FastSpell: the LangId Magic Spell

**Marta Bañón, Jaume Zaragoza-Bernabeu,
Gema Ramírez-Sánchez, Sergio Ortiz-Rojas**
Prompsit Language Engineering, S.L., Spain
{mbanon, jzaragoza, gramirez, sortiz}@prompsit.com

Abstract

Language identification is a crucial component in the automated production of language resources, particularly in multilingual and big data contexts. However, commonly used language identifiers struggle to differentiate between similar or closely-related languages. This paper introduces FastSpell, a language identifier that combines fastText (a pre-trained language identifier tool) and Hunspell (a spell checker) with the aim of having a refined second-opinion before deciding which language should be assigned to a text. We provide a description of the FastSpell algorithm along with an explanation on how to use and configure it. To that end, we motivate the need of such a tool and present a benchmark including some popular language identifiers evaluated during the development of FastSpell. We show how FastSpell is useful not only to improve identification of similar languages, but also to identify new ones ignored by other tools.

Keywords: Language identification, Language resource creation, Multilingual content

1. Introduction

Language identification is at the core of many NLP pipelines and an essential component in the automatic production of language resources. A bad choice of the technology used to perform language identification may have a crucial impact on the rest of the pipeline and the final results, especially in big data and multilingual data contexts. In such contexts, a complex variety of languages need to be language-identified at scale and three factors are of great importance: number of languages covered, accuracy and speed. However, the choice of language identifiers is mostly based on language coverage as this is usually a crucial factor in the decision. Other factors may be disregarded and, once the choice is made, the rest of the pipeline will need to cope with the possible mistakes made by the selected language identifier.

In this paper, we present FastSpell, a language identifier that reviews and complements prior language identification processes. It seeks for a compromise between speed and quality, with special focus on similar languages and language varieties. FastSpell requires a language to focus on, which we define as the `targeted language`, usually coming from a previous language identifier. FastSpell double-checks that targeted language, paying special attention to languages that are often confused with it. To that end, FastSpell first asks fastText[1] to give a prediction. Then, only if the language predicted by fastText falls into a group of languages similar to the targeted language, FastSpell refines its decision by performing extra checks with the Hunspell[2] spell-checker. This allows to double-check the prediction made by fastText as well as to discriminate better between similar languages, identify new languages or group them.

FastSpell, distributed under the GPLv3 license,[3] is officially supported and maintained inside the Bitextor[4] and Monotextor[5] pipelines, two free/open-source tools used to produce parallel and monolingual corpora from web-crawled content. It runs in the corpus cleaning phase of these two tools to refine the decisions made by any language identifier applied earlier. It has been used to produce corpora in projects such as ParaCrawl,[6] MaCoCu,[7] and HPLT,[8] all complex in the number and variety of languages and amounts of data processed.

In the following sections, we explain the motivation of building FastSpell (section 2) and why it relies on fastText (section 3), which was chosen after a careful evaluation of available tools. This evaluation, designed and maintained up to date as an independent benchmark, focuses on particularly challenging cases, that is, deciding between similar languages, and reports both accuracy and speed performance. Section 4 describes the FastSpell algorithm while section 5 explains how to use and configure it. Finally, section 6 draws some conclusions and future working plans.

---

[1] https://fasttext.cc/

[2] http://hunspell.github.io/

[3] https://github.com/mbanon/fastspell/blob/main/LICENSE

[4] https://github.com/bitextor

[5] https://github.com/bitextor/monotextor

[6] https://paracrawl.eu

[7] https://macocu.eu

[8] https://hplt-project.org

2. Why FastSpell?

FastSpell was initially developed as part of the code of the ParaCrawl series of projects aiming at deriving parallel data from web-crawled content (Bañón et al., 2020). Language identification of web-crawled data, usually very noisy (Kreutzer et al., 2022), is a necessary step to derive language-specific textual corpora from them. After manual inspections of the parallel sentences produced in ParaCrawl, we found several issues with CLD2,[9] the tool used at the moment to identify language at document level, then transferred to sentence level:

- Closely related languages often get mixed up. Especially if one of them has significantly more resources, for example, Spanish and Galician or the Bokmål and Nynorsk variants of Norwegian.

