Methods, Models and Tools for Improving the Quality of Textual Annotations

Maria Teresa Artese and Isabella Gagliardi *

IMATI-CNR (National Research Council), Via Bassini, 15, 20133 Milan, Italy; artese@mi.imati.cnr.it
* Correspondence: gagliardi@imati.mi.cnr.it

Abstract: In multilingual textual archives, the availability of textual annotation, that is keywords either manually or automatically associated with texts, is something worth exploiting to improve user experience and successful navigation, search and visualization. It is therefore necessary to study and develop tools for this exploitation. The paper aims to define models and tools for handling textual annotations, in our case keywords of a scientific library. With the background of NLP, machine learning and deep learning approaches are presented. They allow us, in supervised and unsupervised ways, to increase the quality of keywords. The different steps of the pipeline are addressed, and different solutions are analyzed, implemented, evaluated and compared, using statistical methods, machine learning and artificial neural networks as appropriate. If possible, off-the-shelf solutions will also be compared. The models are trained on different datasets already available or created ad hoc with common characteristics with the starting dataset. The results obtained are presented, commented and compared with each other.

Keywords: multilingual datasets; word embedding models; sequence2sequence; LSTM; neural networks; machine learning algorithms; semantic relatedness; syntactic similarity

1. Introduction

Creating large archives or repositories is becoming easier with the availability of open data and tools to manage them. Sometimes it can be useful to integrate different data into the same repository. Textual content should be enriched with semantic tags, either provided by catalogers or extracted automatically, to be effectively usable. To achieve an efficient and effective use of these annotations (in this paper, tags, keywords, and textual annotations are intended as synonyms and therefore used interchangeably) in information retrieval, sentiment analysis, or recommendation systems, they must be general enough to represent many items, but not too many. They should be specific enough to represent some items, but not too few. Methods, models and tools are required for their cleaning, harmonization and categorization, in other words, to optimize their use. The real challenge to face, then, concerns the quality of the data and the users’ satisfaction in being able to respond to their need for information.

This paper focuses on models and tools to improve the quality of data, based on natural language processing with machine learning and deep learning approaches. The main problems with keywords are:

- The use of different terms to express (more or less) the same concept, or spelling variants;
- The presence of tags expressed in different languages;
- The extreme specialization of the terms used;
- The lack of context;
- Sometimes outright misspellings.

The first two issues are frequent when integrating different datasets. The pipeline to improve the quality of textual annotations is addressed: the first step is the automatic detection of the language in which a text is written, in our case very short.
Then, the automatic detection and correction of spelling errors and the syntactic/semantic correlation between terms are dealt with. Classic preprocessing steps is performed if useful and necessary.

The following methods, models and tools are integrated into the pipeline to take full advantage of their specifics and capabilities:

(i) Machine learning and deep learning methods to identify both tag language and the presence of spelling errors;
(ii) Word embeddings models to capture the semantics of words and their context;
(iii) Clustering algorithms to identify/group semantically related terms;
(iv) Syntactic similarity methods between strings to strengthen the identification and correction of spelling errors.

In this paper, they are applied to multilanguage texts in the context of digital libraries and scientific articles produced by researchers at a research institute: the tags suffer from all the problems previously exposed. The languages processed are Italian, English, French, German, and Spanish. To avoid overfitting, due to too small datasets, the models are trained on different datasets found on the web or created ad hoc, which share some characteristics with the source dataset. We present the tools developed and the data used in the experiments, detailing the solutions adopted and the various options tested in the runs. The results obtained are presented and discussed, compared with the state of the art.

The paper is structured as follows: Section 2 quickly outlines related works, then our approach is described in full detail. Section 4 presents the experimentation performed, indicating the datasets used for training and evaluation, the methods implemented, with some preliminary results and a brief discussion to comment on them, including a comparison of the different approaches. Conclusions and future work complete the paper.

2. Related Works

This article deals with several aspects related to the management of annotations in multilingual archives that allow us to make the best use of tags. These tags, already defined and associated with the data, are improved by unifying them, harmonizing them, integrating methods to evaluate their syntactic or semantic similarity. In particular, we deal with three aspects: the automatic recognition of the language in which a text, or rather a string of characters, is written, the automatic identification of writing errors and the calculation of syntactic and semantic similarity.

Automatic language identification is a topic that has received a lot of attention over the years in various fields such as machine translation, speech synthesis and information retrieval, to name a few. In the text processing pipeline, language identification is one of the first activities. For example, think of Google Translate (https://translate.google.com/ accessed on 30 January 2022) or DeepL (https://www.deepl.com/translator accessed on 30 January 2022), which offer the possibility to automatically recognize the language of the text to be translated. A 2019 survey [1] provides a very broad overview of the different features, methods and models used in the literature for language identification, in different domains. According to the authors, under controlled conditions, such as limiting the number of languages to a small set of Western European languages and using long, grammatical and structured text such as government documents as training data, it is possible to achieve near-perfect accuracy. The problem is still complex when dealing with very short texts, with misspelling errors or mixed language.

Regarding automatic error recognition, a variety of papers related to specific aspects of the problem can be found in the literature. Ref. [2] deals with the specific problem of medical records in Persian, capable of identifying nonwords errors. In [3], the authors deal with the Thai language and face the problem of handling words not present in the dictionary. The use of word embedding models, such as word2vec, GloVe or Bert, together with edit distance, is a solution adopted by [4], in which the pretrained models proved their potential in the misspelling correction task. The literature is very extensive regarding the problem of identifying offensive texts. In 2019, one of SemEval’s tasks was the use of
ensemble machine learning to detect hate speech [5]: the paper highlighted the necessity of having more training examples, ranging from more positive to more negative. Simpler models have understandable features, while complex models obtain better results.

Artificial neural networks have been successfully applied to the understanding of text. In [6], Goldberg surveys neural network models from the perspective of natural language processing research, in an attempt to bring natural-language researchers up to speed with the neural techniques. Different neural network methods for automatic language identification have been proposed [7–9]. These papers agree that short texts affect the quality of the results, as well as the size of the training datasets.

In [10], the authors provide a systematic overview of the approaches developed so far. To account for context, LSTM [11] and sequence2sequence [12] architectures were used for automatic error correction. The lack of resources for specific languages requires special consideration, as in [13], for Indic languages. Ref. [14] makes use of sequence-to-sequence (seq2seq) models for spelling correction in the Turkish language, on an artificially created misspelling dataset.

In [15], the authors introduce a general framework of syntactic similarity measures for matching short text. Word embedding models such as word2vec [16,17] or BERT [18] have greatly improved text management. The authors of [19] proposed a learning-based approach for automatic construction of domain glossary from source code and software documentation, using word embedding models integrated with lexical similarity. The integration of word embedding models with clustering algorithms is now being studied by different research groups, mainly with classification purposes and/or in specific contexts (for example [20,21]), due to its ability to extract semantic and discriminative keywords.

3. Materials and Methods

The use of keywords has been identified and studied extensively in the field of information retrieval. Typically, when manually associated by catalogers/researchers, the tags are very specific, to limit the set of results to only highly relevant items. This is known as “precision”-oriented search. However, sometimes, web users are interested in the topic in a broader sense, aiming to retrieve not only specific information, but also other elements around it. In this case, we can talk about a system that operates in a “recall”-oriented way.

