sciendo

Albanian Text Classification: Bag of Words Model and Word Analogies

Arbana Kadriu
SEE University, Tetovo, Macedonia
Lejla Abazi
SEE University, Tetovo, Macedonia
Hyrije Abazi
SEE University, Tetovo, Macedonia

Abstract

Background: Text classification is a very important task in information retrieval. Its objective is to classify new text documents in a set of predefined classes, using different supervised algorithms. **Objectives:** We focus on the text classification for Albanian news articles using two approaches. **Methods/Approach:** In the first approach, the words in a collection are considered as independent components, allocating to each of them a conforming vector in the vector's space. Here we utilized nine classifiers from the scikit-learn package, training the classifiers with part of news articles (80%) and testing the accuracy with the remaining part of these articles. In the second approach, the text classification treats words based on their semantic and syntactic word similarities, supposing a word is formed by n-grams of characters. In this case, we have used the fastText, a hierarchical classifier, that considers local word order, as well as sub-word information. We have measured the accuracy for each classifier separately. We have also analyzed the training and testing time. **Results:** Our results show that the bag of words model does better than fastText when testing the classification process for not a large dataset of text. FastText shows better performance when classifying multi-label text. **Conclusions:** News articles can serve to create a benchmark for testing classification algorithms of Albanian texts. The best results are achieved with a bag of words model, with an accuracy of 94%.

Keywords: data mining, text classification, news articles, machine learning
JEL classification: C00, C30
Paper type: Research article

Received: Dec 01, 2017
Accepted: Feb 22, 2018

Citation: Kadriu, A., Abazi, L., Abazi, H., (2019), "Albanian Text Classification: Bag of Words Model and Word Analogies," Business Systems Research, Vol. 10 No. 1, pp. 74-87.
DOI: 10.2478/bsrj-2019-0006

Introduction

With regard to text classification, the issue is how to build classification functions ("classifiers") if we have a representation of a document d and a fixed set of classes C = {c1, c2, …, cn}. That is how to define the class of d: γ(d) ∈ C, where γ(d) is a

classification function (Manning et al., 2008). The classification problem belongs to the supervised learning approach, where input data must be a hand-classified data aiming to train a classifier. This way of creating hand-labeling training data is very time consuming (Jurka et al., 2013). There are considerable pre-trained classified data for languages for which there is industry interest in text mining. From over 6000 languages from all corners of the world, only a small number have produced resources needed as a basis for future applications (Scannell, 2007).

News articles in news portals usually are pre-categorized in some common sets of categories such as latest news, economy news, world news, sports news, etc. This job is done manually on a daily basis, aiming a better interface with the reader. Categorized news articles are a good source for getting the classified text in case of under-resourced languages, for which there is little or no commercial interest. These categorized corpora can be used to train machine learning algorithms that will learn how to classify a new, unknown text, according to the given trained data.

Different approaches have been used in news articles classification: text pattern mining (Chaudhari et al., 2013), which is considered to be an approach better than the term-based and phrase-based approach; random forests and textual and visual multimodal features (Liparas et al., 2014); named entities, enhancing so the performing of hierarchical text classification for news articles (Gui et al., 2012); Bayesian text classification approaches like Naïve Bayes, Complementary Naïve Bayes, use of {1,2,3}-grams, and use of oversampling (Swezey et al., 2012); Dependency- Latent Dirichlet Allocation model which tested on large-scale datasets generally outperforms binary Support Vector Machines (Rubin et al., 2012); three alternatives of semi-supervised learning propagation algorithms (Absorbing Random Walk, Random Walk with Restart, Local Consistency Global Consistency), aspiring to propagate political leaning of known articles and users to the target nodes (Zhou et al., 2011); Scalable Classification Algorithm for personalized news classification (Antonellis et al., 2006).

In order to apply supervised learning algorithms to process textual documents, the content of these documents should be converted into numerical features. Natural Language Processing (NLP) applications usually take words as basic input units; vectors of real-valued numbers, which represent words in an n-dimensional space (known as word embedding), have been found to efficiently provide such representations (Hartmann et al., 2017). Many existing text processing schemes and algorithms handle words as atomic units, with no conception of similarity among words, as they are denoted as keys in a vocabulary, different from treating word vectors on semantic and syntactic word analogies (Mikolov et al., 2013). In the bag of words model, only the evidence on the number of occurrences of each term is preserved. (Manning et al., 2008). In this case, every term is denoted as a unique vector in the terms vector space, not taking into account the internal term structure. In contrast to this approach is the word analogy approach, where each term is denoted as a bag of character n-grams (Bojanowski et al., 2017).

A comparison of these two approaches is made on Portuguese POS tagging and sentence semantic similarity tasks; the obtained results suggest that word analogies are not appropriate for word embedding evaluation, instead task-specific evaluations maybe a better option (Hartmann et al., 2017).

