Development of Legal Document Classification System Based on Support Vector Machine

Iuri Nasu

IT in Business Department
HSE University in Perm
Studencheskaya street, 38, Perm, Perm Krai, 614070
yunas2002@iCloud.com

Viacheslav V. Lanin

IT in Business Department
HSE University in Perm
Studencheskaya street, 38, Perm, Perm Krai, 614070
vlanin@hse.ru

***Abstract*—This paper was prepared while developing text classification system for legal documents, especially those that issued by Legislative Assembly of Perm Krai. The problem in question is a lack of solutions which meet regional requirements, the main of which is the classification used in region. The research that evaluates applications of Natural Language Processing models is conveyed. The primary result of the study is the actual applicability of Support Vector Machine (SVM) to preprocessed legal document categorization. There were a server-side API constructed to perform the task, and a server-side models pre-trained of which SVM is favored.**

***Index Terms*—FastAPI, Legal Tech, Supervised Learning, Support Vector Machine**

I. INTRODUCTION

The number of normative legal acts in force only at the federal level in Russia is just below 10 000[1]. That quantity of documents is clearly impossible for any lawyer to memorize or access swiftly, not taking into account other professions who also base their work on legislation (e.g., bookkeepers, accountants, engineers, etc.).

The changeability of Law in this situation only aggravates it. New Legal Acts enter into force every month. The volume of the legislative text increases, hence may be a factor that hinders its perception and, thus, execution. Suffice it to point that the rapid tracking of novel legal acts has become possible only due to the development of the Internet.

A partial solution to the problem in question can be a construction of an IT system for classification of legal acts by categories (i.e., classes) and its further implementation to Legal Knowledge Bases. This approach will certainly not reduce the number of legal documents (this requires cultural measures, not technological ones), but it will facilitate the work with legal information.

The system designed and this paper focuces mainly on solving the problem of classification of legal acts. Thus, the process of classification of legal acts, now carried out by experts, is automated.

II. State of Art

Legal Tech systems might be divided in two categories: the ones which automate or facilitate routine organizational business process within legal firm, departments and even courts (e.g., online dispute resolution, chatbots, online consulting, etc.) and ones which work with legal data and automate different legal processes (e.g., automated document analysis, document builders, legal libraries, etc.).

This work will focus on the second type of systems. Several of them should be noted before future exploration. There are *Consultant Plus* assistant system, *Garant* and the *Ministry of Justice'* web portal. They allow user to search Legal Acts, Court Decisions. *Consultant* and *Garant* also provide document builders and several other services such as calculators, online consultations, etc. Each of those systems has its variety of the paid Professional Versions.

The only reason those solutions could not be used is the customer's requirement to use ***regional classification*** (see Appendix 1) for processing.

III. The Task

The main purpose of the system is its application for building catalogues of documents based on their topic/topics. It will make search easier for legislators and lawyers.

Hence, the task in question is to build document classification system that could help to organize files. The automized process could be described as mapping

document (its text) to a class. Thus, the system takes in a file and yields a string name of a class.

The mathematical definition of the task is finding an approximation of the function that maps the set of documents to the set of categories. Each of the documents is a set of lexemes, thus the task could be described as finding a function that maps sets of sets of lexemes to set of classes.

As the words may have different semantic weight, those that contain less meaning (so called stop-words) might be omitted during the analysis enhancing the speed of processing.

IV. Requirments

The vital part of information system development is gathering customer's requirement. Thus, the chances to meet their needs increases. During the interview with the customer the list of requirements and their proper definitions of done were formulated. They are listed in Table 1.

Table 1. Requirements

Requirement	Definition of Done
The System is trustworthy	The System's accuracy metric is no less than 80% on both Orders and Laws
The System accepts 'docx' and 'docm' files	The System takes in 'docx' and 'docx' file and yields correct result. On attempt to put another file the System returns an error message and continues the work.
On damaged document returns respective message	On damaged document a Server returns respective message
Client-Server Architecture	Client and Server is separated
The System is easy to understand	The System returns the code and the name of the category
The system is easy to modify	There must a brief documentation

V. Technologies

As is known text classification is a supervised learning task. Hence, we need a test sample of already classified texts, which was kindly provided by the customer.