Our general aim is to define a pipeline able to improve the quality of tags and optimize their use in online catalogs, making the keywords associated by cataloguers more incisive, inclusive, discriminating and relevant, in a word, fit for the purpose.

The approach presented in the paper consists of many steps (see Figure 1), which offer different types of similarity between tags, both syntactic and semantic.

3.1. Data

The data on which we tested our approach consisted of scientific articles produced by researchers in a research institute. It contained 4000 scientific papers in Italian, English, French, German, and Spanish. The cataloguers/researchers associated 7000 different single or compound keywords with the articles. This dataset is called LIBRARY.

Usually, in the natural language processing (NLP) pipeline, the first operation is pre-processing, composed of tokenization, cleaning, lemmatization, stemming, and part-of-speech tagging of the string that constitutes the input. The output of the process consists of terms, single or compound words, to be used in the following steps. In our case, the dataset was already composed only of simple terms, thus it was already tokenized and cleaned up. The preprocessing step was therefore not performed on the library dataset but was done on the training support datasets, as briefly described later.
3.1. Data
The data on which we tested our approach consisted of scientific articles produced by researchers in a research institute. It contained 4000 scientific papers in Italian, English, French, German, and Spanish. The cataloguers/researchers associated 7000 different single or compound keywords with the articles. This dataset is called LIBRARY.

3.2. Task 1: Automatic Language Identification
Keywords associated with collections of multilingual textual objects may be available in the same language in which the object itself is described or cataloged, or in the language of the country in which it is cataloged. For example, some Italian cataloging standards require that the metadata associated with each item be in Italian, regardless of the language of the object itself. Therefore, when we are dealing with multilingual collections, for example, books and journals of a library, or scientific production of a university or research institute, the tags can be indifferently in English, Italian, French, etc. Automatic language identification, using statistical techniques, machine learning or deep learning methods, is therefore necessary. In the following, ready-made methods are presented, tested and compared with others implemented from scratch.

3.3. Task 2: Misspelling Identification and Correction
This paper is designed for specific and specialized terms related to research activities. Tags in scientific articles can be extremely technical and fall outside of common lexicons. In addition, keywords are subject to frequent changes and updates. For these reasons, using standard methods such as presence/absence in WordNet (https://wordnet.princeton.edu/ accessed on 30 January 2022) [22] or lists of terms alone is not sufficient to determine the presence of spelling errors. In this paper, both statistical methods and deep-learning-based solutions are presented. Statistical methods lead to the identification of the presence of an error and suggestions for its correction, through statistical and probability functions, also related to the specific language model. The deep learning method, trained on different training sets, identifies and corrects errors in a single step. We created synthetic datasets of noisy (and correct) words by collecting highly probable spelling errors and inducing noise in the clean corpus. To obtain accurate results, training requires many epochs and large datasets.

3.4. Task 3: Syntactic Similarity
This step involves computing a similarity matrix based on the syntactic similarity of the tags. From a semantic point of view, the use of singular/plural or the use of different terms—that may appear to the nonexpert as synonyms—conveys a (slightly) different meaning; when using the search engine instead, users will greatly appreciate a system able
to propose, in a clear and visible way, a syntactic clustering for terms that share a high number of letters.

Several methods of calculating similarity can be used, as those presented in [15], e.g., Levenshtein, Damerau, Jaro, Jaro–Winkler, etc. In this paper, after a series of tests, we chose the Jaro–Winkler algorithm, which measures the difference between 2 strings, then normalizes it, with values ranging from 1 (when two strings are identical) to 0 when they are completely dissimilar. The choice of this measure is because of the ease of understanding the results and the fact that it gives results in the same range as semantic similarity, described later, and is therefore seamlessly integrable.

3.5. Task 4: Semantic Relatedness

Catalogers/researchers who associate tags with their items may choose terms that are similar to others while making use of different words. This may be due to the need to differentiate concepts, to the different background of the catalogers/researchers who use distinct languages and jargon, or to the fact that data come from different archives. While it is correct not to replace or suggest changes to tags, it is fundamental that the search engine can retrieve all items that are semantically related.

The semantic relatedness of terms is based on both the term itself and the context in which this term is used. Here, we present a method to compute semantic correlation based on word embedding. Particular attention has been paid to compound tags, which are not present in the compound form in the word embedding model, but as individual components. In this case, a new vector is computed as the average of the composing vectors.

In this work, we are interested in the semantic correlation between tags, in a broad sense. As a result, we did not investigate the type of correlation, e.g., synonyms, antonyms, etc. (see also [23]), being interested in all without the need of distinction, using the cosine between the vectors representing the tags as a similarity measure.

3.6. The Use of LIBRARY Tags

This section describes how Figure 1 is applied to the keywords used in the experimentation. The tags in the LIBRARY dataset are the input to the pipeline specifically defined to deal with them. These tags are used without further processing and first undergo task 1 (T1), i.e., language identification, using previously trained ad hoc models. The output of this task is the association of the language to each tag. For the training of the models, we used large multilingual datasets, such as Wikipedia or papers from scientific journals scraped from Scopus or ArXiv. In this case, individual words, obtained by tokenizing sentences and removing stopwords, were annotated with the language and were the input for training.

Task 2 (T2) led to the identification and correction of misspellings. Again, models were trained on different datasets. LIBRARY tags were used only in the evaluation phase. Two steps were performed on the data: (i) if there were errors in the source data, the correct version of the tag was manually associated; (ii) the tags were artificially modified applying misspelling algorithms. The output of T2 were error-free tags.

At this point, the tags, definitely correct and accompanied by the language, were associated with similar terms, syntactically or semantically. For the latter activity, the language is crucial to translate the tags into vectors and create the similarity matrix. For syntactic similarity, which is based only on the characters present in each term, knowing its language is less important and could even be a limitation in the search for similar terms, in the presence of multilingual archives. The results of T3 and T4 were a list of tags, syntactically or semantically related.

In Section 4 the different steps are described and commented on, highlighting the results obtained.
4. Our Approach: Results and Discussions

Each task was trained and evaluated on separate datasets. The LIBRARY dataset, composed of single or compound keywords, did not require any preprocessing. The tags, exactly as entered by the catalogers/researchers were used. On the other datasets, used for training or evaluation and described below, several preprocessing steps were performed: (1) tokenization, identification of tokens, i.e., words, and (2) removal of stopwords.

4.1. Datasets and Models

The experiments conducted aimed at defining methods and models to improve the quality of the textual keywords within the LIBRARY dataset. It was necessary to identify the best features able to provide the ML and ANN models with the information to identify the language and possible spelling errors. In addition to the word itself, as a unicum (word unigram feature), several attempts were made to consider subsets of the word itself. In this paper, we report character n-grams, i.e., sets of n successive characters, with n = 3. This is a similar approach to a bag-of-words model, except that we are using characters and not words.

For each feature, the m most frequent elements (word unigram or character trigrams) were extracted, with m = 400, 800 and 1200. The constraints led to the actual extraction of features used for training and subsequent validation of the results (see Table 2, column features extracted).