The Albanian language is an Indo-European language, but it has no noticeable similarity to any other Indo-European language, being a language group of its own. In the case of rich-resource languages, there are a lot of corpora with categorized text, which can be used for training and further NLP applications. However, there are low-resourced languages that are not driven by the market and industry, and not

much is done about them. In such situations, it is essential to find alternative ways to obtain categorized texts in this class of languages that can be trained and tested with existing tools produced for this purpose.

Methodology

For the purpose of this research, we have extracted the news from five different Albanian news portals, taking the articles, together with their titles, links, sources, summaries, and categories. This news belongs to the one-month period, with news portals originating from Albania and Kosovo. Part of the news is extracted using RSS feeds provided from the sources. In the case of sources that do not offer such services, we use regular expressions to extract news and their related fields. We excerpt wanted text and eliminate needless HTML tags. HTML texts commonly contain jumble around the body of an item, what is confusion concerning the actual content. Firstly, we want to reduce the "muted" content and find the subject matter areas. After parsing a web page, noise points are eliminated, and in this way, the dominant subject article is extracted. Parts like scripts, styles, container parts (table, div), etc. are also removed. We have also identified spam string list for portal separately, and all the paragraphs that contain some word from this spam list are removed. All those patterns are manually detected and corrected. This way, we have created separate (cleaned) text for categories: latest, economy, sport, showbiz, technology, culture, world. Table 1 gives an illustration of the numbers of news articles for each category, together with its size in kilobytes. Each text is then tokenized in sentences. From the overall gained corpus, stop-words list for Albanian is created. This list then is used to remove all occurrences of the stop-words, aiming for better efficiency. In the end, we have created a list of tuples, where each tuple consists of a sentence and the category where it belongs. This list of tuples served as input for comparison of different classification algorithms when training and testing obtained news articles.

Table 1
Number of categories and overall size for each category

Category	# of news articles	Size in KB
latest	998	1868
economy	60	91
sport	98	181
showbiz	62	115
technology	48	81
culture	69	128
world	95	175

Source: Authors' work

A cutoff of 80% is used to separate the training and testing set. First, we do the feature extraction of the training and testing set, which will be used later as input for classification algorithms, treating words as atomic units. We allocate to all terms a weighting according to the tf-idf scheme described in (Manning et al., 2008): "In a document a weight that depends on the number of incidences of the term in the document and in the collection. The simplest methodology is to assign the weight to be equal to the number of occurrences of term t in document d. This weighting scheme is referred to as term frequency and is denoted $tf_{t,d}$, the frequency with the subscripts denoting the term and the document in order. The document frequency d_{ft} is defined to be the number of documents in the collection that contain a term t, and the inverse document frequency (idf) of a term t is defined as follows:

$$idf_t = \log(N/d_{ft}) \quad (1)$$

We now combine the definitions of term frequency and inverse document frequency to produce a composite weight for each term in each document." This scheme gives to term t an in document d weight calculated by this formula:

$$tf\_idf_{t,d} = df_{t,d} x\ idf_t \quad (2)$$

The algorithms that we have used for this comparison involve the following classifiers: *Multinomial, LinearSVC, Neighbour, Bernoulli, Centroid, SGD, Perceptron, Ridge, PassiveAggressive*. We have used for this purpose the package *scikit-learn*, an efficient tool for data mining and data analysis (Pedregosa et al, 2011), using the Python language, which is also used for news crawling and data preparation process:

```
results = []
results.append(benchmark(MultinomialNB(),"Multinomial"))
results.append(benchmark(svm.LinearSVC(),"LinearSVC"))
results.append(benchmark(KNeighborsClassifier(),"Neighbour"))
results.append(benchmark(BernoulliNB(alpha=.005),"Bernoulli"))
results.append(benchmark(NearestCentroid(),"Centroid"))
results.append(benchmark(SGDClassifier(alpha=.0001,
n_iter=50,penalty="elasticnet"),"SGD"))
results.append(benchmark(Perceptron(n_iter=50),"Perceptron"))
results.append(benchmark(RidgeClassifier(tol=1e-2,
solver="lsqr"),"Ridge"))
results.append(benchmark(PassiveAggressiveClassifier(n_iter=50),"P
assiveAggressive"))
```

To evaluate these algorithms, the benchmark function returns the description of the classifier, the accuracy score (how accurate is the classification of a model), the training time (time to construct the model) and testing time (time to test the model).

The obtained output from this experiment is then compared with the performance on the same data gained when classifying with *fastText*, a library created by the Facebook Research Team for efficient learning of sentence classification (Joulin et al., 2016). For this approach, the format of the training/testing test is changed in that way that every sentence is in a new line, starting with _ _ label _ _ classNam. FastText is a method that considers word analogies when creating word embeddings, representing words as sum of the n-gram character vectors (Bojanowski et al., 2017).

Classification algorithms

Since creating a hand-labeled classified data has a high cost in terms of human resources and duration, different supervised learning algorithms are developed with the goal to automatically assign a label to a new document, having initially trained the algorithm. There are many such algorithms, broadly classified in linear classifiers, probabilistic classifiers, and vector space classifiers.

