0.408	0.699
fastText	176	OpenSubtitles 20	0.263	0.635
fastText	20	Toftrup OS 20	0.675	
langid.py	20	Toftrup OS 20	0.543	

Table 6: Results of the evaluations using Open Subtitles.

after ten characters so that the length of each text to be examined was exactly ten characters. The results of the experiments can be seen in Table 6. We used the `-p` option with HeLI, as the test length was relatively short, and it seemed probable that the last tokens would not be complete words. Without restricting the HeLI-OTS to the 20 relevant languages, the macro F_1 score attained was 0.408, which is quite good compared with the 0.263 attained by fastText for the same data. The micro F_1 for HeLI-OTS was also better (0.699) than the one for fastText (0.635).[25] The HeLI-OTS 1.2 processed the 200,000 text excerpts in 59 seconds.[26] When we restricted the HeLI-OTS to the 20 relevant languages, the macro F_1 score rose to 0.824, clearly better than those attained by either off-the-shelf fastText or langid.py in the evaluations of Toftrup et al. (2021). Restricting fastText to a subset of its language repertoire is impossible without producing results for all the languages and extracting the relevant subset or, alternatively, retraining the whole model (Toftrup et al., 2021). When restricted to the 20 relevant languages, it took only 8 seconds for HeLI-OTS to process the texts.

6. Limitations

Most of the languages included in the repertoire of the HeLI-OTS can be considered easy to distinguish from each other. Having rare languages in the repertoire of an identifier creates practical problems, especially if these languages are close relatives or easily mixed with more common languages. However, as we took the ULI shared task language repertoire as our starting point, the HeLI-OTS identifier's repertoire includes several rare Uralic languages, such as five separate Saami languages.

The HeLI-OTS does not include unseen language detection, so it always labels the text as one of the languages it knows unless the text is completely removed during preprocessing, in which case it gives "xxx" as the result instead of the ISO 639-3 identifier of one of the languages. The software includes an option to print confidence scores which can be, and have been, used to detect unseen languages. However, this is not a default option as the threshold depends on the intended usage of the program. As the system maps a text into exactly one language, it does not solve the problem of the identification of multilingual texts. It can, however, be used to detect the languages in multilingual documents by dividing the documents into smaller parts, for example, sentences. It can also be used as a component in the language set identification system we have published previously on GitHub (Jauhiainen et al., 2019a; Jauhiainen et al., 2020a).[27]

7. Conclusions

As can be witnessed from the research articles listed in Section 2.2, off-the-shelf language identifiers are widely used in a variety of digital humanities-related research. The validity of the results of such research depends on the results given by the language identifier used. From our evaluation results (Tables 4 and 5), we can see that fastText favors the recall of common languages over their precision and the recall of more rare languages. This behavior might not be a problem for specific research questions, especially those focusing on one of the more common languages. However, this behavior might be detrimental to research focusing on the less common languages. We have shown that, by using the HeLI-OTS 1.2, it is possible to reach both high recall and high precision for all the languages, even if the number of languages is reasonably large.

[25] For both programs, we mapped the various Norwegian languages to one.

[26] fastText managed to process the excerpts in less than 6 seconds.

[27] `https://github.com/tosaja/TunnistinPalveluMulti`

8. Future Work

We have identified a few improvement targets for future versions of the HeLI-OTS language identifier:

- The handling of languages not using clearly defined word boundary markers should be improved in future releases.
- As mentioned earlier, the HeLI method combined with automatic language model adaptation was very successful in some of the shared tasks. We plan to incorporate this function into HeLI-OTS in the future.

9. Acknowledgments

The production and publication of the HeLI-OTS tool, as well as the writing of this article, have been partly supported by the Finnish Research Impact Foundation[28] from its Tandem Industry Academy 2020 call in cooperation with Lingsoft Oy.[29] We thank Lingsoft and especially Ilkka Koskenniemi for testing the software and good improvement suggestions. We also thank Tuomo Hiippala for providing a Python user's point of view.

10. Bibliographical References

Abromeit, F. and Chiarcos, C. (2019). Automatic detection of language and annotation model information in conll corpora. In *2nd Conference on Language, Data and Knowledge (LDK 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Abromeit, F., Fäth, C., and Glaser, L. (2020). Annohub – annotation metadata for linked data applications. In *Proceedings of the 7th Workshop on Linked Data in Linguistics (LDL-2020)*, pages 36–44, Marseille, France, May. European Language Resources Association.

Abu Kwaik, K., Saad, M. K., Chatzikyriakidis, S., and Dobnik, S. (2018). Shami: A corpus of levantine arabic dialects. In *Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)*.