These “objects”—data, models and algorithms—were used in the creation of the datasets, in the training and testing of the models and in the validation of the results on the data we were interested in:

- WIKIPEDIA datasets: title and abstract of Wikipedia articles, as stored in the dumps of 20210801 https://dumps.wikimedia.org/backup-index.html accessed on 30 January 2022, in the different languages considered;
- ArXiv dataset: set of all papers in the ArXiv repository [24] (about 1,700,000 papers). This dataset ensures the presence of specialized scientific terms that might be excluded from a standard vocabulary;
- PAPERS: papers harvested from Scopus databases in the languages considered. Only title, language and keywords, if any, were collected:
  - PAPERS: it contains titles as a single string, consisting of multiple words;
  - PAPERS_1GRAM: each word, excluding stopwords, was evaluated individually;
- Keywords taken from the catalog of the LIBRARY of interest;
- Pretrained WORD EMBEDDING MODELS: here, we used pretrained Wikipedia models using word2vec model (\textit{W2V}_{WIKI});
- Word embedding model trained on ArXiv dataset, available only for the English language (\textit{W2V}_{ARXIV}).

4.1.1. Language Identification

The starting dataset is the WIKIPEDIA datasets in the different languages considered, in this case, Italian, English, French, Spanish and German. Two experiments were conducted:

- WIKITXT: For each language, all words (in the title and abstract) in the annotated dataset for that language were kept, except for stopwords. This implies that the same word could appear annotated in multiple languages. For each language, 100,000 records were used, divided into training/test sets. From the training set, 386,228, 231,816, 229,483, 369,906 and 151,173 single words were extracted in English, Italian, French, Spanish and German, respectively. They were 65,714, 51,779, 45,178, 70,541, 44,121 when taken once. The total number of char trigrams extracted were 24,609 for English, 22,194 for Italian, 20,892 for French, 29,487 for Spanish and 23,017 for German.
- WIKI_NOOOV: From the WIKITXT dataset, the terms overlapping in different languages and those not present in the pretrained models of word2vec were eliminated. Using the same parameters as for WIKITXT, we obtained 14,584, 38,932, 30,465, 39,769, 47,853 single
words, and 9622, 25,737, 19,994, 26,202, 31,555 words taken once in English, Italian, French, Spanish and German, respectively. The total char trigrams extracted were 4508 for English, 3033 for Italian, 5231 for French, 4827 for Spanish and 7318 for German.

The machine learning and artificial neural network models were then trained on the WIKIPEDIA datasets split into training/test sets and validated both on the training set and test set and on a set of data taken from the LIBRARY we were working on.

4.1.2. Misspelling Identification and Correction

Datasets were artificially created, adding misspellings. We used two starting datasets and two misspelling algorithms. For the starting datasets, we extracted single terms as in the following: from the ARXIV dataset for English and from WIKIPEDIA (the same used above) for the other languages: this dataset is called GENERAL; from PAPERS_1GRAM terms.

The two misspelling algorithms mimicked the most frequent errors that are introduced when writing:
- Simple changes to a word, such as a deletion (removing a letter), a transposition (swapping two adjacent letters), a substitution (changing one letter for another), or an insertion (adding a letter). This algorithm is called EDIT1 and is based on the correction algorithm of Peter Norvig (http://norvig.com/spell-correct.html accessed on 30 January 2022).
- Replacement of a letter with another nearby on the keyboard or deletion of a letter. Small variations on the set of possible characters that were replaced gave rise to:
  - NEARBY (the one used in the tests reported here) replaces one letter with another nearby on the keyboard;
  - NEARBY_COMPLETE: one letter is replaced with zero, one or two nearby letters.

We applied the two misspelling algorithms to the GENERAL, PAPERS_1GRAM and LIBRARY (used for evaluation purpose only) datasets, obtaining circa 200,000 artificially misspelled words for each language. The same word was used several times to obtain large corpora.

4.2. Task 1: Automatic Language Identification

The need to identify the language of a tag arises from the fact that algorithms, methods, and models for misspelling identification and syntactic and semantic relatedness computation are language-dependent.

Experimentations of different methods for language identification were carried out on WIKI_NOOOV and WIKITXT, consisting of many thousands of individual terms, annotated with the language they belong to, using three different approaches:
(1) Off-the-shelf standard packages;
(2) Machine learning approach;
(3) Deep learning approach.

4.2.1. Off-the-Shelf Standard Packages

On the Internet, there are many sites for the translation of texts from one language to another, such as Google Translate (http://translate.google.com accessed on 30 January 2022) or DeepL (https://www.deepl.com/translator accessed on 30 January 2022), which make use of neural networks, very large datasets and the help of users to improve or correct the translation. There are several sites like Booking (https://www.booking.com/index.it.html accessed on 30 January 2022) or even Amazon, which offer automatically translated reviews and comments.

With the idea of using ready-made tools if the results were good enough, therefore without the need to implement others from scratch, python packages were tested, which make use of statistical models or pretrained machine learning methods, such as polyglot (https://github.com/saffsd/polyglot accessed on 30 January 2022), fasttext
(https://fasttext.cc/docs/en/support.html accessed on 30 January 2022) and langid (https://github.com/saffsd/langid.py accessed on 30 January 2022).

In Table 1, the accuracy of the off-the-shelf methods tested are reported, and the ensemble—in grey—is obtained as the majority of votes [25]. Results are reported for both WIKIPEDIA and PAPERS/PAPERS_1GRAM datasets. The methods were applied as is, without fine-tuning or retraining on the specific data used, which could have brought further improvement. It can be seen that the voting mechanism improves the accuracy of the results. FastText, the package based on Facebook’s text management tools, achieves the best results for all corpora.

Table 1. Accuracy of the off-the-shelf methods.

| Methods   | WIKITXT | WIKI_NOOOV | PAPER  | PAPERS_1GRAM |
|-----------|---------|------------|--------|--------------|
| Langid    | 0.569   | 0.513      | 0.721  | 0.518        |
| Polyglot  | 0.368   | 0.413      | 0.567  | 0.521        |
| Fasttext  | 0.648   | 0.721      | 0.889  | 0.668        |
| Ensemble  | 0.769   | 0.783      | 0.921  | 0.718        |

4.2.2. Machine Learning Approach

For language identification, we implemented two machine learning classification methods (logistic regression and multinomial naïve Bayes classifier), using those features that better characterize our data. Each term of the training/validation set was then “vectorized” using unigram or character trigram features, with Python standard feature extraction methods.

Figure 2 shows the comparison in the overlapping of the character trigram features for the different number of most frequent elements (400 and 1200), extracted from WIKI_NOOOV and WIKITXT, respectively. French and German are the languages where there is the least overlap, while Spanish, Italian and English share more character trigrams. This allows us to hypothesize that French and German give better results in terms of precision, while the other languages are more recall-oriented. Given the model of dataset construction, WIKITXT has a higher number of overlaps, since the same word can belong to different classes. In WIKI_NOOOV, words that belong to more than one language were removed.

Tests were done with a different maximum number of features extracted, to confirm the intuitive idea that more data lead to greater accuracy. Table 2 reports results from the logistic regression and naïve Bayes classifiers trained on datasets using character trigrams with n top results (we have not reported the results for individual words here because they are not valid enough to compare). In the table, train indicates the subset of WIKI_NOOOV or WIKITXT used to train and evaluate the data, while test was used for evaluation only.

It can be seen that the results obtained with the trained model on WIKITXT are worse than those obtained using cleaner data.