- Text containing all or mostly uppercase letters is very often classified as the highest-resourced language using a particular writing system. Many Latin script languages end up badly identified as English, Spanish or French and Cyrillic script languages as Russian.

- Languages that use two or more scripts are usually identified with just one of them. For example, although Serbo-Croatian languages can technically be written in Cyrillic and Latin, Cyrillic gets most of the time classified as Serbian and Latin as Croatian. While these are the mostly used scripts for those languages, this just does not cover all cases well enough, e.g. Serbian written in Latin.

FastSpell was developed to be able to cope with these issues and refine the decisions made by CLD2 at the beginning of the pipeline.

3. Benchmarking Language Identifiers

FastSpell starts by launching automatic language identification over a text, sentences in our case. To be able to make an informed decision on which tool was better suited for this task, we benchmarked several language identification tools (see Table 1) focusing on performance (runtime) and accuracy (F-score). This benchmark is constantly evolving to incorporate new tools and languages and is publicly available at `https://github.com/mbanon/benchmarks` with results available at `https://tinyurl.com/2u48kycz`.

The benchmark includes a diverse set of languages added in two batches according to projects needs. The first batch, introduced during the

[9]`https://github.com/CLD2Owners/cld2`

ParaCrawl project, covered Spanish (es), Galician (gl), Catalan (ca), Danish (da), Norwegian Bokmål (nb) and Norwegian Nynorsk (nn). The second one, introduced during the MaCoCu project, included Bulgarian (bg), Czech (cz), Greek (el), Macedonian (mk), Romanian (ro), Slovak (sk), Slovene (sl), Albanian (sq), Maltese (mt), Turkish (tr), Bosnian (bs), Montenegrin (me), Croatian (hr), and Serbian (sr). For convenience, we grouped several times Croatian, Bosnian, Serbian and Montenegrin under the Serbo-Croatian (hbs) macrolanguage.

	F1 scores				
Lang	pyCLD2	pyCLD3	fastText 176	HeLI	FastSpell
es	0,890	0,932	0,929	**0,957**	0,954
gl	0,866	0,903	0,808	**0,939**	0,800
ca	0,904	0,935	0,951	**0,961**	0,935
da	0,914	0,868	0,824	**0,930**	0,799
nb	0,713	-	-	**0,724**	0,675
nn	0,781	-	0,433	0,787	**0,810**
bg	0,939	0,959	0,979	0,972	**0,990**
cs	**0,965**	0,908	0,918	0,941	0,962
el	0,976	0,964	**1,000**	0,991	**1,000**
mk	0,883	0,981	**0,993**	0,982	0,985
ro	0,960	0,947	0,970	**0,988**	0,975
sk	**0,966**	0,934	0,917	0,952	0,937
sl	0,925	0,887	0,847	**0,942**	0,880
sq	0,981	0,986	**0,993**	0,992	0,990
mt	0,984	0,976	0,896	**0,996**	0,914
tr	0,978	0,985	**0,990**	0,983	0,988
bs	**0,443**	0,336	0,416	-	0,370
me	-	-	-	-	**0,458**
hr	**0,641**	0,423	0,557	-	0,541
sr	0,529	0,327	**0,565**	-	0,493
hbs	0,940	0,941	0,922	0,982	**0,983**
Avg. Runtime	**0,011**	0,097	0,019	2,688	1,076

Table 1: F1 scores and average runtime for all the languages and some of the language identification tools benchmarked. Best scores in bold.