According to (Manning et al., 2008): "*Naïve Bayes* is a probabilistic classifier, where the probability of a document *d* being in class c is computed as follows:

$$P(\frac{c}{d}) \propto P(c) \prod_{1 \le k \le n_d} P(\frac{t_k}{c}) \quad (3)$$

where n_d is length of the document, $P(t_k/c)$ is the conditional probability of term t_k occurring in a document of class c, $P(t_k/c)$ as a measure of how much evidence t_k contributes that c is the correct class, P(c) is the prior probability of c".

Bernoulli is an alternative of Naïve Bayes if the data is spread according to multivariate *Bernoulli* distributions. *Multinomial* also is an alternative of Naïve Bayes, used merely in text classification, representing documents as word vector counts:

$$\text{Multinomial: } P\left(\frac{d}{c}\right) = P(< t_1, \dots, t_k, \dots, t_{nd} >/c) \quad (4)$$

$$\text{Bernoulli: : } P\left(\frac{d}{c}\right) = P(< e_1, \dots, e_i, \dots, e_m >/c) \quad (5)$$

$<t_1, \dots, t_{nd}>$ presents the list of terms appearing in the document d and $<e_1, \dots, e_i, \dots, e_m>$ is a vector with binary values that specify for particular term whether it occurs in the document or not (Manning et al., 2008).

In vector space classifiers, words are axis and documents are presented as points in the vector space defined by this axis.

Centroid classifier computes a centroid for each predefined class, allocating a new document to the class which centroid is closest to this document. The centroid of some class is calculated as the vector average of its documents:

$$\vec{\mu}(c) = \frac{1}{|Dc|}\sum_{d\in \mathrm{Dc}} \overrightarrow{v(d)} \quad (6)$$

where D_c is the set of documents in D whose class is c: $D_c = \{d : <d, c> \in D\}$.

Neighbour classifier is like the centroid classifier, but it allocates a new document to the class of its nearest neighbor in the training data:

$$P(c_j/S_k) = P(c_j/d) \quad (7)$$

where c_j represents the group of documents in the class c_j, S_k is the set of d's k nearest neighbors and $I_c(d') = 1$ iff d' is in class c and 0 otherwise, assigning the document to the class with the highest score.

Support vector machines are another vector space-based classifier that implements the following idea: input vectors are non-linearly mapped to a very high dimension feature space; in this feature space a linear decision surface is constructed (Cortes et al., 1995):

$$f(x) = \sum \propto_i y_i x_i^T x + b \quad (8)$$

where $a_1 \dots a_N$ are such that: $Q(\propto) = \sum \propto_i - 1/2 \sum\sum \propto_i \propto_j y_i y_j x_i^T x_j$ is maximized and $\sum \propto_i y_i = 0$, $a_i \geq 0$ for all a_i.

Linear Support Vector Classification is a Support Vector Machines for the case of a linear kernel. Linear classifiers compute a linear combination or weighted sum of the feature values, where the classification is done by comparing to some limit θ:

$$\sum_i w_i x_i > \theta \quad (9)$$

where $(x_1, x_2)^T$ is the two-dimensional vector representation of the document and $(w_1, w_2)^T$ is the parameter vector. The linear model is a simplification of ordinary linear

regression, which includes reaction variables with error distribution models apart from a normal distribution (Pedregosa et al., 2011).

Perceptron is a linear classifier, which involves a simultaneous update of each weight w_j in the weight vector w (Raschka, 2015):

$$w_j := w_j + \Delta w_j \tag{10}$$

where $\Delta w_j = l * (e - p) * x$, l - learning rate that must be configured (e.g. 0.01),, (e - p) is the prediction error for the model on the training data attributed to the weight and x is the input value.

Stochastic gradient descent classifier is a very efficient approach to fit linear models, mainly suitable once the total of samples is huge. It uses a linear activation function, cost function J(w), which is the Sum of Squared Errors (Raschka, 2015):

$$J(w)_{ridge} = \frac{1}{2} \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2 \tag{11}$$

where $\hat{y}$ is the predicted value $\hat{y} = w^T x$.

Ridge regression is a model which adds a penalty, using the squared sum of the weights to the least-squares cost function, aiming to reduce the complexity of a model by penalizing large individual weights (Raschka, 2015):

$$J(w)_{ridge} = \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2 + \lambda ||w||_2^2 \tag{12}$$

where: $\lambda ||w||_2^2 = \lambda \sum_{j=1}^{m} w_j^2$ and by increasing λ , we increase the regularization strength and shrink the weights of our model.