Adams, A., Aili, E., Aioanei, D., Jonsson, R., Mickelsson, L., Mikmekova, D., Roberts, F., Valencia, J. F., and Wechsler, R. (2019). Anonymate: A toolkit for anonymizing unstructured chat data. In *Proceedings of the Workshop on NLP and Pseudonymisation*, pages 1–7.

Ali, M., Fathalla, S., Ibrahim, S., Kholief, M., and Hassan, Y. F. (2019). Cloe: a cross-lingual ontology enrichment using multi-agent architecture. *Enterprise Information Systems*, 13(7-8):1002–1022.

Alshaabi, T., Dewhurst, D. R., Minot, J. R., Arnold, M. V., Adams, J. L., Danforth, C. M., and Dodds, P. S. (2021). The growing amplification of social media: measuring temporal and social contagion dynamics for over 150 languages on twitter for 2009–2020. *EPJ data science*, 10(1):1–28.

Aulamo, M. and Tiedemann, J. (2019). The opus resource repository: An open package for creating parallel corpora and machine translation services. *NoDaLiDa 2019*, page 389.

Bérard, A., Calapodescu, I., and Roux, C. (2019). Naver labs europe's systems for the wmt19 machine translation robustness task. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 526–532.

Biemann, C., Heyer, G., Quasthoff, U., and Richter, M. (2007). The Leipzig Corpora Collection - monolingual corpora of standard size. *Proceedings of Corpus Linguistic 2007*.

Brown, R. D. (2013). Selecting and Weighting N-grams to Identify 1100 Languages. In *Proceedings of the 16th International Conference on Text, Speech and Dialogue (TSD 2013)*, pages 475–483, Plzeň, Czech Republic.

Cavnar, W. B. and Trenkle, J. M. (1994). N-Gram-Based Text Categorization. In *Proceedings of SDAIR-94, Third Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, Las Vegas, USA.

Chakravarthi, B. R., Mihaela, G., Ionescu, R. T., Jauhiainen, H., Jauhiainen, T., Lindén, K., Ljubešić, N., Partanen, N., Priyadharshini, R., Purschke, C., Rajagopal, E., Scherrer, Y., and Zampieri, M. (2021). Findings of the VarDial evaluation campaign 2021. In *Proceedings of the Eighth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 1–11, Kiyv, Ukraine, April. Association for Computational Linguistics.

Ferrés, D., Saggion, H., Ronzano, F., and Bravo, À. (2018). Pdfdigest: an adaptable layout-aware pdf-to-xml textual content extractor for scientific articles. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Frey, J. C. (2020). *Using data mining to repurpose German language corpora. An evaluation of data-driven analysis methods for corpus linguistics.* Ph.D. thesis, University of Bologna Digital Library.

Gaman, M., Hovy, D., Ionescu, R. T., Jauhiainen, H., Jauhiainen, T., Lindén, K., Ljubešić, N., Partanen, N., Purschke, C., Scherrer, Y., and Zampieri, M. (2020). A report on the VarDial evaluation campaign 2020. In *Proceedings of the 7th Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 1–14, Barcelona, Spain (Online), December. International Committee on Computational Linguistics (ICCL).

Glass, M. and Gliozzo, A. (2018). A dataset for web-scale knowledge base population. In Aldo Gangemi, et al., editors, *The Semantic Web*, pages 256–271, Cham. Springer International Publishing.

[28]`https://www.vaikuttavuussaatio.fi/en/`
[29]`https://www.lingsoft.fi/en`

Godinez, E. V., Szlávik, Z., Santamaría, S. B., and Sips, R.-J. (2020). Language identification for short medical texts. In *HEALTHINF*, pages 399–406.

Hiippala, T., Hausmann, A., Tenkanen, H., and Toivonen, T. (2019). Exploring the linguistic landscape of geotagged social media content in urban environments. *Digital Scholarship in the Humanities*, 34(2):290–309.

Hovy, D., Bianchi, F., and Fornaciari, T. (2020). "you sound just like your father" commercial machine translation systems include stylistic biases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1686–1690, Online, July. Association for Computational Linguistics.

Jauhiainen, H., Jauhiainen, T., and Lindén, K. (2015a). The Finno-Ugric Languages and The Internet Project. *Septentrio Conference Series*, 0(2):87–98.

Jauhiainen, T., Jauhiainen, H., and Lindén, K. (2015b). Discriminating similar languages with token-based backoff. In *Proceedings of the Joint Workshop on Language Technology for Closely Related Languages, Varieties and Dialects*, pages 44–51, Hissar, Bulgaria, September. Association for Computational Linguistics.

Jauhiainen, T., Lindén, K., and Jauhiainen, H. (2016). HeLI, a word-based backoff method for language identification. In *Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial3)*, pages 153–162, Osaka, Japan, December. The COLING 2016 Organizing Committee.

