Development of System for Auto-Tagging Articles, Based on Neural Network

Pavlo Mukalov[1] [0000-0002-0808-5809], Oleksandr Zelinskyi[2] [0000-0003-1247-7511], Roman Levkovych[2] [0000-0001-9393-714X],Petro Tarnavskyi[2] [0000-0003-3265-8168], Anastasiia Pylyp[1] [0000-0002-0222-4687], Nataliya Shakhovska[1] [0000-0002-6875-8534]

[1]Lviv Polytechnic National University, Lviv 79013, Ukraine

[2]Ivan Franko National University, Lviv 79000, Ukraine

pmykalov@gmail.com, sashko.zel2000@gmail.com, rlevkovych098@gmail.com, petro28062000@gmai.com, anastasiia.pylyp@gmail.com, nataliya.b.shakhovska@lpnu.ua

Abstract. The paper describes possibilities of natural language processing in data classification. In last decade AI technologies became widespread and easy to implement and use. One of the most perspective technology in the AI field is natural language processing. New technologies will become a central part of future life because they save a lot of time. In addition, the articles shows a complete article tagging cycle using Neural Networks, ranging from data acquisition to tag storing.

Keywords: auto-tagging, language processing, neural network with LSTM layers, multilayered system.

1 Introduction

Natural language processing or NLP is a part of computer science and artificial intelligence associated with the interactions between computers and human (natural language). The main tasks of NLP are data extraction, speech synthesis, language generation, speech recognition, machine translation, information receiving and many others.

That is why NLP used in many spheres of life from auto-supplementation in the iPhone to marketing and advertising. Many modern resources provide an analysis of trends over the past few years, while no one focus on predicting the popularity of things for today.

The basis of project, given in this paper, is the processing of articles from popular web forums using neural networks. The project now focuses on articles processing and anticipating trends in the IT industry. That is why this project can be useful when people choose a stack of technologies for their new project.

2 State of arts

Today, the basic solution to the problem of recognition of entities is a combination of gazers, basic rules and the Conditional random field (CRF). CRF is one of the classic machine learning algorithms.

Such a set of algorithms was used, for example, as a baseline in the WNUT2015 competition [1]. Most of the participants used CRF, as well as classic forward-propagation neural networks (FFNN) and Markov algorithms. In addition to the texts analysis, many participants also used the meanings of vector representations of words: word embedding, using the word2vec and GloVe algorithms.

In [2], the authors propose an algorithm for automatically constructing newspapers based on WordNet and Wikipedia by identifying the type of entity. The algorithm is moving up through the hierarchy of hypernames. The method shows rather weak results for such types of named entities as persona and organization, but it works better for geographic locations. In addition, it is limited to the data available in the mentioned systems. As far as we know, this method of development has not received. Today, rule-based systems are considered rather primitive, suitable only for automating the process of extracting information, which is already quite well structured. The main disadvantage of rule-based systems is their limitations, that each new knowledge section requires the development of its own set of rules capable of taking into account the specificity of texts in this area, which requires the involvement of a large amount of human resources. At the same time, the performance of more automated systems based on machine learning algorithms has increased enough to compete with the best rule-based systems.

In [3], the authors stated the possibility of developing a rule-based system that can be compared in quality with machine learning algorithms, if you first spend 8 person-weeks to develop rules for a specific subject area.

Speaking of machine learning algorithms, it is worthwhile to separate the algorithms by two groups. The first one is learning with the teacher, when algorithm is trained in sufficiently pre-marked manually examples. The second one is learning without the teacher. In this case, algorithm learns to recognize entities using only the information provided in the processed data and some previously known heuristics. Algorithms with a teacher have a disadvantage akin to rule-based systems: their training requires a rather time-consuming process of preparing training data.

Among machine learning algorithms with a teacher, most of the classical methods reduce the task of recognizing entities to the markup of sequences and their subsequent element-by-element classification. From more specific examples, we can highlight the CRFs mentioned above. CRF is one of the most popular search patterns for named entities, defining tags based on attributes, but taking into account both the current and previous words and the subsequent words in the text. So, this algorithm forms the basis of a number of popular sequence markers [4-5].

The next group of algorithms for sequences labeling is based on maximum entropy [6]. They predict the label of a sequence element based on the probabilities of occurrence of certain attributes of a word and its predecessors, and Markov models that perceive text markup as a Markov process, where states are the required classes , and the probabilities of the labels of the current element are determined by the previous state of the process.