Tables 3 and 4 show the recall, precisions and F1 values for term classification in the different languages, on the different datasets on which they were trained and validated, and on PAPERS and PAPERS_1GRAM. The average is also shown, and the weighted average computed on the numerosity of the different classes. Data for the 1200 features are shown.

The values of the evaluation measures are consistent with the reported confusion matrices for the same values. The values for the WIKI_NOOOV dataset, which shows very few overlaps for the extracted trigrams, are very high in the evaluation measures. In Table 4 for the training data, using logistic regression, the recall values are very high compared to the precision values, for all languages except German.
Figure 2. Feature overlapping for WIKI_NOOOV and WIKITXT datasets for 400 and 1200 elements. 
(a) WIKI_NOOOV, 400 features, trigram; (b) WIKI_NOOOV, 1200 features, trigram (c) WIKITXT, 
400 features, trigram (d) WIKITXT, 1200 features, trigram.

Table 2. Results for the classifiers trained on the WIKI_NOOOV and WIKITXT datasets.

| Feature | Training Dataset | No. Features | Features Extracted | Data                | Accuracy (Logreg/Naïve Bayes) |
|---------|------------------|--------------|--------------------|---------------------|------------------------------|
| trigrams | WIKI_NOOOV      | 1200         | 2622               | Train               | 0.907/0.818                  |
|         |                  |              |                    | Test                | 0.901/0.816                  |
|         |                  |              |                    | PAPERS              | 0.923/0.912                  |
|         |                  |              |                    | PAPERS_1GRAM        | 0.607/0.539                  |
|         |                  | 800          | 1831               | Train               | 0.881/0.798                  |
|         |                  |              |                    | Test                | 0.877/0.796                  |
|         |                  |              |                    | PAPERS              | 0.904/0.904                  |
|         |                  |              |                    | PAPERS_1GRAM        | 0.582/0.518                  |
|         |                  | 400          | 1025               | Train               | 0.831/0.761                  |
|         |                  |              |                    | Test                | 0.829/0.759                  |
|         |                  |              |                    | PAPERS              | 0.877/0.882                  |
|         |                  |              |                    | PAPERS_1GRAM        | 0.527/0.476                  |
### Table 2. Cont.

| Feature        | Training Dataset | No. Features | Features Extracted | Data | Accuracy (Logreg/Naïve Bayes) |
|----------------|------------------|--------------|--------------------|------|--------------------------------|
| trigrams       | WIKITXT          | 1200         | 2721               | Train | 0.789/0.710                   |
|                |                  |              |                    | Test  | 0.764/0.708                   |
|                |                  | 800          | 1897               | PAPERS | 0.822/0.926                  |
|                |                  |              |                    | PAPERS | 0.395/0.523                  |
|                |                  | 400          | 1810               | PAPERS | 0.813/0.877                 |

### Table 3. Results for the classifiers trained on WIKI_NOOOV dataset, using character trigrams and 1200 max features.

| Eval. Measures | Data        | en | it | fr | es | de | Average | Weighted Avg |
|----------------|-------------|----|----|----|----|----|---------|--------------|
| Precision      | Train/logreg| 0.9638 | 0.7859 | 0.9000 | 0.8860 | 0.9097 | 0.8891 | 0.9087 |
| Recall         | Train/logreg| 0.9683 | 0.8678 | 0.8776 | 0.8929 | 0.8908 | 0.8995 | 0.9078 |
| F1-score       | Train/logreg| 0.9660 | 0.8248 | 0.8887 | 0.8894 | 0.9001 | 0.8938 | 0.9081 |
| Precision      | Test/logreg  | 0.9651 | 0.8459 | 0.8732 | 0.8837 | 0.8839 | 0.8904 | 0.9009 |
| Recall         | Test/logreg  | 0.9590 | 0.7702 | 0.8930 | 0.8752 | 0.9068 | 0.8808 | 0.9010 |
| F1-score       | Test/logreg  | 0.9621 | 0.8063 | 0.8830 | 0.8794 | 0.8952 | 0.8852 | 0.9008 |
| Precision      | Train/naive_bayes | 0.9145 | 0.6374 | 0.8048 | 0.7865 | 0.8059 | 0.7898 | 0.8205 |
| Recall         | Train/naive_bayes | 0.8956 | 0.7096 | 0.7941 | 0.8402 | 0.7687 | 0.8017 | 0.8182 |
| F1-score       | Train/naive_bayes | 0.9049 | 0.6716 | 0.7995 | 0.8125 | 0.7869 | 0.7951 | 0.8189 |
| Precision      | Test/naive_bayes | 0.8940 | 0.7025 | 0.7946 | 0.8393 | 0.7672 | 0.7995 | 0.8163 |
| Recall         | Test/naive_bayes | 0.9147 | 0.6309 | 0.8035 | 0.7806 | 0.8078 | 0.7875 | 0.8169 |
| F1-score       | Test/naive_bayes | 0.9042 | 0.6648 | 0.7991 | 0.8089 | 0.7870 | 0.7928 | 0.8161 |
| Precision      | PAPERS/logreg  | 0.9148 | 0.8708 | 0.9620 | 0.9593 | 0.9897 | 0.9393 | 0.9249 |
| Recall         | PAPERS/logreg  | 0.8010 | 0.9448 | 0.9779 | 0.9456 | 0.8972 | 0.9133 | 0.9230 |
| F1-score       | PAPERS/logreg  | 0.8541 | 0.9063 | 0.9699 | 0.9524 | 0.9412 | 0.9248 | 0.9227 |
| Precision      | PAPERS/naive_bayes | 0.8750 | 0.8708 | 0.9728 | 0.9390 | 0.9691 | 0.9253 | 0.9146 |
| Recall         | PAPERS/naive_bayes | 0.9006 | 0.9278 | 0.9227 | 0.9362 | 0.7833 | 0.8941 | 0.9120 |
| F1-score       | PAPERS/naive_bayes | 0.8876 | 0.8984 | 0.9471 | 0.9376 | 0.8664 | 0.9074 | 0.9120 |
| Precision      | PAPERS_1GRAM/logreg | 0.5841 | 0.4536 | 0.7442 | 0.6861 | 0.7345 | 0.6405 | 0.6500 |
| Recall         | PAPERS_1GRAM/logreg | 0.5683 | 0.7827 | 0.4971 | 0.6710 | 0.4472 | 0.5933 | 0.6076 |
| F1-score       | PAPERS_1GRAM/logreg | 0.5761 | 0.5743 | 0.5960 | 0.6785 | 0.5599 | 0.5962 | 0.6084 |
| Precision      | PAPERS_1GRAM/ naïve bayes | 0.7006 | 0.3772 | 0.5940 | 0.6010 | 0.7366 | 0.6019 | 0.6048 |
| Recall         | PAPERS_1GRAM/ naïve bayes | 0.3676 | 0.7706 | 0.5257 | 0.6997 | 0.3058 | 0.5339 | 0.5400 |
| F1-score       | PAPERS_1GRAM/ naïve bayes | 0.4822 | 0.5065 | 0.5578 | 0.6466 | 0.4322 | 0.5251 | 0.5324 |
Table 4. Results for the classifiers trained on WIKITXT dataset, using character trigrams, and 1200 max features.