Passive/Aggressive algorithms are a type of algorithms for large-scale learning. They are like Perceptron in that they do not require a learning rate. However, opposite to Perceptron, they include a regularization parameter. This subsection cites material from (Pedregosa et al., 2011) to explain different GLM classifiers, and it is based on the Hinge loss function (Crammer et al. 2006):

$$L\varepsilon\left(w; (x, y)\right) = \begin{Bmatrix} 0 & |w \cdot x - y| \leq \varepsilon \\ |w \cdot x - y| - \varepsilon & otherwise \end{Bmatrix} \tag{13}$$

where ε is a non-negative constant which manages the relation to prediction mistakes. This algorithm work with this update rule:

$$w_{t+1} = \underbrace{argmin}_{w \in R^n} \frac{1}{2} ||w - w_t||^2 + C\varepsilon^2 \tag{14}$$

All the above described algorithms do not consider the analogy between words. They belong to the bag of word models, where each word is considered as a unique entity, giving no relevance to its internal structure. *FastText* on the other side treats each word as a bag of character n-grams. Given a dictionary of n-grams of size and a word w, by $G_w \subset \{1, \ldots, G\}$ has denoted the set of n-grams appearing in w; to each n-gram, g is associated a vector representation z_g., and a word is represented by the sum of the vector representations of its n-grams (Bojanowski et al., 2017):

$$s(w, c) = \sum_{g \in G} v_c \, z_g^T \tag{15}$$

Results

We have trained and then tested our articles measuring the accuracy for each classifier separately. The training/testing process is done iteratively for a different number of inputs, starting from 1000 sentences, and continuing in increasing order by 1000, until the level of 20000 sentences is reached. Table 2 illustrates the gained results for nine different approaches in classification (rows in tables). The values of a row corresponding to one classification approach for twenty input sets (columns in the table).

Table 2
Accuracy scores for nine classifiers, applied on twenty input sets with different size

	M	L	N	B	C	S	P	R	PA	AV
S1	0.67	0.74	0.80	0.53	0.74	0.77	0.84	0.73	0.77	0.73
S2	0.70	0.79	0.88	0.65	0.81	0.84	0.90	0.77	0.83	0.79
S3	0.73	0.82	0.87	0.67	0.82	0.86	0.90	0.80	0.85	0.81
S4	0.77	0.85	0.89	0.70	0.86	0.88	0.89	0.84	0.88	0.84
S5	0.75	0.86	0.88	0.69	0.85	0.88	0.91	0.85	0.88	0.84
S6	0.73	0.87	0.88	0.72	0.87	0.89	0.92	0.85	0.90	0.85
S7	0.71	0.85	0.85	0.73	0.85	0.87	0.90	0.83	0.89	0.83
S8	0.72	0.87	0.72	0.75	0.86	0.87	0.91	0.85	0.88	0.83
S9	0.72	0.87	0.72	0.76	0.86	0.87	0.92	0.86	0.90	0.83
S10	0.75	0.90	0.74	0.81	0.88	0.89	0.93	0.88	0.92	0.86
S11	0.76	0.90	0.75	0.78	0.89	0.88	0.92	0.88	0.91	0.85
S12	0.75	0.89	0.75	0.78	0.88	0.87	0.91	0.87	0.91	0.85
S13	0.76	0.90	0.76	0.80	0.89	0.87	0.92	0.88	0.92	0.85
S14	0.76	0.90	0.76	0.80	0.89	0.87	0.92	0.88	0.92	0.86
S15	0.77	0.91	0.77	0.82	0.88	0.89	0.92	0.90	0.92	0.86
S16	0.77	0.91	0.72	0.82	0.87	0.88	0.92	0.90	0.93	0.86
S17	0.75	0.91	0.70	0.82	0.89	0.87	0.92	0.89	0.93	0.85
S18	0.75	0.91	0.71	0.82	0.89	0.87	0.93	0.89	0.92	0.85
S19	0.76	0.91	0.73	0.83	0.89	0.87	0.93	0.90	0.93	0.86
S20	0.75	0.92	0.72	0.83	0.89	0.87	0.94	0.91	0.94	0.86

Note: The first column shortcuts: M - Multinomial, L- LinearSVC, N - Neighbour, B - Bernoulli, Centroid, S - SGD, P - Perceptron, Ridge, PA – PassiveAggressive; S_i in the heading is input set with 1000*i sentences; AV – Average
Source: Authors' work

Main metrics such as maximum and minimum value, average, mean, and deviation, are calculated for each classifier (Table 3).