Considering the most efficient stack of technologies, we should consider the predominant use of Python programming language for solving that kind of tasks (first of all, due to the quantity of libraries written for this language). The libraries often used are *pandas*, *matplotlib* (for data analysis), *word2vec* and *sklearn* libraries (for machine learning models). Use of any other language would lead to solving many already decided problems (from preprocessing to building the logic of the classification model itself) from scratch.

Now that the choice of the main programming language is resolved, the Python Web framework must be chosen. Using FastApi is in the context of this context proves as the most effective solution, primarily because of its speed [2].

VI. Exploring Data

As is known, Legal Documents have the structure which could be used to identify and parse them. Documents usually contain a Title, an Issue Number, Dates of Registration and Adoption (that could easily be confused while parsing), an Authority that signs the document. It should be noted that several documents may have the very same Title or Number. Thus, it requires several parameters to identify the document.

In our research we focus on two types of Legal Acts: Laws and Orders. They severely differ in their structure. Laws are usually voluminous documents which have much less references to another laws (usually 1-2) per page than Orders, which are based on Laws and often are one-page long. Classification system may malfunction on such a little amount of unique data as present in Orders. It is worth to use references to already processed documents to classify them properly (especially, if those Orders are 0.5-1 page long).

The other problem which is met is the data provided. The regional document collection, which is 3000 items long, is obviously unbalanced. The distribution of documents over the classes is shown on Figure 1.

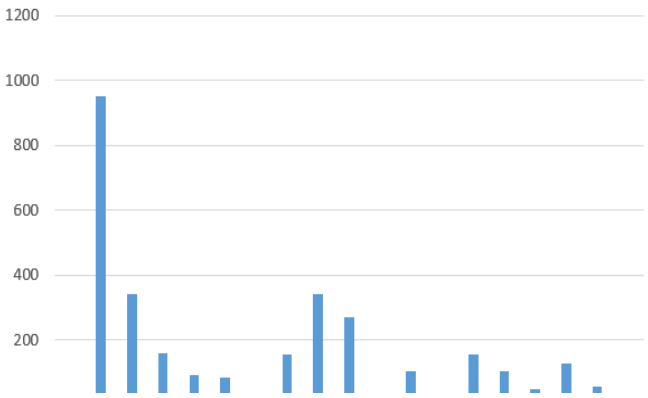

Fig. 1. Distribution of Documents

Classes 06. “Labor and Employment”, 10. “Interregional Economic Affairs”, 12. “Information and Informatization”, 20. “International Affairs” are represented by less than twenty documents each (they may also contain subclasses within them). To solve this issue, two solutions might be utilized:

1. Adding Legal Documents from other regions (or the ones accepted by the Federation);
2. Using models which sustain the imbalance.

This research focuses mainly on the second way, but it proves that the first is efficient as well.

VII. Classification model

There were three classification models used [3]:

1. Naive Bayes Classifier,
2. Support vector machine,
3. Logistic regression.

Tokenization, stop-words removal, and lemmatization had been performed before passing the training data to TF-IDF vectorizer. Then the data was passed to classifiers.

To select a model, each of them was built and trained on the very same sets. Thus, we might choose the best one. The metrics shall be listed in Table 2.

Table 2. Models and Metrics

	Naive Bayes	SVM	Logistic Classification
Accuracy. 600 items	0.17	0.48	0.23
F1-Score[1] 600 items	0.18	0.50	0.25
Accuracy. 750 items	0.3	0.6	0.37
F1-Score 750 items	0.3	0.61	0.39
Accuracy. 900 items	0.34	0.7	0.6
F1-Score 900 items	0.36	0.71	0.66
Accuracy. 1500 items	-	0.76	-
F1-Score 1500 items	-	0.77	-

As is seen, Support Vector Machine proves to be more accurate than its counterparts (due to the ability to describe even the smallest classes), therefore it is chosen to be used in production. The metrics listed prove that quality might be improved via using the vaster dataset. The only drawback is difficulty of interpretation (find more detailed works on this topic in [4]– [10]).

VIII. Ways to Improve

As is said the main problem is unbalanced data which worsens results (fig. 2). Thus, focusing on less presented classes is the way to improve the whole picture.