For each language in each batch, we built a gold-standard corpus made of sentences. We used SETIMES[10] for Bosnian, Macedonian, Albanian, Serbian and Turkish, MontenegrinSubs (Božović et al., 2018) and texts from the Government of Montenegro webpage[11] for Montenegrin and Paracrawl human annotations (Ramírez-Sánchez et al., 2022) for the other languages. We also built an "anti-gold standard" by combining all sentences in a batch and excluding the sentences for the targeted language (for example, the anti-gold standard for Norwegian Nynorsk included the sentences in the Danish and Norwegian Bokmål gold-standards).

The benchmarked tools and versions are pyCLD2,[12] pyCLD3,[13] langid,[14] FastLang,[15]

[10]`https://opus.nlpl.eu/SETIMES.php`
[11]`https://www.gov.me/`
[12]`https://github.com/aboSamoor/pycld2`
[13]`https://github.com/bsolomon1124/pycld3`
[14]`https://github.com/saffsd/langid.py`
[15]`https://spacy.io/universe/project/spacy_fastlang`

langdetect,[16] NLTK,[17] GuessLanguage,[18] FastText (lid.176 model compressed and binarized version) and HeLI-OTS (Jauhiainen et al., 2022). The same benchmarking was performed combining Hunspell with PyCLD2 and PyCLD3 instead of fastText.

As shown in Table 1, in terms of F-scores and runtime, CLD2 would have been the best candidate for language identification in FastSpell but, since it was already used in previous steps in the Bitextor production pipeline, we decided to use a different and complementary tool. HeLIOTS would have also been a good choice, but it was, at the moment, up to 100 times slower than fastText on average. For these reasons, fastText was selected as the language identification tool to be included in FastSpell.
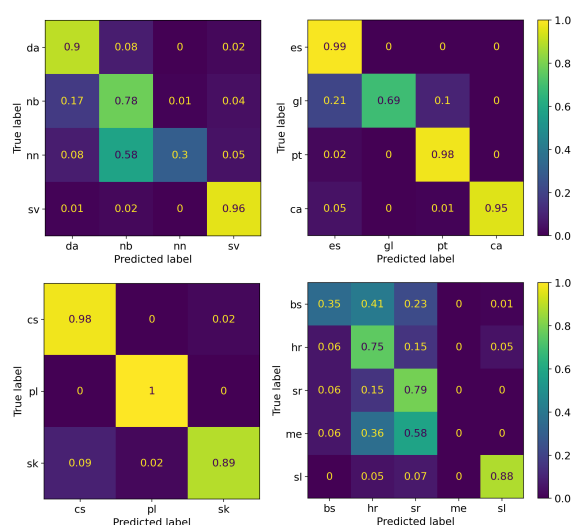

Figure 1: fastText confusion matrices for some groups of similar languages.

4. The FastSpell Spell

In order to address the issues mentioned in section 2 with similar languages being confused, FastSpell makes a two-step decision using two different, well-known tools: *fastText* and *Hunspell*.

FastText (Joulin et al., 2016a,b) is a free/open-source library for fast text representation and classification, developed by Meta.[19] In particular, the language identification model used in FastSpell is the `lid.176.bin` model, trained on data from Wikipedia, Tatoeba and SETIMES and able to recognize up to 176 languages.[20]

Using fastText alone to make this second prediction was deemed insufficient to make good predictions in many cases. As shown in Figure 1, fastText has issues discriminating among similar languages which it confuses frequently (for example, Galician with Spanish), struggling on Norwegian Nynorsk with Norwegian Bokmål, or making inaccurate predictions for many South-Slavic closely-related languages. In other cases, predictions look quite good, but still not completely accurate as in the case of Slovak. Thus, a second step relying on spell-checking results was introduced.

The technology chosen to perform spell-checking is Hunspell, the well-known free/open-source spell checker and morphological analyser widely used in also well-known tools, such as Mozilla Firefox, Google Chrome or the LibreOffice suite. For spell checking, Hunspell uses affix files and dictionaries, that can be obtained from different sources[21] independently or be custom-built by users.