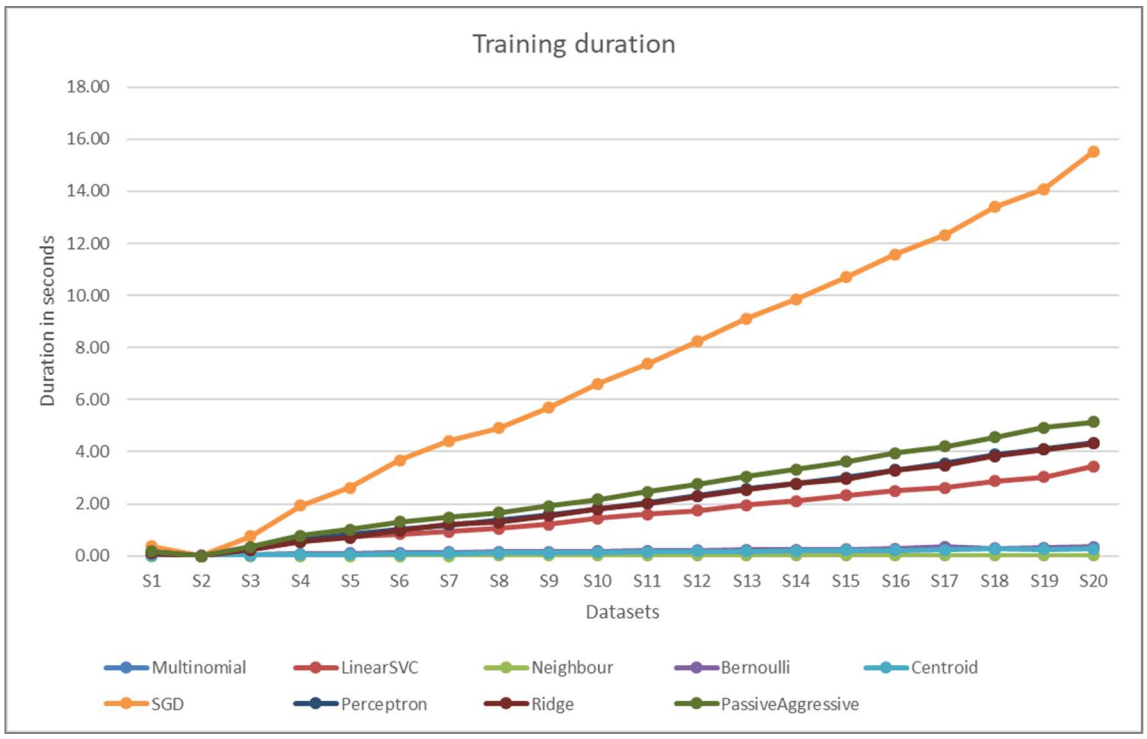
Table 3
Main Metrics for Investigated Classifiers

Classifier	MAX	MIN	Average	Median	Deviation
Multinomial:	0.77	0.67	0.74	0.74	0.03
LinearSVC:	0.92	0.74	0.87	0.87	0.05
Neighbour:	0.89	0.70	0.78	0.78	0.07
Bernoulli:	0.83	0.53	0.76	0.76	0.08
NearestCentroid:	0.89	0.74	0.86	0.86	0.04
SGD:	0.89	0.77	0.87	0.87	0.03
Perceptron:	0.94	0.84	0.91	0.91	0.02
Ridge:	0.91	0.73	0.86	0.86	0.05
PassiveAggressive:	0.94	0.77	0.90	0.90	0.04

Source: Authors' work

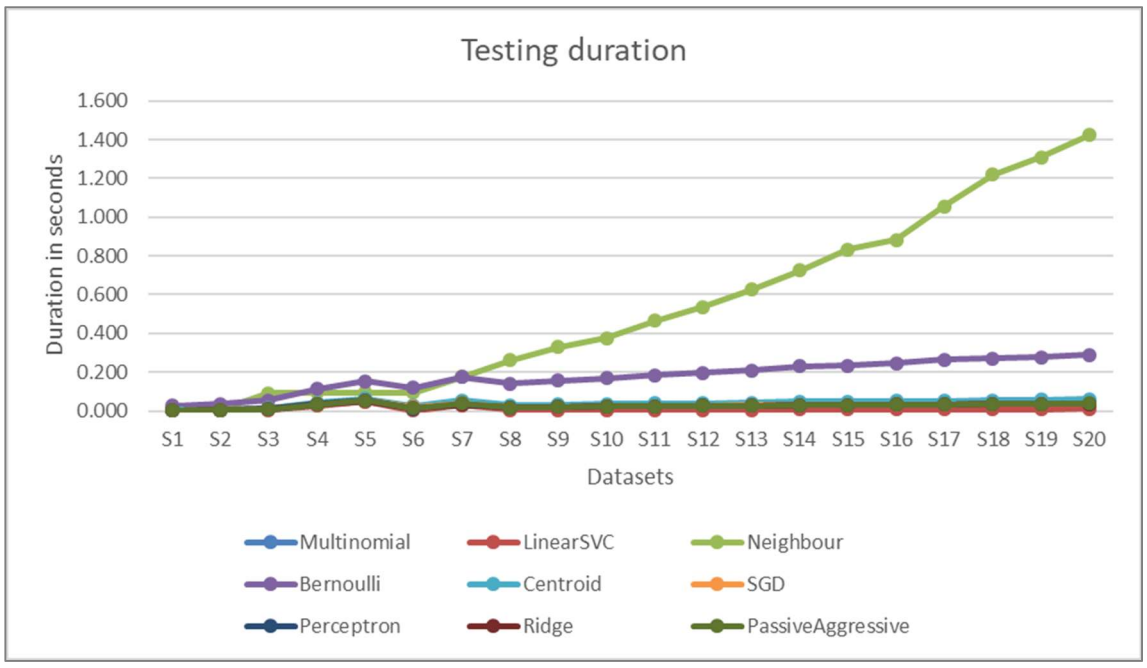
The training and testing time is measured too, for each classifier and each training/testing set. Figure 1 and Figure 2 give the training and testing time in seconds depending on the input size.