More sophisticated sequence classification algorithms can rely on complex neural network models, such as LSTM, which has gained popularity in working with text data due to its ability to take into account the history of sequences skipped through it. Examples of using such models can be found in [7]. The use of bidirectional LSTM networks makes it possible to simultaneously take into account the attributes of both previous and subsequent words in a sentence when assigning an entity tag.

In [8], the authors compare the performance of unidirectional and bidirectional LSTM with use of CRF at the network output to account for tagged adjacent words and improve quality. Unlike algorithms with a teacher, unsupervised algorithms often identify entities in the text based on the search for similar words in a document, in an attempt to identify named entities in common groups, based on context. An example of this approach is [9, 10], in which authors use Word2vec to generate clusters of words with similar contexts. This approach shows the best results in comparison with the classical CRF for languages with a low volume of labeled cases.

C-LSTM Neural Network for Text Classification is used in [11]. Authors use Convolutional neural network (CNN) and recurrent neural network (RNN) for sequence extraction with higher-level phrase representations.

In [12, 13] the method of text classification based on Big data approach and pattern recognition is proposed.

However, all these approaches are just mostly for text classification.

The purpose of this paper is to design a system of looking for articles, auto-tagging them and demonstrating results using RESTful subsystem.

3 Main part

The system architecture and main methods for text analysis will be proposed in the paper.

3.1 System architecture

The proposed system consists of three main parts:

1. Data providing – exploitation open source data from open web resources using their API.
2. Data processing – processing of articles using neural networks.
3. Calculation of statistical indicators.

The architecture of the system is represent in the Fig. 1.

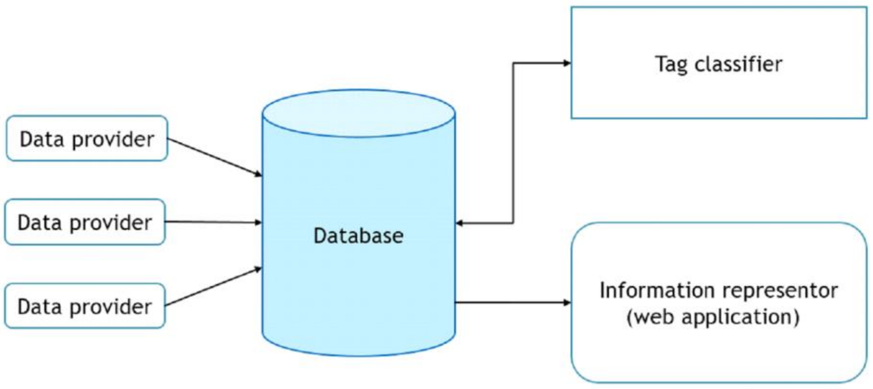


Fig. 1. The system architecture

From the beginning, Data providers scraped the text of articles from the sites of open web resources using their APIs and sent to the database. Than Tag classifier obtained data from the database, processes the text and writes the results to the same database. The Information representor retrieves data from the database and displays them on a web site.

The Azure functions are used to enable data providers from time to time without explicitly managing their call, Html Agility Pack for web scraping, Python 3.6.6 and Keras are used for article typing, also SQLAlchemy and Entity Framework for working with the database, React JS is used to display data on a site.

3.2 Data Providers

In order to determine the popularity of certain tags, we need to find which percentage of articles has this tag. The more articles are processed, the more precisely this statistic is. Manually processing articles from different sites would take a lot of time. An optimal solution to this task is to parsing an article by its reference. In order to find a reference to an article we need to parse the site tab “Newer”, “Latest”, etc.

A good tool for parsing web pages is HtmlAgilityPack. With this tool, own parsers were created. Using parsers, we get references to new articles. The other parser processes each received reference with the article, after which all the necessary information about the article is transferred to the database. The design of data parsers is such that it is possible to carry out periodic diagnostics for them. This diagnostic is required to check whether the site markup for which the parser was written has not changed. If the site markup has changed then this diagnostic will inform the developer about it. If the site markup has changed then developer should change the parser for this site.