Jauhiainen, T., Lindén, K., and Jauhiainen, H. (2017a). Evaluating HeLI with non-linear mappings. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, pages 102–108, Valencia, Spain, April. Association for Computational Linguistics.

Jauhiainen, T., Lindén, K., and Jauhiainen, H. (2017b). Evaluation of language identification methods using 285 languages. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 183–191, Gothenburg, Sweden, May. Association for Computational Linguistics.

Jauhiainen, T., Jauhiainen, H., and Lindén, K. (2018a). HeLI-based experiments in discriminating between Dutch and Flemish subtitles. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*, pages 137–144, Santa Fe, New Mexico, USA, August. Association for Computational Linguistics.

Jauhiainen, T., Jauhiainen, H., and Lindén, K. (2018b). HeLI-based experiments in Swiss German dialect identification. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*, pages 254–262, Santa Fe, New Mexico, USA, August. Association for Computational Linguistics.

Jauhiainen, T., Jauhiainen, H., and Lindén, K. (2018c). Iterative language model adaptation for Indo-Aryan language identification. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*, pages 66–75, Santa Fe, New Mexico, USA, August. Association for Computational Linguistics.

Jauhiainen, H., Jauhiainen, T., and Linden, K. (2019a). Wanca in Korp: Text corpora for underresourced Uralic languages. In Jarmo Harri Jantunen, et al., editors, *Proceedings of the Research data and humanities (RDHUM) 2019 conference*, number 17 in Studia Humaniora Ouluensia, pages 21–40, Finland. University of Oulu.

Jauhiainen, T., Jauhiainen, H., and Lindén, K. (2019b). Discriminating between Mandarin Chinese and Swiss-German varieties using adaptive language models. In *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 178–187, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Jauhiainen, T., Lindén, K., and Jauhiainen, H. (2019c). Language model adaptation for language and dialect identification of text. *Natural Language Engineering*, 25(5):561–583.

Jauhiainen, T., Lui, M., Zampieri, M., Baldwin, T., and Lindén, K. (2019d). Automatic Language Identification in Texts: A Survey. *Journal of Artificial Intelligence Research*, 65:675–782.

Jauhiainen, H., Jauhiainen, T., and Lindén, K. (2020a). Building web corpora for minority languages. In *Proceedings of the 12th Web as Corpus Workshop*, pages 23–32, Marseille, France, May. European Language Resources Association.

Jauhiainen, T., Jauhiainen, H., Partanen, N., and Lindén, K. (2020b). Uralic language identification (ULI) 2020 shared task dataset and the wanca 2017 corpora. In *Proceedings of the 7th Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 173–185, Barcelona, Spain (Online), December. International Committee on Computational Linguistics (ICCL).

Jauhiainen, T., Jauhiainen, H., and Lindén, K. (2021a). Naive Bayes-based experiments in Romanian dialect identification. In *Proceedings of the Eighth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 76–83, Kiyv, Ukraine, April. Association for Computational Linguistics.

Jauhiainen, T., Jauhiainen, H., and Linden, K. (2021b). Suomalais-ugrilaiset kielet ja internet-projekti 2013-2019. In *Multilingual Facilitation*. University of Helsinki Library.

Jauhiainen, T. (2019). *Language identification in texts*. Ph.D. thesis, University of Helsinki, Finland.

Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., and Mikolov, T. (2016). Fasttext.zip:

Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2017). Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain, April. Association for Computational Linguistics.

King, J. and Dehdari, J. (2008). An N-gram Based Language Identification System. The Ohio State University.

Kreutz, T. and Daelemans, W. (2020). Streaming language-specific Twitter data with optimal keywords. In *Proceedings of the 12th Web as Corpus Workshop*, pages 57–64, Marseille, France, May. European Language Resources Association.

Li, M., Cheng, H., Wang, Y., Zhang, S., Wu, L., and Guo, Y. (2020). BIT's system for the AutoSimTrans 2020. In *Proceedings of the First Workshop on Automatic Simultaneous Translation*, pages 37–44, Seattle, Washington, July. Association for Computational Linguistics.

Lison, P. and Tiedemann, J. (2016). OpenSubtitles2016: Extracting large parallel corpora from movie and TV subtitles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 923–929, Portorož, Slovenia, May. European Language Resources Association (ELRA).

Lui, M. and Baldwin, T. (2012). langid.py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 System Demonstrations*, pages 25–30, Jeju Island, Korea, July. Association for Computational Linguistics.

Malmasi, S., Zampieri, M., Ljubešić, N., Nakov, P., Ali, A., and Tiedemann, J. (2016). 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 (VarDial3)*, pages 1–14, Osaka, Japan, December. The COLING 2016 Organizing Committee.