Figure 1
Classifiers Training Time



Source: Authors' work

Figure 2
Classifiers Testing Time



Source: Authors' work

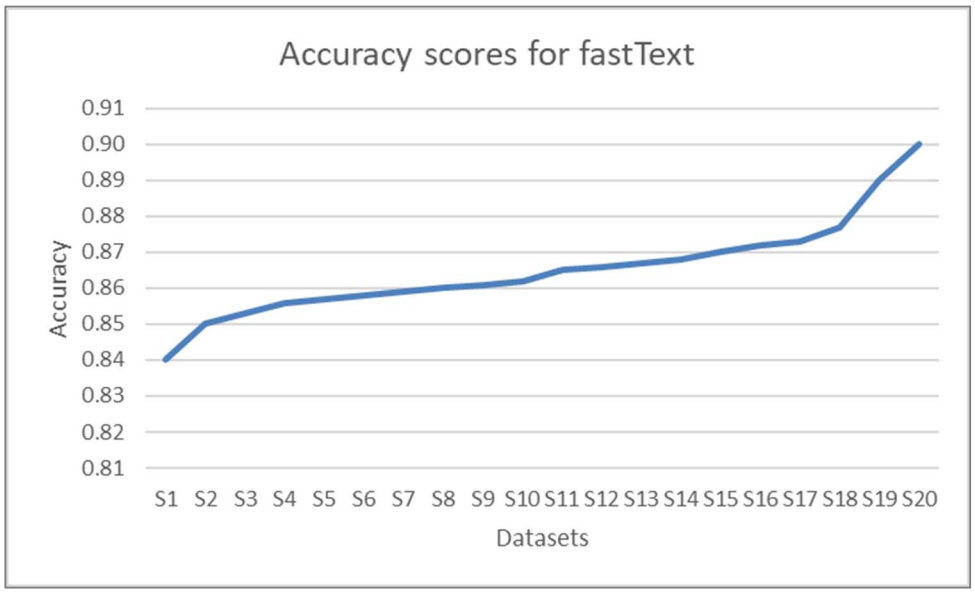
Table 4 gives the average, maximum and minimum values for accuracy score, training duration and testing duration when *fastText* is used for classification. Figure 3 gives a visualization of the performance of this classifier in terms of accuracy, while Figure 4 gives the training and testing time in seconds depending on the input size.

Table 4
Main Metrics for *fastText*

Metric	MAX	MIN	Average	Median	Deviation
Accuracy:	0.9	0.84	0.86	0.86	0.01
Training time:	12.29	2.65	7.64	7.36	3.03
Testing time:	2.27	2.15	2.215	2.21	0.04

Source: Authors' work

Figure 3
Accuracy scores for *fastText*



Source: Authors' work

Figure 4
Training and Testing Time for *fastText*

Source: Authors' work

Discussion

Table 2 shows the classifiers' accuracy scores when the bag of words model is used for representing words. Some classifiers perform nearly the same, and some perform very differently in terms of accuracy. What all classifiers have in common is the fact that they tend to increase classification accuracy with input size, but this increase is minimal even after the third input set. The last row of this table gives the average accuracy score, which for 1000 sentences is 0.73, for 2000 sentences 0.79, which is near to maximal average accuracy 0.86. One can notice that some of smaller sets have higher accuracy than some larger sets, but this difference is very low (0.01), and it happens only because of the random sample of the input.

Table 3 gives a better understanding of classifiers behavior. By observing the scores in this table, we can conclude that *Perceptron* and *Passive/Aggressive* classifiers show the highest accuracy of 0.94, but *Perceptron* has higher minimal accuracy (and the highest accuracy of all classifiers) of 0.84, which logically results in the highest average accuracy of 0.91. This classifier also has the highest mean (0.91) and lowest deviation (0.02). We can conclude that *Perceptron* gives the best results in terms of accuracy in a corpus of news documents written in Albanian. The "worst" classifier in this context happens to be *Multinomial* classifier, followed by *Bernoulli* classifier.

While accuracy remains the most important metric when classifying text, training time is also of vital importance in cases of a large input. Figure 1 illustrates the obtained outcome: SGD classifier slows rapidly when the size grows (15.52 seconds for the last set), giving an average of 7.16 seconds. This value is 4.72 seconds greater than the second slowest classifier *Passive/Aggressive* (2.44 seconds for training time). The fastest classifier for training is *Neighbour* classifier, with an average of just 0.01 seconds. Contrary to the training duration result, the testing duration measurement is the slowest for this classifier (Figure 2, average 0.53 seconds). Bernoulli classifier follows as the second slowest with an average testing time of 0.17 seconds. *LinearSVC* is the fastest classifier when testing (average of 0.01 seconds). All other classifiers perform solidly with an average testing time less than 0.028 seconds. Here we see some small "oscillations" of the values again because of the input randomness. *Perceptron* as a classifier with best accuracy results has an average training time 2.04 seconds and testing time of 0.026 seconds.

Table 4 shows that we have very stable accuracy performance when treating word vectors based on semantic and syntactic significance, with a minimum value of 0.84 and a maximum value of 0.9. Comparing with the classifiers from Table 3, we can say that only three of these classifiers perform worse than *fastText* in terms of accuracy scores.