In order for our parsers to work automatically, we decided to use the Azure features. Azure Functions is a great solution running functions, such as our parsers, in the cloud. Write only the code for the problem in the locale and do not worry about how to run it. Azure Functions can be used with different languages, such as C#, F#, Node.js, Java, or PHP. Azure Function automatically starts every 24 hours and adds new articles to the database, which improves the accuracy of our statistics.

3.3 Text pre-processing

Before training models, texts are processed according to the following principles:

- multi-line texts are combined into one line;
- texts are cleared of all characters that are not letters, numbers, space characters, or some special characters;
- each token is subjected to morphological analysis and is reduced to normal form (if possible);
- for normalized tokens, the mark of a part of speech is added;
- removal of the service parts of speech (conjunctions, prepositions and pronouns).

3.4 Text processing

To begin with, we decided to use neural network with LSTM layer to classify article by tag, because there it allows us configure how many previous sentences will influence current output of neural network. As text can be spelled by "." and fitted to neural network sentence by sentence. Next, a sentence is transformed and passed through the network LSTM (Long short-term memory). The layer remembers the sentence and influences output of neural network in feature.

The Word2vec can be used for text processing. This approach is presented in the form of two variations of the neural network architecture containing a single hidden layer. The final model, relying on the distributive hypothesis (linguistic units with similar distributions have a similar meaning), learns to match the words and contexts of their use. Training takes place without the help of a teacher, using only unplaced texts, producing at the output a set of vectors of a given dimension for any word encountered in the learning process. At the same time, the resulting vectors reflect the closeness of these words: closer words have closer vectors and vice versa. The positive characteristics of this model are the low rarefaction of the final vectors, the ability to set their dimensions, as well as the speed of operation (compared to more complex models that give a similar level of quality). The main disadvantage is the impossibility of interpreting the values of the coordinates of some vector. To obtain a vector representation of the whole text, it is necessary to combine vector representations of individual words, which is carried out, as a rule, by taking the mean value of the vectors.

That is why we propose to use Paragraph2Vec. This model has an architecture similar to word2vec, with the only difference that, in addition to contextual words, the model also takes into account the contextual document, learning in the process of learning and its vector representation. As a result, paragraph2vec is able to return vectors of whole texts of similar quality with vectors of individual words to word2vec. At the same time, for previously not met documents, the vector can be generated based on the words included in the document. Thus, using paragraph2vec, you can get vector representations of texts without any additional actions (Fig. 2).

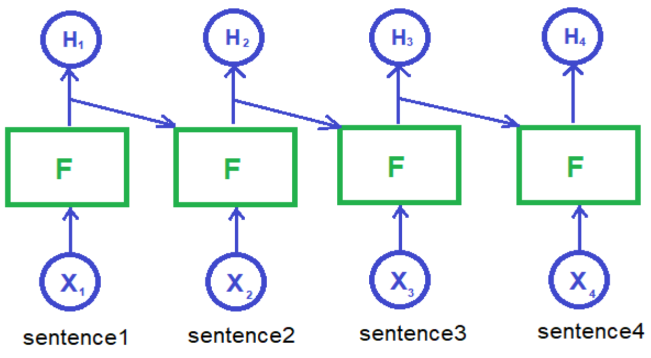
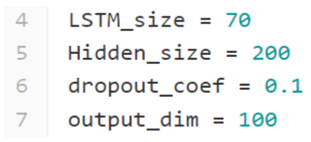


Fig. 2. The LSTM representation

Actually the network remembers previous sentence and it allows the network set a linguistically covariance between the words in different sentences. Besides, LSTM layer lets us configure how much time it will remember the previous input. The most effective setting is to remember 5-6 sentences as in articles the most paragraph consists of such an amount of sentences.

Architecture of neural network is given below:



```
LSTM_size = 70
Hidden_size = 200
dropout_coef = 0.1
output_dim = 100
```

There is no static input size because it defines dynamically from training sample. Every input neuron is a word that will be known by the neural network.

3.5 Data processor responsibilities and design

Data processor is an independent hostable service that is responsible of data organizing and processing it in database. This part works only with database and has no external dependencies in project.

For this part of system, we used Keras [14 – 15] for neural network, SQLAlchemy for interaction with database, Pandas [16 – 18] for loading sample from .csv file. For subsystem deploying to Azure we used Docker container.

Logical entities:

1. Classifier - provides generalized wrapper for RNN created using Keras with methods for training and configurating it.
2. TextClassifier - inherits Classifier, expands functionality of Classifier with possibility of preparing training sample (text to sequences).