| Eval. Measures | Data | eval | en | it | fr | es | de | Average | Weighted Avg |
|----------------|------|------|----|----|----|----|----|---------|---------------|
| Precision      | Train/logreg | 0.8058 | 0.7518 | 0.7201 | 0.7221 | 0.9443 | 0.7888 | 0.8222 |
|                | Test/logreg  | 0.9126 | 0.9198 | 0.9197 | 0.8934 | 0.5231 | 0.8337 | 0.8287 |
| Recall         | Train/logreg | 0.9411 | 0.9447 | 0.9431 | 0.9299 | 0.5364 | 0.8590 | 0.7898 |
|                | Test/logreg  | 0.7814 | 0.7242 | 0.6964 | 0.6924 | 0.9234 | 0.7636 | 0.7650 |
| F1-score       | Train/logreg | 0.8682 | 0.8373 | 0.8167 | 0.8129 | 0.6842 | 0.8039 | 0.7781 |
|                | Test/logreg  | 0.8419 | 0.8104 | 0.7926 | 0.7802 | 0.6678 | 0.7786 | 0.7759 |
| Precision      | Train/naïve bayes | 0.7524 | 0.6795 | 0.7268 | 0.5946 | 0.7867 | 0.7080 | 0.7250 |
| Recall         | Train/naïve bayes | 0.8285 | 0.8090 | 0.7004 | 0.8629 | 0.5388 | 0.7479 | 0.7105 |
| F1-score       | Train/naïve bayes | 0.7886 | 0.7386 | 0.7133 | 0.7041 | 0.6396 | 0.7169 | 0.7061 |
| Precision      | Test/naïve bayes | 0.8212 | 0.8154 | 0.6969 | 0.8501 | 0.5433 | 0.7454 | 0.7381 |
| Recall         | Test/naïve bayes | 0.7511 | 0.6786 | 0.7232 | 0.5914 | 0.7879 | 0.7064 | 0.7088 |
| F1-score       | Test/naïve bayes | 0.7846 | 0.7407 | 0.7098 | 0.6975 | 0.6431 | 0.7152 | 0.7128 |
| Precision      | PAPER/logreg    | 0.7159 | 0.7585 | 0.9511 | 0.8547 | 0.9691 | 0.8498 | 0.8334 |
| Recall         | PAPER/logreg    | 0.7975 | 0.8443 | 0.8102 | 0.8497 | 0.7287 | 0.8061 | 0.8225 |
| F1-score       | PAPER/logreg    | 0.7545 | 0.7991 | 0.8750 | 0.8522 | 0.8319 | 0.8225 | 0.8242 |
| Precision      | PAPERS/naïve bayes | 0.8920 | 0.8898 | 0.9783 | 0.9535 | 0.9794 | 0.9386 | 0.9291 |
| Recall         | PAPERS/naïve bayes | 0.9023 | 0.9545 | 0.9045 | 0.9292 | 0.8879 | 0.9157 | 0.9269 |
| F1-score       | PAPERS/naïve bayes | 0.8971 | 0.9211 | 0.9399 | 0.9412 | 0.9314 | 0.9261 | 0.9272 |
| Precision      | PAPERS_1GRAM/logreg | 0.4640 | 0.2470 | 0.5165 | 0.4100 | 0.6531 | 0.4581 | 0.4848 |
| Recall         | PAPERS_1GRAM/logreg | 0.3268 | 0.5416 | 0.4468 | 0.5917 | 0.1791 | 0.4172 | 0.3956 |
| F1-score       | PAPERS_1GRAM/logreg | 0.3835 | 0.3392 | 0.4791 | 0.4844 | 0.2811 | 0.3935 | 0.3858 |
| Precision      | PAPERS_1GRAM/naïve bayes | 0.6359 | 0.3850 | 0.6269 | 0.5347 | 0.7302 | 0.5825 | 0.5825 |
| Recall         | PAPERS_1GRAM/naïve bayes | 0.4289 | 0.7065 | 0.4579 | 0.7181 | 0.2887 | 0.5200 | 0.5231 |
| F1-score       | PAPERS_1GRAM/naïve bayes | 0.5123 | 0.4984 | 0.5292 | 0.6130 | 0.4138 | 0.5134 | 0.5150 |

4.2.3. Deep Learning Approach

There are several different deep learning approaches to language identification, such as reported in [1]. Here, we used a simple RNN, using the same two features used in the ML approach: character n-grams and single words. The neural network model was constituted by three layers fully connected that classified each input word as belonging to a language class. The first layers were composed of 500 or 250 nodes, with ReLu activation function, while the last one, which associated each word to a class (i.e., a language), was composed of five nodes with a SoftMax activation function. Table 5 reports the results of the training/test set on different datasets. The training phase was conducted with 10 epochs using an Adam optimizer and loss categorical cross-entropy function. The choice of the architecture of the ANN was made both based on the literature [1,7], and on the basis of other experiments carried out [26].

The first step was to transform the data into vectors, according to the features selected. Then, a feature matrix was created, based on the occurrences of each character trigram/word in the dataset. In the training phase, for all the cases tested there was no or little overlap on the features in the various languages (see Figure 2). This resulted in the model being able to discriminate between languages very well.

Table 5 reports the results, in terms of loss/accuracy, obtained applying the model comparing the two features, and a different number of features. The greater the number of features extracted, the better the results obtained. As n increases, the number of features extracted from the data increases, resulting in greater accuracy in the results. Results with 10 epochs are shown here. 50 epochs were tested, but the accuracy of the results...
did not change, while the computation time increased significantly. The results obtained using trigrams are better than those obtained on words. Results on unigram terms are not reported here.

Table 5. Accuracy values for ANN methods.

| Feature   | No. Features | Epochs | Data                | Loss/Accuracy          |
|-----------|--------------|--------|--------------------|------------------------|
| trigrams  | 1200         | 10     | Train              | Loss: 0.011; accuracy: 0.995 |
|           |              |        | Test               | Loss: 0.131; accuracy: 0.977 |
|           |              |        | PAPERS             | Accuracy: 0.889        |
|           |              |        | PAPERS_1GRAM       | Accuracy: 0.638        |
| WIKI_NOOOV|              |        | Train              | Loss: 0.026; accuracy: 0.988 |
|           |              |        | Test               | Loss: 0.153; accuracy: 0.967 |
|           | 800          | 10     | PAPERS             | Accuracy: 0.892        |
|           |              |        | PAPERS_1GRAM       | Accuracy: 0.623        |
|           |              |        | Train              | Loss: 0.088; accuracy: 0.961 |
|           |              |        | Test               | Loss: 0.23; accuracy: 0.938 |
|           | 400          | 10     | PAPERS             | Accuracy: 0.871        |
|           |              |        | PAPERS_1GRAM       | Accuracy: 0.567        |

4.2.4. Discussion

Tables 1–5 report the results in terms of accuracy of both the off-the-shelf methods and the models implemented and trained from scratch. In bold are the best results for each model/dataset.

Here are some considerations:

- Results on the dataset where there were overlaps (WIKITXT) were worse than those with the cleanest data (WIKI_NOOOV) with all methods.
- The PAPERS_1GRAM dataset consists of single words: it achieved very low precision with all methods. On the contrary, the PAPERS dataset, made up of strings with more words, obtained better results. In all tests, the methods generalized quite well on the PAPERS dataset, while on PAPERS_1GRAM they did not perform as well. This is the only case where the off-the-shelf packages achieved results comparable with the ML/ANN methods.
- More data lead to better results: this was valid for the size of the training set, for the length of the string to be evaluated and for the number of features to be extracted.
- In cases of nonbinary classification, such as this one, if the training data for a class are less numerous than the others, the results for that class are worse in terms of precision, while the effects on the other classes are worse in terms of recall. Generally, the general accuracy worsening was not very pronounced.