Regarding the training duration, fastText has average training duration of 7.65 seconds, which is higher than all other classifiers (shown in Figure 1). For comparison, only SGD classifier has an average training equal to 7.16 seconds, three classifiers have average training duration little over 2, and the other classifiers have training duration time less than 2 seconds. Testing duration for fastText is 2.21 seconds, which again is higher compared to the testing duration for the other classifiers. For illustration, the highest testing duration from this group has Neighbour classifier (average of 0.53 seconds), followed by Bernoulli classifier (average of 0.17 seconds). The remaining classifiers have an average of fewer than 0.04 seconds.

All the used news articles are examples of single modality – they belong to a single class. Since *fastText* is aimed for fast classification of larger text, we have used this tool also to classify a corpus grounded on the news corpora available from the SETimes.com portal. The South-East European Times website is a news site which covers current events in the Balkans in ten languages: Bulgarian, Bosnian, Greek, English,

Croatian, Macedonian, Romanian, Albanian and Serbian (Tyers et al., 2010). We have utilized the Albanian-English corpus available online (Natural Language Processing Group, 2014). This corpus has about 228.000 sentences, all of them labeled regarding their category, and most of them have more than one label (they are multi-labeled). For example, the following sentence belongs to the *Politics* category, as well as to *Labor* category:

```
<tu>
  <prop type="x-keywords"><seg>Topic:Labor Topic:Politics</seg></prop>
  <tuv xml:lang="en"><seg>On paper at least, it looks like a great idea.</seg></tuv>
  <tuv xml:lang="sq"><seg>Në letër të paktën, duket si një ide e madhe.</seg></tuv>
</tu>
```

The first sentence is in English and the second one is in Albanian. We have extracted from this corpus all Albanian sentences, together with their categories, which are: *politics, labor, law_crime, integration, war_conflict, social_issues, business_finance, environment, human_interest, sports, hospitality_recreation, entertainment_culture, technology_internet, war_crimes, religion_belief, disaster_accident, weather, education*. In addition to these traditional categories, sentences are also categorized according to their source country: *Bosnia &_Herzegovina, Hungary, Albania, Macedonia, Turkey, Moldova, Croatia, Greece, Kosovo, Slovenia, Serbia, Bulgaria, Montenegro, Romania, Cyprus*. Moreover, there are categories for persons too, who are political influencers in these countries, which jointly with the above-listed categories produce 160 categories in total for gained Albanian corpus.

As it can be seen from the example above, sentences form this corpus are multi-label. This means that one sentence belongs to more than one category. FastText supports multiple labels for text classification. It was used to train and test this corpus, giving an accuracy of 0.79, training time 41 seconds, and testing time only 2.5 seconds.

The other classifiers do not support multi-label output. For this reason, the corpus is reformatted in such a way that each multi-label sentence is split into one-label sentences. The results were far below compared to the fastText classifier: the best accuracy result shows Ridge classifier with 45% accuracy, with training duration 324 seconds.

Conclusions

Text classification is very important in different text mining applications. For under-resourced languages, which have very little or no available categorized text corpora, it is difficult to investigate how different existing classification algorithms behave in the case of a specific language. Crawling news portals is an efficient way to achieve such a benchmark. We showed that this methodology gives valuable results in classifying text written in Albanian, concluding that Perceptron gives the best performance in terms of accuracy in a collection of news articles in this language. FastText shows better performance when classifying multi-label text. We plan in the future to extend the corpus with more pre-classified text from the web, adding new categories.