3. Interactor - implementation of repository design pattern that provides a set of SQL queries created with SQLAlchemy as ORM.
4. MSSQLInteractor - inherits Interactor. Connects to MSSQL server (connection can be configured in config.py)

Design layers:

1. Classifying - includes all business logic of auto tagging of articles.
2. Data interaction - includes logic of storing intermediate data and interaction with database.
3. Entry - includes warming up Classifier and services logic.

4 Results

As far as it concerns training sample we created sample that consists of 200 articles about programming and tagged them.

The Table 1 presents the parameters of training. The both methods – Word2vec and Paragraph2vec, were compared. Basically, to predict a word, Word2Vec uses its surrounding words as predictors.

Paragraph2Vec, on the other hand, uses the resident paragraph id as an additional predictor. After the algo finishes, it has learned an embedding for each word and an embedding for each paragraph.

Table 1. The parameters of training.

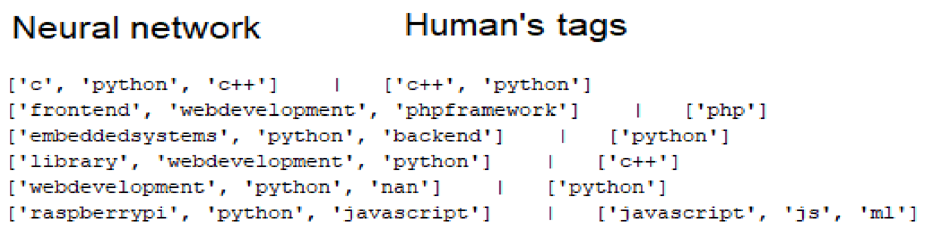
Model	Architecture	Dimension	Min frequency	Epoch
Paragraph2vec	PV-DM	200	3	5
Paragraph2vec	PV-DBOW	200	3	5
Word2vec	skip-gram	300	3	5
Word2vec	CBOW	300	3	5

Paragraph Vector is capable of constructing rrepresentations of input sequences of variable length. Unlike some of the previous approaches, it is general inapplicable to texts of any length: sentences, paragraphs, and documents. It does notrequiretask-specifictuningofthewordweightingfunctionnordoesitrelyontheparsetrees.

Then after 100 epoch of training with 30 batch size we have got the next result:

Loss: 0.0262
Accuracy: 0.9806
Value loss: 0.0649
Value accuracy: 0.9641

As to testing it on non-automatically tagged articles we have got the next result:



```
Neural network                        Human's tags

['c', 'python', 'c++']    |    ['c++', 'python']
['frontend', 'webdevelopment', 'phpframework']    |    ['php']
['embeddedsystems', 'python', 'backend']    |    ['python']
['library', 'webdevelopment', 'python']    |    ['c++']
['webdevelopment', 'python', 'nan']    |    ['python']
['raspberrypi', 'python', 'javascript']    |    ['javascript', 'js', 'ml']
```

Fig. 3. Comparison of human and system tags

As you can see the even with such a small training sample the neural network starts to understand content of articles written by human. The bigger sample - the more accurate result of classifier.

For training neural model, we tried different configuration and chosen the most effective one. It started to understand an essence of an article and tag it as a human. The main purpose is to train with a bigger sample and increase number of tags that the AI model know.

After first batches of articles analysing, we have seen that the most popular back-end language is Java and Python, as far as it concerns front-end the first place was acquired by Javasript. Not all other statistics can be defined as objectively correct due to low sizes of sample.

5 Conclusion

In this paper, we introduced our approach to solve article auto tagging problem. In order to find a solution, NLP were considered as the best option that suits our requirements. The method for determining the type of entity when extracting information from texts by calculating the semantic proximity of vectors obtained using neural network language models was proposed and experimentally investigated. The method has the advantage of low laboriousness of text corpus preparation in comparison with traditional methods of learning with a teacher and methods based on rules. The experiment also showed the advantage of using word2vec model vectors without TF-IDF or SIF weighting schemes in conditions of limited vocabulary of texts from the knowledge base, automatically generated from professional standards [20].

Taking everything into consideration a system that consists from four main parts (data providers, database, tag classifier and information representor) were built. The project automatically gets certain web-sites, save data in database, then the neural network handles new information and saves it.