FastSpell focuses on a given language (the targeted language), that is provided as a parameter. Given a text (usually, a sentence), FastSpell will first predict its language by using fastText. For efficiency, only if the predicted language is the targeted language or a similar language according to a configurable list (see section 2), FastSpell will try to refine the fastText prediction by checking the sentence spelling with Hunspell for the targeted language and its similar languages. Depending on the ratio of spelling errors for each of the spell-checked languages, FastSpell will confirm the targeted language as the winner or replace it by the similar language with the lowest number of spelling errors. This is true for the *aggressive* mode . However, in the *conservative* mode, when there is not a clear winner after the spell-checking step, the language for that text can be tagged as "unknown". The full algorithm is shown in Appendix B.

5. Using FastSpell

FastSpell can be installed to be used as a CLI tool or as a Python package. In both cases, besides a text, the only parameters that are needed are the targeted language (i.e. the language predicted by a prior language identification tool and the mode (either aggressive or conservative). In the example be-

---

[16] https://github.com/Mimino666/langdetect

[17] https://www.nltk.org/api/nltk.classify.html#nltk.classify.textcat.TextCat.guess_language

[18] https://pypi.org/project/guess_language-spirit

[19] https://opensource.fb.com

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

[21] https://github.com/LibreOffice/dictionaries, https://github.com/wooorm/dictionaries/, https://extensions.libreoffice.org/?q=spellchecker, http://hlt.sztaki.hu/resources/hunspell/

low, FastSpell receives English (`en`) as the targeted language and conservative (`cons`) as the mode. For the first sentence (`Hello, world`), FastSpell outputs English as the predicted language, for the second one (`Hola, mundo`), it outputs Spanish (`es`), refining the a priori prediction.

```
from fastspell import FastSpell
fsobj=FastSpell("en", mode="cons")
fsobj.getlang("Hello, world")
#'en'
fsobj.getlang("Hola, mundo")
#'es'
```

Out-of-the-box FastSpell comes pre-configured for several targeted languages and the necessary linguistic resources (fastText model and Hunspell dictionaries) are installed as a dependency.[22] This default configuration mainly focuses on the languages of interest of the three projects in which FastSpell has been used: Paracrawl (Bañón et al., 2020), MaCoCu (Bañón et al., 2022) and HPLT (Aulamo et al., 2023). Besides addressing the issue of identifying similar or closely related languages (for example, Spanish and Galician), we also provide a solution to identify single languages from a macrolanguage (languages in the Serbo-Croatian family or Norwegian Bokmål/Nynorsk), languages not supported by the current fastText model (for example, Montenegrin) and languages that in principle are not similar but that fastText seems to confuse (for example, Somali and English). The pre-configured targeted languages and the ones similar to each of them are shown in Appendix A.

Custom configuration The default configuration of FastSpell might not be suitable in some cases, for example, when it does not support the targeted language by default, when there's a need to change the pre-defined similar languages for a targeted language, or when a different Hunspell dictionary needs to be used. FastSpell is easily customizable in these cases, only requiring to modify some configuration text files located in the fastspell/config directory.

One of those files is `similar.yaml` (whose first lines can be seen in Figure 2), a file containing the pre-defined targeted languages and their associated similar languages. These languages (see the full list in Appendix A) will be double-checked with Hunspell after being predicted by fastText. Adding or removing a new language from this file will activate or deactivate the spellchecking step in case it matches the criteria of the FastSpell algorithm (see section 4). The list of similar languages associated to a targeted language can also be modified. Note that the list of similar languages is not necessarily

```
#Targeted langs(keys) dict for
#mistakeable languages (values)
similar:
    af: [nl, de, af]
    az: [tr, az]
    be: [ru, uk, be]
    bg: [mk, ru, bg]
    bs: [hr, sr, sl, bs]
    ca: [es, oc, ca]
    cs: [sk, cs]
    cy: [ga, en, cy]
```

Figure 2: First lines of the default *similar.yaml* file

symmetrical: a targeted language *A* may have language *B* as similar, but a targeted language *B* may not have language *A* as similar instead. Indeed, it may not even be necessary to have language *B* configured as a targeted language.