Mrabet, Y., Sarrouti, M., Abacha, A. B., Gayen, S., Goodwin, T., Rae, A., Rogers, W., and Demner-Fushman, D. (2020). Nlm at trec 2020 health misinformation and deep learning tracks. In *Proceedings of the Twenty-Ninth Text Retrieval Conference (TREC 2020)*.

Noei, E., Zhang, F., Wang, S., and Zou, Y. (2019). Towards prioritizing user-related issue reports of mobile applications. *Empirical Software Engineering*, 24(4):1964–1996.

Saggion, H., Ronzano, F., Accuosto, P., and Ferrés, D. (2017). Multiscien: a bi-lingual natural language processing system for mining and enrichment of scientific collections. In *Mayr P, Chandrasekaran MK, Jaidka K, editors. Proceedings of the 2nd Joint Workshop on Bibliometric-enhanced Information Retrieval and Natural Language Processing for Digital Libraries (BIRNDL 2017); 2017 Aug 11; Tokyo, Japan.[place unknown]: CEUR Workshop Proceedings; 2017. p. 26-40.* CEUR Workshop Proceedings.

Somboonviwat, K., Tamura, T., and Kitsuregawa, M. (2006). Finding thai web pages in foreign web spaces. In *22nd International Conference on Data Engineering Workshops (ICDEW'06)*, pages x135–x135.

Stab, C., Daxenberger, J., Stahlhut, C., Miller, T., Schiller, B., Tauchmann, C., Eger, S., and Gurevych, I. (2018). ArgumenText: Searching for arguments in heterogeneous sources. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 21–25, New Orleans, Louisiana, June. Association for Computational Linguistics.

Tebbifakhr, A., Agrawal, R., Negri, M., and Turchi, M. (2018). Multi-source transformer with combined losses for automatic post editing. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 846–852, Belgium, Brussels, October. Association for Computational Linguistics.

Toftrup, M., Asger Sørensen, S., Ciosici, M. R., and Assent, I. (2021). A reproduction of apple's bi-directional LSTM models for language identification in short strings. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 36–42, Online, April. Association for Computational Linguistics.

Vatanen, T., Väyrynen, J. J., and Virpioja, S. (2010). Language identification of short text segments with n-gram models. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May. European Language Resources Association (ELRA).

Vogel, J. and Tresner-Kirsch, D. (2012). Robust Language Identification in Short, Noisy Texts: Improvements to LIGA. In Martin Atzmueller et al., editors, *Proceedings of the 3rd International Workshop on Mining Ubiquitous and Social Environments (MUSE)*, pages 43–50, Bristol, UK.

Wenzek, G., Lachaux, M.-A., Conneau, A., Chaudhary, V., Guzmán, F., Joulin, A., and Grave, E. (2020). CCNet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4003–4012, Marseille, France, May. European Language Resources Association.

Zampieri, M., Malmasi, S., Nakov, P., Ali, A., Shon, S., Glass, J., Scherrer, Y., Samardžić, T., Ljubešić, N., Tiedemann, J., van der Lee, C., Grondelaers, S., Oostdijk, N., Speelman, D., van den Bosch, A., Kumar, R., Lahiri, B., and Jain, M. (2018). Language identification and morphosyntactic tagging: The second VarDial evaluation campaign. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*, pages 1–17, Santa Fe, New Mexico, USA, August. Association for Computational Linguistics.

11. Language Resource References

Goldhahn, Dirk and Eckart, Thomas and Quasthoff, Uwe. (2012). *Leipzig Corpora Collection*. University of Leipzig: `https://wortschatz.uni-leipzig.de/en/download`.

Jauhiainen, Heidi and Jauhiainen, Tommi. (2020). *Wanca 2016, source*. University of Helsinki, distributed via the Language Bank of Finland: `http://urn.fi/urn:nbn:fi:lb-2020022901`.

Jauhiainen, Tommi and Jauhiainen, Heidi. (2022). *HeLI-OTS 1.2, off-the-shelf language identifier*. University of Helsinki, distributed via Zenodo: DOI 10.5281/zenodo.5890998.

Joulin, Armand and Grave, Edouard and Bojanowski, Piotr and Douze, Matthijs and Jégou, Hérve and Mikolov, Tomas. (2016). *fastText*. distributed via `https://fasttext.cc/docs/en/language-identification.html`.

Lui, Marco. (2017). *langid.py*. distributed via GitHub: `https://github.com/saffsd/langid.py`.

van Noord, Gertjan. (1997). *TextCat*. University of Groningen: `https://www.let.rug.nl/vannoord/TextCat/`.