Models based on an ANN, with the different hidden levels and the many parameters used in training, are able to outperform other methods in the task of language identification: ANN methods in the same tests achieved better results than ML and off-the-shelf methods.
4.3. Task 2: Misspelling Identification and Correction

Due to the specific scope of the experiment, identifying misspelled words was a particularly difficult task due to: (i) different languages; (ii) a lack of context; (iii) very specialized scientific terms; (iv) the presence of proper nouns/project names.

4.3.1. Statistical Approach

At the stage of identifying spelling errors, several strategies were identified to be integrated, since individually none proved sufficient in the specific case. All these strategies indicated clues about the correctness of the tags:

- Presence of the tag in the word embedding model adopted. The word embedding models, both the pretrained ones and the ones trained on the ArXiv repository, produced for each word, single or compound, a vector that allowed us to compare it with those of other words to obtain semantic similarities, usage contexts, etc. An indicator of whether the term was spelled correctly/had spelling errors was whether the term was identified in the model or classified oov (out of vocabulary) in the word embedding models. To minimize the probability that a correct but infrequent term, because it belonged to a specific context or was newly introduced, was classified as oov, we used both pretrained (W2V_WIKI) and specifically trained word embedding (W2V_ARXIV). The last model was only for the English language.

- Presence of “as-is” tags (both compound and as single components) in the ARXIV/WIKIPEDIA datasets. The presence of the term in the dataset offered us the possibility of identifying the presence/absence of possible misspellings. It also helped us compute the probability that that word appears in the chosen/identified language, thus providing indications that were proper to the language model. This information was then used to calculate the probability of a specific correction, in the ascertained presence of an error.

- Presence of “as-is” tags in the first results of Google.

Once the presence of misspellings was detected with a certain degree of confidence, the second step of the approach was to apply statistical and/or machine learning methods to assess the probability of replacing one term with another. It also used the first results of a Google search or the “do you mean” function to propose a correction: it proposed several tools for automatic correction as follows:

- Automatic correction of the error, due to, e.g., the inversion of two characters, to the substitution of two close characters on the keyboard;
- Google search first n results, with n = 1;
- Split terms proved to be pasted (lack of space).

A voting system (ensemble) was defined, which associated to each term (suggestion of correction) a degree of confidence; the ensemble was evaluated based on a voting mechanism determined by the overlapping/equality of the proposed solution and eventually by the probability of the correction. Table 6 reports the results for each step of the automatic correction tools and the final ensemble (in grey) accuracy values. Due to the method used to introduce errors, the EDIT1 misspelling algorithm yielded better results than those NEARBY. We report only the results of the EDIT1 algorithm, better than those of NEARBY, which are more comparable with those (even better) obtained by the artificial neural network.

4.3.2. Deep Learning Approach

The basic idea is to use an artificial neural network designed to transform an input into an output, adapted to natural language, thus capable of remembering the various states of the input. Recurrent neural networks, such as the long short-term memory, or LSTM, network, are specifically designed to support sequences of input data. In Figure 3a, the LSTM cell is represented.
Table 6. Accuracy values for the statistical approach to misspelling correction.

| Lang                | Italian | English | French | German | Spanish |
|---------------------|---------|---------|--------|--------|---------|
| Presence as is      | 0.196   | 0.206   | 0.154  | 0.170  | 0.164   |
| Automated correction| 0.763   | 0.781   | 0.572  | 0.638  | 0.597   |
| Google first result | 0.251   | 0.309   | 0.245  | 0.170  | 0.186   |
| Split pasted terms  | 0.269   | 0.270   | 0.220  | 0.228  | 0.238   |
| Ensemble            | 0.794   | 0.823   | 0.672  | 0.672  | 0.644   |

The network chosen was sequence2sequence, widely used for machine translation purposes. This network, also called NN encoder–decoder, consists of two recurrent neural networks (RNN) that act as an encoder and a decoder pair. This architecture allows the model both to support variable-length input sequences and to predict or output variable-length output sequences. An encoder LSTM model reads the input sequence step-by-step. After reading in the entire input sequence, the hidden state or output of this model represents an internal learned representation of the entire input sequence as a fixed-length vector. This vector is then provided as an input to the decoder model that interprets it as each step in the output sequence is generated.

The two networks were trained jointly to maximize the conditional probability of the target sequence given a source sequence. Additionally, hidden units were inserted to improve both the memory capacity and the ease of training. The model was based on the character unigram feature. The number of input/output tokens varies from a minimum of 33/29 to a maximum of 61/53, respectively. The different number of characters in the input/output depends both on the language and on the error-introduction algorithm chosen. The lower bound was that of the English dataset, which has no accented letters, while the upper bound was that of the mixed dataset. Figure 3b represents the sequence2sequence neural network implemented using two LSTM layers.

4.3.3. Discussion

Table 7 shows the loss and accuracy of the network on the augmented data broken down by the languages of the experiment and a dataset that takes and mixes data in the different languages. The model was trained on two different datasets, GENERAL and PAPERS: the table reports both the accuracy/loss obtained on the datasets used in the training, split in train/test and on the LIBRARY dataset, for which these experiments were carried out. The model generalized quite well for Italian and mixed data, maintaining high accuracy, but losing a few points when using the ANN trained on the other dataset. This is because the LIBRARY dataset is mostly composed of Italian tags.
Table 7. Accuracy values for the statistical approach to misspelling correction.