The `hunspell.yaml` file, which contains Hunspell-related information such as the path to the location of Hunspell dictionaries and the name of the dictionary for each language, may also be configured. This allows some flexibility in usage, for example, to assemble a dictionary for a macrolanguage and use it as a targeted or similar language. This is the case of Serbo-Croatian (hbs), for which a single Hunspell dictionary gathers together Serbian, Croatian and Bosnian in FastSpell.

FastSpell results, included in 1, show how some languages not supported or badly supported by other tools can be more reliably identified using FastSpell, as is the case of Montenegrin or Norwegian Nynorsk.

6. Conclusions and Future Work

We have presented FastSpell, a second-opinion language identifier that reviews and refines decisions made by a previous language identifier from which a targeted language is set. FastSpell is able to distinguish better between closely-related languages or to discover new languages or language varieties not predicted by a language identifier (for example, Norwegian Nynorsk), usually hidden or confused with a larger-resource language (for example, Norwegian Bokmål). There is still room for enhancements that could be added to FastSpell, for example:

- Exploring possible replacements for the current fastText model, lid.176.bin, by, for example, the 201-language model introduced in (Burchell et al., 2023).

- Exploring faster implementations of fastText such as fasterText.[23]

- Curating Hunspell dictionaries for languages not having publicly available dictionaries to extend FastSpell's language support. For example, Pashto seems to be frequently misslabelled as Arabic, but we have not been able to find any Pashto dictionaries that can be integrated into FastSpell. A similar situation is found with Sindhi and Farsi.

- Exploring proper tokenization and/or stemming of sentences before spellchecking with Hunspell to improve language identification accuracy after it. Since Hunspell is based in dictionaries of known words, spellchecking stems instead of full words will probably result in less false negatives, specially in inflected languages.

- Adding non-targeted identification, that is, not focusing on a given language (by applying extra checks only for it and its similar languages), but applying the extra checks for any identified language. This is expected to be substantially slower, but will provide more accurate language identification.

- Exploring using different error thresholds, depending on the targeted language.

- Write a Hunspell-like engine which is capable to process more than one language at once to avoid repeated checks.

Current language extensions and enhancements to FastSpell have been made inside the HPLT project, still ongoing, which has recently produced language resources for more than 75 languages (de Gibert et al., 2024). Among those, there are many that have different variants or are closely-related languages which have benefited from FastSpell refinements.

7. Bibliographical References

Mikko Aulamo, Nikolay Bogoychev, Shaoxiong Ji, Graeme Nail, Gema Ramírez-Sánchez, Jörg Tiedemann, Jelmer van der Linde, and Jaume Zaragoza. 2023. HPLT: High performance language technologies. In *Proceedings of the 24th Annual Conference of the European Association for Machine Translation*, pages 517–518, Tampere, Finland. European Association for Machine Translation.

Marta Bañón, Pinzhen Chen, Barry Haddow, Kenneth Heafield, Hieu Hoang, Miquel Esplà-Gomis, Mikel L. Forcada, Amir Kamran, Faheem Kirefu, Philipp Koehn, Sergio Ortiz Rojas, Leopoldo Pla Sempere, Gema Ramírez-Sánchez, Elsa Sarrías, Marek Strelec, Brian Thompson, William Waites, Dion Wiggins, and Jaume Zaragoza. 2020. ParaCrawl: Web-scale acquisition of parallel corpora. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4555–4567, Online. Association for Computational Linguistics.