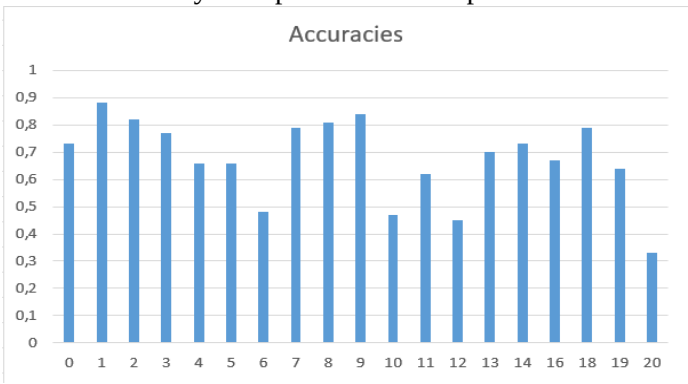

Fig. 2. Accuracy measures for classes

The very first addition made is filtering non-Legal Documents. It could be done two ways: adding a filtering model (Legal/Non-Legal Document Classifier), using the structure of the document that is provided by the user.

Those two ways could be combined in different ways: structure-oriented and content-oriented. The first approach mainly focuses on similarity between the document structures (i.e., document provided by the user is compared with “typical” document which we use to measure incoming), the second evaluates the content inside the document.

Let us compare the advantages and disadvantages of both these approaches. The structure-oriented approach is easier to perform, there is little if any machine learning required. Its main flaw is a subsequence of simplicity—the “typical” structures of Legal Acts allow little flexibility. Thus, it would not pass Legal Documents that does not follow the structure but those Non-Legal Documents that do.

The second approach is a little more complex. It requires to gather different kinds of Non-Legal Documents and build a binary classifier. Suffice it to say the classifier should be adjusted to detect all the Legal Documents.

Thus, the model should bolster recall over precision due to false positives (i.e., Non-Legal Documents passed) be not such a problem as inability to classify Legal

[1] Macro-averaged.

Document. Nevertheless, the "force-check" option could be provided to user, allowing to check any documents they wish.

IX. Web API

The Web API in question was made using FastAPI Python Framework. The system support client-server architecture (fig. 3).

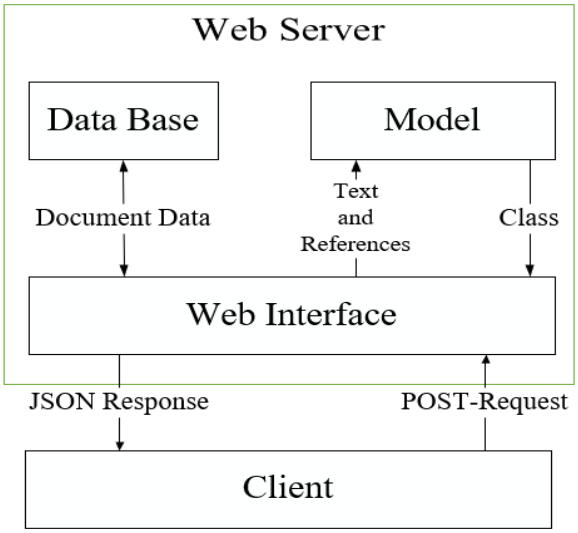

Fig. 3. Scheme of Architecture

It follows the REST pattern by saving all the necessary information to Microsoft SQL Server Database, which now contains four data tables.

The one called "Document" contains fields for its name and for date it is received.

The "Text" table encapsulates texts present in the incoming documents. Those texts could be used to retrain the model.

Along with those tables there are "Document Classifier" (which contains all the possible classes) and "Classified" (that saves the result of processing) tables.

The database is in the Third Normal Form since tables contain only complete primary key and lack non-trivial dependencies (fig. 4).

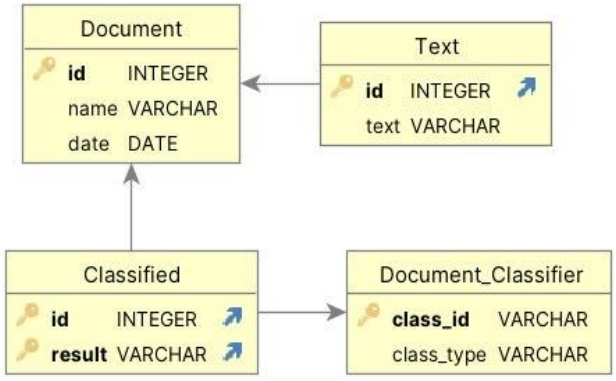

Fig. 4. Data Base Scheme

The API has three endpoints:

— /classify,
— /help
— /

The first of them is used to perform classification. It utilizes POST-request method and requires a ".docx" or ".docm" files. It returns JSON response (1):

(1) {"id": <*class id*>, "name": <*class name*>}

The other two endpoints use GET-request method and provide information about the system. Thus, "/help" method returns the regional classifier and "/" returns information about the maker and the resources on which the API is tested.