References

1. Antonellis, I., Bouras, C., Poulopoulos, V. (2006), "Personalized news categorization through scalable text classification", in Zhou, X., Li, J., Shen, H. T., Kitsuregawa, M., Zhang, Y. (Eds.) Frontiers of WWW Research and Development – APWeb 2006, Springer, Berlin, Heidelberg, pp. 391-401.
2. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T. (2017), "Enriching word vectors with subword information", Transactions of the Association of Computational Linguistics, Vol. 5, pp.135-146.
3. Chaudhari, S. V., Lade, S. (2013), "Classification of News and Research Articles Using Text Pattern Mining", IOSR Journal of Computer Engineering (IOSR-JCE), Vol. 14, No. 5, pp. 120-126.
4. Cortes, C., Vapnik, V. (1995), "Support-vector networks", Machine Learning, Vol. 20, No. 3, pp. 273-297.
5. Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., Singer, Y., (2006), "Online passive-aggressive algorithms", Journal of Machine Learning Research, Vol. 7, pp. 551-585.
6. Gui, Y., Gao, Z., Li, R., Yang, X. (2012), "Hierarchical text classification for news articles based-on named entities", in Zhou, S., Zhangs, S., Karypis, G. (Eds.) Advanced Data Mining and Applications, Springer, Berlin, Heidelberg, pp. 318-329.
7. Hartmann, N., Fonseca, E., Shulby, C., Treviso, M., Rodrigues, J., Aluisio, S. (2017), "Portuguese Word Embeddings: Evaluating on Word Analogies and Natural Language Tasks", in Proceedings of Symposium in Information and Human Language Technology, Uberlandia, MG, Brazil, pp. 122-131.
8. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T. (2016), "Bag of tricks for efficient text classification", in Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Vol. 2, Short Papers, pp. 427-431.
9. Jurka, T. P., Collingwood, L., Boydstun, A. E., Grossman, E., van Atteveldt, W. (2013) "RTextTools: A supervised learning package for text classification", The R Journal, Vol. 5, No. 1, pp. 6-12.
10. Liparas, D., HaCohen-Kerner, Y., Moumtzidou, A., Vrochidis, S., Kompatsiaris, I. (2014), "News Articles Classification Using Random Forests and Weighted Multimodal Features", in Lamas, D., Buitelaar, P. (Eds.), Multidisciplinary Information Retrieval, Springer, Cham, pp. 63-75.
11. Manning, C. D., Raghavan, P., Schutze, H. (2008). Introduction to Information Retrieval, New York, Cambridge University Press.
12. Mikolov, T., Chen, K., Corrado, G., Dean, J. (2013), "Efficient estimation of word representations in vector space", in Proceedings of the International Conference on Learning Representations (ICLR 2013), available at: https://arxiv.org/pdf/1301.3781.pdf
13. September 2013).
14. Natural Language Processing Group (2014). Web corpora of Bosnian, Croatian and Serbian top-level domain published, available at: http://nlp.ffzg.hr/web-corpora-of-bosnian-croatian-and-serbian-top-level-domain-published/ (7 September 2014).
15. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V. Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E. (2011), "Scikit-learn: Machine Learning in Python", In Journal of Machine Learning Research, Vol. 12, pp. 2825-2830.
16. Raschka, S. (2015). Python machine learning, Birmingham, Packt Publishing Ltd.
17. Rubin, T. N., Chambers, A., Smyth, P., Steyvers, M. (2012), "Statistical topic models for multi-label document classification", Machine Learning, Vol. 88, No. 1-2, pp. 157-208.
18. Scannell, K. P. (2007), "The Crúbadán Project: Corpus building for under-resourced languages", in Fairon, C., Naets, H., Kilgarriff, A., de Schryver, G. M. (Eds.), Building and Exploring Web Corpora, Proceedings of the 3rd Web as Corpus Workshop, Vol. 4, pp. 5-15.
19. Swezey, R. M., Sano, H., Shiramatsu, S., Ozono, T., Shintani, T. (2012), "Automatic detection of news articles of interest to regional communities", International Journal of Computer Science and Network Security, Vol. 12, No. 6, pp. 99-106.

20. Tyers, F. M., Alperen, M. S. (2010), "South-east European times: A parallel corpus of Balkan languages", in Proceedings of the LREC Workshop on Exploitation of Multilingual Resources and Tools for Central and (South-) Eastern European Languages, pp. 49-53.
21. Zhou, D., Resnick, P., Mei, Q. (2011), "Classifying the Political Leaning of News Articles and Users from User Votes", in 5th International AAAI Conference on Web and Social Media, North America, pp. 417-424.

About the authors

Arbana Kadriu holds a Ph.D. degree in Computer Sciences from Ss. Cyril and Methodius University in Skopje from 2008, with a focus on natural language processing and information retrieval. She is an associate professor at the Faculty of Contemporary Sciences and Technologies at SEE University in Macedonia. She has also background in artificial intelligence, machine learning, programming paradigms, software engineering, and e-learning. Also, she is mentoring several master thesis that involves web information retrieval and e-learning. She is the author of more than 30 research papers. The author can be contacted at **a.kadriu@seeu.edu.mk**.

Lejla Abazi-Bexheti is Associate Professor at the Faculty of Contemporary Sciences and Technologies at South East European University in Macedonia. She holds a Ph.D. Degree in Computer Science and has been part of the CST teaching staff since 2002. During her teaching experience, she has taught courses from the area of algorithms, programming and web programming. Her main research activity is in the area of Learning Systems and eLearning, and she has been involved in many projects and research activities from this area. At SEE University she was involved in resolving the issue of the Learning Contents and Learning Management System. She was Director of eLearning Center at SEEU and managing Online Studies at SEEU for the period 2006-2014. Currently, she is Pro-rector for academic issues at SEEU. The author can be contacted at **l.abazi@seeu.edu.mk**.

Hyrije Abazi-Alili finished her Ph.D. in Economics at Staffordshire University, the UK in 2013. She is engaged as Lecturer Assistant of Economics, Quantitative Methods and Corporate Finance courses at the Faculty of Business and Economics since 2005. Her current position is Research Group Leader at SEEU. She is currently holding a teaching fellowship at CERGE-EI, Prague. Her field of research is on: (i) social issues social inclusion, female labor force participation, education, remittances, poverty, etc.; and (ii) firm performance - innovation activities, R&D, ICT in business, knowledge spillovers, skilled workers, etc. The author can be contacted at **h.abazi@seeu.edu.mk**.