| Language | Training Dataset | Data | Loss/Accuracy/NEARBY Misspelling | Loss/Accuracy EDIT1 Misspelling |
|----------|------------------|------|---------------------------------|---------------------------------|
|          |                  |      | Loss | Accuracy | NEARBY | Loss | Accuracy |
| Italian  | GENERAL          | Train| L: 0.008; A: 0.997 | L: 0.005; A: 0.997 |
|          |                  | Test | L: 0.018; A: 0.994 | L: 0.007; A: 0.996 |
|          | LIBRARY          |      | L: 0.045; A: 0.923 | L: 0.47; A: 0.915 |
|          | PAPERS           | Train| L: 0.005; A: 0.9976 | L: 0.0006; A: 0.999 |
|          |                  | Test | L: 0.006; A: 0.9974 | L: 0.003; A: 0.9994 |
|          | LIBRARY          |      | L: 0.165; A: 0.794 | L: 0.155; A: 0.788 |
| French   | GENERAL          | Train| L: 0.008; A: 0.996 | L: 0.005; A: 0.998 |
|          |                  | Test | L: 0.019; A: 0.994 | L: 0.02; A: 0.997 |
|          | LIBRARY          |      | L: 1.288; A: 0.823 | L: 1.21; A: 0.839 |
|          | PAPERS           | Train| L: 0.003; A: 0.998 | L: 0.006; A: 0.997 |
|          |                  | Test | L: 0.011; A: 0.996 | L: 0.007; A: 0.996 |
|          | LIBRARY          |      | L: 1.50; A: 0.805 | L: 1.515; A: 0.809 |
| Spanish  | GENERAL          | Train| L: 0.009; A: 0.997 | L: 0.004; A: 0.998 |
|          |                  | Test | L: 0.018; A: 0.995 | L: 0.035; A: 0.992 |
|          | LIBRARY          |      | L: 0.914; A: 0.855 | L: 1.089; A: 0.850 |
|          | PAPERS           | Train| L: 0.004; A: 0.998 | L: 0.005; A: 0.998 |
|          |                  | Test | L: 0.006; A: 0.997 | L: 0.02; A: 0.998 |
|          | LIBRARY          |      | L: 1.79; A: 0.778 | L: 1.703; A: 0.791 |
| German   | GENERAL          | Train| L: 0.006; A: 0.998 | L: 0.005; A: 0.997 |
|          |                  | Test | L: 0.013; A: 0.996 | L: 0.005; A: 0.997 |
|          | LIBRARY          |      | L: 1.041; A: 0.847 | L: 1.172; A: 0.844 |
|          | PAPERS           | Train| L: 0.0012; A: 0.9999 | L: 0.002; A: 0.9991 |
|          |                  | Test | L: 0.0013; A: 0.9993 | L: 0.002; A: 0.9991 |
|          | LIBRARY          |      | L: 2.76; A: 0.706 | L: 3.138; A: 0.693 |
| English  | GENERAL          | Train| L: 0.008; A: 0.997 | L: 0.006; A: 0.998 |
|          |                  | Test | L: 0.021; A: 0.994 | L: 0.0134; A: 0.996 |
|          | LIBRARY          |      | L: 1.079; A: 0.830 | L: 1.109; A: 0.829 |
|          | PAPERS           | Train| L: 0.004; A: 0.998 | L: 0.004; A: 0.998 |
|          |                  | Test | L: 0.06; A: 0.997 | L: 0.005; A: 0.997 |
|          | LIBRARY          |      | L: 2.44; A: 0.715 | L: 2.70; A: 0.698 |
| Mixed languages | GENERAL | Train| L: 0.013; A: 0.995 | L: 0.01; A: 0.996 |
|          |                  | Test | L: 0.050; A: 0.987 | L: 0.053; A: 0.988 |
|          | LIBRARY          |      | L: 0.183; A: 0.961 | L: 0.27; A: 0.948 |
|          | PAPERS           | Train| L: 0.009; A: 0.995 | L: 0.007; A: 0.996 |
|          |                  | Test | L: 0.027; A: 0.992 | L: 0.009; A: 0.996 |
|          | LIBRARY          |      | L: 0.454; A: 0.921 | L: 0.566; A: 0.904 |

By analyzing the results of both misspelling algorithms, we can see that the datasets obtained by applying EDIT1 obtain almost the same results, compared to the NEARBY datasets. Comparing the results, we can see that the artificial neural networks outperform the ML voting mechanism on all tests.

4.4. Task3: Syntactic Similarity

The syntactic similarity between the two terms implies that these terms share many characters in common. When two strings have a very high similarity, it means that they are virtually identical; it can be the same singular/plural tag or the presence of one more letter or one less or a substituted letter. In this case, strings with a similarity greater than 0.84 were considered related and stored in a syntactic similarity matrix. To calculate syntactic similarity, we used the Jaro–Winkler algorithm (using the Jellyfish Python package), creating a correlation matrix with a minimum similarity value. For example, a French web user interested in “algèbre” should also be suggested “algebrique”, and “algebriques”, all with similarity values greater than 0.86. The algorithm also proposes syntactically but non semantically related terms such as “alger” and “algorithm”. Figure 4 shows the syntactic similarity graph for “algèbre”.

The semantic similarity was used here to overcome the problem of synonyms: for example, catalogers/researchers can use the same concept using different terms, or data from different sources can address the same topic from different perspectives. Moreover, the suggestion of tags to web users allows them to continue their exploration according to content-based browsing.

The first step was the computation of the similarity matrix to the terms (and their vectors) of the tags. In a second phase, the cosine similarity as a criterion of similarity was applied to the similarity matrices and/or directly to word2vec models. The similarity matrix was based on the vectors from the word2vec model. Currently, each language is processed individually. The example uses the word2vec pretrained model for French.

For cosine similarity, similarity values of 0.5, 0.6, 0.7, 0.75, 0.8, 0.85 and 0.90 were tested. In the end, 0.75 was chosen as a reasonable trade-off between a too low number of similar tags and the retrieval of elements that were not too closely related. Figure 5 shows the tags most similar to “algèbre” using the pretrained model for French.
5. Discussion

Language identification works well on long strings of standard terms, i.e., frequent terms in that language. The problems encountered in the experimentation were due both to the presence of extremely specialized terms related to innovative research activities and the use of single or short words. In addition, the training on the set of clean terms, i.e., without overlapping, allowed us to obtain better results. ANN methods outperformed the others, and it can be said that off-the-shelf packages obtained results comparable to machine learning methods.

Misspelling correction is a very useful tool because it improves not only the quality of the data but also the overall evaluation of the system. In our case, we were interested in the ability to correct highly specialized data. Typically, word correction tools are language-dependent because words simply written in another language could be considered errors. We experimented with both monolingual and mixed data. In general, we can say that ANN performed much better than statistical methods. ANN results were always very good on the data they were trained on. Models trained on the GENERAL dataset generalized better. While there was no significant difference for ANN models across languages, statistical methods performed best in English.

Regarding syntactic similarity, the creation of the similarity matrix using the Jaro–Winkler algorithm allowed us to overcome singular/plural problems as well as different spellings of the same tag, possibly also in different languages. For example, in British English color is spelled “colour”, in American English it is spelled “color”, in Italian “colore”, while in French it is “couleur”. Moreover, in French, Spanish, German, Italian, etc., words are spelled with accents: you do not have to worry about how you spell them anymore, because the system will be able to retrieve the different spellings. On the other hand, it can be difficult to set the degree of similarity to find similar tags without introducing terms that are of no help in the search.

Semantic similarity is something we intuitively need to group similar terms within the scope of the specific application. Semantic similarity also allows us to overcome the problems of different spellings and the presence/absence of accents, with the condition that the words are present in the vocabulary of the word embedding model used, i.e., they are not oov. Thus, it could be the case for extremely specific oov terms or not being able to identify similar terms. Unlike syntactic similarity, which can be used on tags in different languages, semantic similarity requires a vectorization of the terms, which is language-dependent.

The experimental setup was implemented in Python 3.7, using standard packages like NumPy, Matplotlib, Pandas and other more specific ones for the processing of textual data such as NLTK, Treetagger, Gensim, Scikit-learn, Pytorch, Tensorflow, Keras, together with some experimental packages in GitHub.

6. Conclusions and Future Work

In this paper, we presented some tools useful to handle the problems of textual annotations in digital libraries, or more generally in archives that rely heavily on the use of keywords in searching and browsing. These tools were designed both for catalogers/researchers who find themselves managing data from a variety of sources and for those who require data cleaning based on the language in which it is written. These tools will be integrated into web catalogs to enhance traditional ways of searching and browsing data on the web and offer a multimodal search and visualization tools. Language detection and typos detection and correction were addressed with both machine learning and deep learning techniques. To train and validate the models, several datasets were used/created/scraped, both in a single language (Italian/English/French/German and Spanish) and in mixed languages. The mixed datasets were the basis for the language detection models. They were also used in Task 2, to train the deep learning model. In addition, datasets were artificially constructed for automatic error correction based on two
different error models. Syntactic and semantic similarity rely on edit distance among tags, the use of word embedding methods and similarity measures.