X. Future Updates

As a result, the only requirement that is not fully fulfilled is the quality of prediction. Thus, future development will involve application of the deep leaning method and use of N-Grams to the task in question.

A table named "References" would be present in the next update, it will contain all the links to other Legal Acts present in the document. It will be a "many-to-many" table linking two IDs. This table will be a groundwork of a classification approach based on references to already classified documents.

By enlarging the training base there were better results produced. Thus, there will be more documents used for classification in next updates.

References

[1] "Search Results" in *Russian Legislation.* URL: http://pravo.gov.ru/proxy/ips/?searchres=&bpas=cd00000&a3=102000505&a3type=1&a3value=%D4%E5%E4%E5%F0%E0%EB%FC%ED%FB%E9+%E7%E0%EA%EE%ED&a6=&a6type=1&a6value=&a15=&a15type=1&a15value=&a7type=1&a7from=&a7to=&a7date=&a8=&a8type=1&a1=&a0=&a16=&a16type=1&a16value=&a17=&a17type=1&a17value=&a4=102000037%3B102000038&a4type=1&a4value=&a23=&a23type=1&a23value=&textpres=&sort=7&x=65&y=10 (accessed on 06 Apr. 2023)

[2] A. Shaji, "Serving a Machine Learning Model with FastAPI and Streamlit" in *Turing.* URL: https://www.turing.com/kb/fastapi-vs-flask-a-detailed-comparison (accessed on 01 Apr. 2023)

[3] S. Li, "Multi-Class Text Classification Model Comparison and Selection" in TDS. URL: https://towardsdatascience.com/multi-class-text-classification-model-comparison-and-selection-5eb066197568 (accessed on 01 Apr. 2023)

[4] T. Joachims, "Transductive inference for text classification using support vector machines," in ICML, vol. 99, 1999, pp. 200–209.

[5] S. Tong and D. Koller, “Support vector machine active learning with applications to text classification,” Journal of machine learning research, vol. 2, no. Nov, pp. 45–66, 2001.
[6] M. Fernandez-Delgado, E. Cernadas, S. Barro, and D. Amorim, “Do we ´ need hundreds of classifiers to solve real world classification problems,” J. Mach. Learn. Res, vol. 15, no. 1, pp. 3133–3181, 2014.
[7] C. S. Leslie, E. Eskin, and W. S. Noble, “The spectrum kernel: A string kernel for svm protein classification.” in Pacific symposium on biocomputing, vol. 7, no. 7, 2002, pp. 566–575.
[8] E. Eskin, J. Weston, W. S. Noble, and C. S. Leslie, “Mismatch string kernels for svm protein classification,” in Advances in neural information processing systems, 2003, pp. 1441–1448.
[9] A. McCallum, K. Nigam et al., “A comparison of event models for naive bayes text classification,” in AAAI-98 workshop on learning for text categorization, vol. 752. Citeseer, 1998, pp. 41–48.
[10] S.-B. Kim, K.-S. Han, H.-C. Rim, and S. H. Myaeng, “Some effective techniques for naive bayes text classification,” IEEE transactions on knowledge and data engineering, vol. 18, no. 11, pp. 1457–1466, 2006

Appendix 1. Classifier

1. Constitutional system;
2. Basics of governing;
3. Civil law;
4. Family law;
5. Housing law;
6. Labor and employment;
7. Social welfare and insurance;
8. Finances;
9. Economic activities;
10. Foreign-economic affairs;
11. Natural resources and environmental control;
12. Information and Informatization;
13. Education. Science. Culture;
14. Public health. Sports. Tourism;
15. —;
16. Public order and security;
17. —;
18. Justice;
19. Public prosecutor. Legal authorities. Advocacy;
20. International affairs;