Marta Bañón, Miquel Esplà-Gomis, Mikel L. Forcada, Cristian García-Romero, Taja Kuzman, Nikola Ljubešić, Rik van Noord, Leopoldo Pla Sempere, Gema Ramírez-Sánchez, Peter Rupnik, Vít Suchomel, Antonio Toral, Tobias van der Werff, and Jaume Zaragoza. 2022. MaCoCu: Massive collection and curation of monolingual and bilingual data: focus on under-resourced languages. In *Proceedings of the 23rd Annual Conference of the European Association for Machine Translation*, pages 303–304, Ghent, Belgium. European Association for Machine Translation.

Laurie Burchell, Alexandra Birch, Nikolay Bogoychev, and Kenneth Heafield. 2023. An open dataset and model for language identification. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 865–879, Toronto, Canada. Association for Computational Linguistics.

Tommi Jauhiainen, Heidi Jauhiainen, and Krister Lindén. 2022. HeLI-OTS, off-the-shelf language identifier for text. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 3912–3922, Marseille, France. European Language Resources Association.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016a. Fasttext.zip: Compressing text classification models.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016b. Bag of tricks for efficient text classification.

Julia Kreutzer, Isaac Caswell, Lisa Wang, Ahsan Wahab, Daan van Esch, Nasanbayar Ulzii-Orshikh, Allahsera Tapo, Nishant Subramani, Artem Sokolov, Claytone Sikasote, Monang Setyawan, Supheakmungkol Sarin, Sokhar Samb, Benoît Sagot, Clara Rivera, Annette Rios, Isabel Papadimitriou, Salomey Osei, Pedro Ortiz Suarez, Iroro Orife, Kelechi Ogueji, Andre Niyongabo Rubungo, Toan Q. Nguyen, Mathias Müller, André Müller, Shamsuddeen Hassan Muhammad, Nanda Muhammad, Ayanda Mnyakeni, Jamshidbek Mirzakhalov, Tapiwanashe Matangira, Colin Leong, Nze Lawson, Sneha Kudugunta, Yacine Jernite, Mathias Jenny, Orhan Firat, Bonaventure F. P. Dossou, Sakhile Dlamini, Nisansa de Silva, Sakine Çabuk Ballı, Stella Biderman, Alessia Battisti, Ahmed Baruwa, Ankur Bapna, Pallavi Baljekar, Israel Abebe Azime, Ayodele Awokoya, Duygu Ataman, Orevaoghene Ahia, Oghenefego Ahia, Sweta Agrawal, and Mofetoluwa Adeyemi. 2022. Quality at a glance: An audit of web-crawled multilingual datasets. *Transactions of the Association for Computational Linguistics*, 10:50–72.

Gema Ramírez-Sánchez, Marta Bañón, Jaume Zaragoza-Bernabeu, and Sergio Ortiz Rojas. 2022. Human evaluation of web-crawled parallel corpora for machine translation. In *Proceedings of the 2nd Workshop on Human Evaluation of NLP Systems (HumEval)*, pages 32–41, Dublin, Ireland. Association for Computational Linguistics.

8. Language Resource References

Petar Božović, Tomaž Erjavec, Jörg Tiedemann, Nikola Ljubešić, and Vojko Gorjanc. 2018. Opus-montenegrinsubs 1.0: First electronic corpus of the montenegrin language.

Ona de Gibert, Graeme Nail, Nikolay Arefyev, Marta Bañón, Jelmer van der Linde, Shaoxiong Ji, Jaume Zaragoza-Bernabeu, Mikko Aulamo, Gema Ramírez-Sánchez, Andrey Kutuzov, Sampo Pyysalo, Stephan Oepen, and Jörg Tiedemann. 2024. A new massive multilingual dataset for high-performance language technologies.