Finally, the system has not been built into one app and tested. Each part of the system was checked and it works correctly.

The proposed system can be used for authorship recognizing [21 – 22], for data imputation in user profile [23] too.

References

1. Baldwin, T., de Marneffe, M. C., Han, B., Kim, Y. B., Ritter, A., Xu, W.: Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In: Proceedings of the Workshop on Noisy User-generated Text, 126-135 (2015)
2. Toral, A., Munoz, R.: A proposal to automatically build and maintain gazetteers for Named Entity Recognition by using Wikipedia. In: Proceedings of the Workshop on NEW TEXT Wikis and blogs and other dynamic text sources (2006)
3. Chiticariu, L., Krishnamurthy, R., Li, Y., Reiss, F., Vaithyanathan, S.: Domain adaptation of rule-based annotators for named-entity recognition tasks. In: Proceedings of the 2010 conference on empirical methods in natural language processing Association for Computational Linguistics, 1002-1012 (2010)
4. Lehman, Jill Fain.: Adaptive parsing: self-extending natural language interfaces. In: Springer Science & Business Media, vol. 161 (2012)
5. Finkel, J. R., Grenager, T., Manning, C.: Incorporating non-local information into information extraction systems by gibbs sampling. In: Proceedings of the 43rd annual meeting on association for computational linguistics, 363-370 (2015)
6. Toutanova, K., Manning, C. D.: Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In: Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics, vol. 13, 63-70 (2000)
7. Chiu, J. P., Nichols, E.: Named entity recognition with bidirectional LSTM-CNNs. In: Transactions of the Association for Computational Linguistics, vol. 4, 357-370. (2016)
8. Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF models for sequence tagging. In: arXiv preprint arXiv:1508.01991 (2015).
9. Shakhovska, N., Shvorob, I.: The method for detecting plagiarism in a collection of documents. In: 2015 Xth International Scientific and Technical Conference of Computer Sciences and Information Technologies (CSIT), 142-145 (2015)
10. Shvorob, I.: New Approach for Saving Semistructured Medical Data. In: Advances in Intelligent Systems and Computing, 29-40 (2017)
11. Zhou, C., Sun, C., Liu, Z., Lau, F.: A C-LSTM neural network for text classification. In: arXiv preprint arXiv:1511.08630 (2015).
12. Shakhovska, N.B., Noha, R.Y.: Methods and tools for text analysis of publications to study the functioning of scientific schools. In: Journal of Automation and Information Sciences, vol. 47(12) (2015)
13. Shakhovska, N., Vovk, O., Hasko, R., Kryvenchuk, Y.: The method of big data processing for distance educational system. In: Conference on Computer Science and Information Technologies, 461-473 (2017)
14. Antonio Gulli, Sujit Pal.: Deep Learning with Keras: Implementing deep learning models and neural networks with the power of Python. ISBN: 978-1787128422 (2017)
15. Keras Documentation. At: https://keras.io (2019)
16. David Taieb.: Data Analysis with Python.Packt Publishing. ISBN: 9781789958195 (2018)
17. Fabio. M. Soares, Rodrigo Nunes.: Neural Network Programming with Python. ISBN: 978-1784398217 (2019)
18. Sarkar, Dipanjan: Text Analytics with Python: A Practical Real-World Approach to Gaining Actionable Insights from your Data. ISBN 978-1-4842-2388-8 (2018)

19. Pranjal Srivastava: How to create a poet / writer using Deep Learning (Text Generation using Python)? At: https://www.analyticsvidhya.com/blog/2018/03/text-generation-using-python-nlp/ (2018)
20. Chapman, Nigel P., LR Parsing: Theory and Practice, Cambridge University Press. ISBN 0-521-30413-X (1987)
21. Vysotska, V., Kanishcheva, O., Hlavcheva, Y.: Authorship Identification of the Scientific Text in Ukrainian with Using the Lingvometry Methods. In: 2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT) , vol. 2, 34-38 (2018)
22. Lytvyn, V., Vysotska, V., Burov, Y., Bobyk, I., Ohirko, O.: The Linguometric Approach for Co-authoring Author's Style Definition. In: 2018 IEEE 4th International Symposium on Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS), 29-34 (2018).
23. Fedushko, S., Syerov, Yu., Korzh, R.: Validation of the user accounts personal data of online academic community. In: IEEE XIIIth International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science, 863–866 (2016).