We plan to include these tools both at the cataloging stage, to help catalogers detect errors early, and to facilitate the use of legacy library archives in the presence of dirty data. Researchers, students and scholars often use keywords to search for items of interest. The presence of errors could penalize them greatly.

This work is still in progress. Preliminary results have been discussed with catalogers to evaluate them, both qualitatively and quantitatively. They have been very positively received. For the tags organization, analyzing the results, according to our experience and sensibility, we can say that actually proposing a search engine based on clusters can be of great value for a web user, integrating it with other more standard methods.

Some problems have already emerged in both language identification and automatic error detection because, especially when tags refer to very specific instances such as ongoing project titles, it is impossible to automatically determine whether the tag is correct or not.

Another problem, intrinsic to the data we work on, is to deal with short strings that have no context: in fact, the models we have presented do not generalize well on the PAPERS 1GRAM dataset. We intend to address the problem in a combined way: on the one hand using other data, if any, to help create a context, or integrating/annotating them from the web, and on the other hand, improving the training/testing datasets.

Another issue concerns semantic similarity between different language tags. We need to further investigate how to train models whose terms are aligned, how to define datasets for this purpose, and how to fully exploit the results in the search phase.

Moreover, we intend to work on:

- Extending the managed languages to additional Western languages and considering the inclusion of language extensions to languages that use different alphabets, such as Chinese (simplified), Japanese, Russian, etc.
- Integrating syntactic and similarity relatedness in a unique similarity measure;
- Considering not only syntactic errors, but also semantic ones. This will require considering the context in which the keywords are used. If available, the title and an abstract will also be used.

In addition, we intend to experiment with an evolution of word embedding models, using Bert [18], Elmo, etc., which are able to identify and manage homonyms.

Author Contributions: Methodology, I.G.; software, I.G. and M.T.A.; data curation, I.G. and M.T.A.; writing—original draft preparation, I.G.; writing—review and editing, M.T.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Data used in this experiment are extracted from—(1) WIKIPEDIA dataset: title and abstract of Wikipedia articles, as stored in the dumps of 20210801 https://dumps.wikimedia.org/backup-index.html accessed on 30 January 2022, in the different languages considered; (2) ARXIV: https://www.kaggle.com/Cornell-University/arxiv accessed on 30 January 2022 and (3) papers scraped, via crossref API, from Scopus (Elsevier’s abstract and citation database) in the 5 languages considered.

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

References
1. Jauhiainen, T.L. Automatic language identification in texts: A survey. J. Artif. Intell. Res. 2019, 65, 675–782. [CrossRef]
2. Yazdani, A.; Ghazisaeedi, M.; Ahmadinejad, N.; Giti, M.; Amjadi, H.; Nahvijou, A. Automated misspelling detection and correction in persian clinical text. J. Digit. Imaging 2020, 33, 555–562. [CrossRef] [PubMed]
3. Somboonsak, P. Misspelling error detection in Thai language application. In Proceedings of the 6th International Conference on Information Technology: IoT and Smart City, New York, NY, USA, 31 December 2018; pp. 19–24.
4. Hu, Y.; Jing, X.; Ko, Y.; Rayz, J.T. Misspelling Correction with Pre-trained Contextual Language Model. In Proceedings of the 2020 IEEE 19th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC), Beijing, China, 26–28 September 2020; pp. 144–149.

5. Ramakrishnan, M.; Zadrozy, W.; Tabari, N. UVA Wahoos at SemEval-2019 Task 6: Hate Speech Identification using Ensemble Machine Learning. In Proceedings of the 13th International Workshop on Semantic Evaluation; Association for Computational Linguistics, Nanchang, China, 27–29 September 2019; pp. 806–811.

6. Goldberg, Y. A Primer on Neural Network Models for Natural Language. *J. Artif. Intell. Res.* 2016, 57, 345–420. [CrossRef]

7. Simões, A.A. Language Identification: A Neural Network Approach. In Proceedings of the 3rd Symposium on Languages, Applications and Technologies, Bragança, Portugal, 19–20 June 2014.

8. Botha, G.R. Factors that affect the accuracy of text-based language identification. *Comput. Speech Lang.* 2012, 26, 307–320. [CrossRef]

9. López-Moreno, I.G.-D.-R. Automatic language identification using deep neural networks. In Proceedings of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, Italy, 4–9 May 2014; pp. 5337–5341.

10. Hládek, D.J. Survey of Automatic Spelling Correction. *Electronics* 2020, 9, 1670. [CrossRef]

11. Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv* 2014, arXiv:1406.1078.

12. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. *arXiv* 2014, arXiv:1409.3215.

13. Etoori, P.M. Automatic spelling correction for resource-scarce languages using deep learning. In Proceedings of the ACL 2018, Student Research Workshop, Melbourne, Australia, 15–20 July 2018.

14. Büyüktürk, O. Context-Dependent Sequence-to-Sequence Turkish Spelling Correction. *ACM Trans. Asian Low-Resour. Lang. Inf. Processing (TALLIP)* 2020, 19, 1–16. [CrossRef]

15. Gali, N.; Mariescu-Istodor, R.; Hostettler, D.; Fränti, P. Framework for syntactic string similarity measures. *Expert Syst. Appl.* 2019, 129, 169–185. [CrossRef]

16. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J.; Sutskever, L.; Zweig, G. Tool for Computing Continuous Distributed Representations of Words: Word2vec. Available online: https://code.google.com/p/word2vec (accessed on 30 January 2022).

17. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. *Adv. Neural Inf. Processing (TALLIP)* 2013, 26, 3111–3119.

18. Devlin, J.M.-W. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* 2018, arXiv:1810.04805.

19. Wang, C.; Peng, X.; Liu, M.; Xing, Z.; Bai, X.; Xie, B.; Wang, T. A learning-based approach for automatic construction of domain glossary from source code and documentation. In Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Athens, Greece, 12 August 2019; pp. 97–108.

20. Comito, C.; Forestiero, A.; Pizzuti, C. Word Embedding based Clustering to Detect Topics in Social Media. In Proceedings of the 2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI), Thessaloniki, Greece, 14–17 October 2019.

21. Hu, J.; Li, S.; Yao, Y.; Yang, G.; Hu, J. Patent keyword extraction algorithm based on distributed representation for patent classification. *Entropy* 2018, 20, 104. [CrossRef] [PubMed]

22. Fellbaum, C. WordNet. In *Theory and Applications of Ontology: Computer Applications*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 231–243.

23. Mohammed, N. Extracting word synonyms from text using neural approaches. *Int. Arab J. Inf. Technol.* 2020, 17, 45–51. [CrossRef]

24. Cornell_University. ArXiv Dataset. 2020. Available online: https://www.kaggle.com/Cornell-University/arxiv (accessed on 30 January 2022).

25. Mukherjee, H.D. An Ensemble Learning-Based Language Identification System. In *Computational Advancement in Communication Circuits and Systems*; Springer: Singapore; 2020; pp. 129–138.

26. Artese, M.T.; Gagliardi, I. *Machine Learning and Neural Networks Tools to Address Noisy Data Issues*; Dipp: Burgas, Bulgaria, 2021.