A. Pre-configured similar languages

Targeted language	Similar languages
Afrikaans	Dutch, German
Azerbaijani	Turkish
Belarusian	Russian, Ukrainian
Bulgarian	Macedonian, Russian
Bosnian	Croatia, Serbian, Slovene
Catalan	Spanish, Occitan
Czech	Slovak
Welsh	Irish, English
Danish	Norwegian Bokmål, Swedish
Spanish	Galician, Catalan
Farsi	Arabic, Azerbaijani
Irish	Welsh, English
Galician	Spanish, Portuguese
Serbo-Croatian (Latin)	Slovene
Serbo-Croatian (Cyrillic)	Russian, Macedonian, Bulgarian
Hindi	Marathi, Nepali
Croatian	Bosnian, Serbian, Slovene
Indonesian	Malay
Icelandic	Danish, Norwegian Bokmål, Norwegian Nynorsk, Swedish
Hebrew	Yiddish
Kazakh	Kyrgyz, Tatar, Russian
Kyrgyz	Russian, Kazakh, Tatar, Mongolian
Latvian	Lithuanian
Montenegrin	Croatian, Serbian, Slovene, Bosnian
Macedonian	Bulgarian, Serbian, Russian
Mongolian	Russian, Kyrgyz, Bulgarian
Marathi	Hindi
Malay	Indonesian
Norwegian Bokmål	Danish, Swedish, Norwegian Nynorsk
Nepali	Marathi, Hindi
Dutch	Afrikaans
Norwegian Nynorsk	Norwegian Bokmål, Danish, Swedish
Norwegian	Danish, Swedish, Norwegian Nynorsk
Portuguese	Spanish, Galician
Russian	Ukrainian, Bulgarian
Slovak	Czech, Polish
Slovene	Serbian, Croatian, Bosnian
Somali	English, Finnish, Welsh, Kannada
Serbian	Bosnian, Croatian, Slovene, Montenegrin
Swedish	Danish, Norwegian Bokmål
Tatar	Kazakh, Kyrgyz, Russian
Ukrainian	Belarusian, Russian, Macedonian, Bulgarian
Urdu	Farsi, Arabic
Uzbek	Turkish
Yiddish	Hebrew

Table 2: FastSpell preconfigured similar languages

B. The FastSpell Algorithm

---

function GETLANGUAGE(target_lang, sentence, strategy)
    similar_langs ← SimilarLanguages(target_lang) ∪ {target_lang}
    pred_FT ← FastText(lowercase(sentence))
    if |similar_langs|=1 **then**   ▷ similar_langs={target_lang} only
        return pred_FT
    end if
    if pred_FT ∉ similar_langs **then**
        return pred_FT
    end if
    candidate_langs ← ∅
    best_error_rate ← 0
    for all sim_lang ∈ similar_langs **do**
        relevant_tokens ← remove_uppercased(remove_non_alphabetic(tokens(sentence)))
        correct_tokens ← collect_correct_tokens(Hunspell(relevant_tokens, sim_lang))
        error_rate ← 1 − (|correct_tokens|/|relevant_tokens|)
        if error_rate ≤ error_threshold **then**
            candidate_langs ← candidate_langs ∪ {sim_lang}
            if error_rate < best_error_rate **then**
                best_error_rate ← error_rate
            end if
        end if
    end for
    refined_candidates ← candidates_with_lowest_error_rate(candidate_langs)
    if |refined_candidates| = 1 **then**
        return first(refined_candidates)   ▷ The first and only language in the set
    else if |refined_candidates| > 1 **then**
        if strategy = aggressive **then**
            if target_lang in refined_candidates **then**
                return target_lang
            else if pred_FT in refined_candidates **then**
                return pred_FT
            else
                return first(refined_candidates)   ▷ The first language will do in case of a draw
            end if
        else if strategy = conservative **then**
            if target_lang ∈ refined_candidates ∧ best_error_rate = 0 **then**
                return target_lang
            else
                return unknown_lang   ▷ A special code
            end if
        end if
    else if |refined_candidates|=0 ∨ |candidate_langs|=0 **then**
        if strategy = aggressive **then**
            return pred_FT
        else if strategy = conservative **then**
            return unknown_lang
        end if
    end if
end function

Figure 3: The FastSpell Algorithm