TorontoCL at the CLEF 2018 eHealth Challenge Task 1

Serena Jeblee[1], Akshay Budhkar[1], Saša Milić[1], Jeff Pinto[1], Chloé Pou-Prom[1], Krishnapriya Vishnubhotla[1], Graeme Hirst[1], and Frank Rudzicz[1,2,3]

[1] University of Toronto, Toronto Canada
{sjeblee,abudhkar,sasa,jeffpinto,chloe,vkpriya,gh}@cs.toronto.edu
[2] Toronto Rehabilitation Institute-UHN
[3] The Vector Institute
{frank}@spoclab.com

Abstract. We assign ICD-10 codes to cause-of-death phrases in multiple languages by creating rich and relevant word embedding models. We train 100-dimensional word embeddings on the training data provided, combined with language-specific Wikipedia corpora. We then use n-gram matching of the raw text to the provided ICD dictionary followed by an ensemble model which includes predictions from a CNN classifier and a GRU encoder-decoder model.

Keywords: GRU · CNN · ensemble · word embeddings · medical coding

1 Introduction

The International Classification of Diseases (ICD), established by the World Health Organization, provides a standardized and universal way to encode medical diagnoses. Used for the purposes of determining health trends and reporting statistics, the ICD assigns each medical concept, disease, or disorder to a hierarchical letter-digits combination (e.g., *A*08 denotes "viral and other specified intestinal infections") [23]. Medical coding is important for public health research and for making clinical and financial decisions. However, the coding process is often expensive and time-consuming, and can be error-prone due to its complex pipeline [11]. Automated medical coding could offer a potential solution to these problems.

Here we present our methodology and results for the task 1 of the CLEF 2018 eHealth challenge: "Multilingual information extraction — ICD10 coding" [12, 20]. This task consists of assigning ICD-10 codes [23] (the tenth revision of ICD)[4] to the text of death certificates. Datasets are provided for three languages: French, Italian, and Hungarian, and we submit results for all three.

[4] http://apps.who.int/classifications/icd10/browse/2016/en

2 Related work

The recent popularity of neural network-based methods has spread to the healthcare domain, especially convolutional neural networks (CNNs) and recurrent neural networks (RNNs), both of which we use in this paper. A variety of models have been successfully applied to health-related automated tasks, such as predicting hospital readmissions and suicide risk [19, 13] and automated diagnosis and information extraction using deep convolutional belief networks [9].

For this task, we can treat ICD coding as a sequence-to-sequence problem of words to ICD codes. Indeed, the best submission from the CLEF eHealth 2017 challenge achieved an F_1 score of 85.01% on the test data using an encoder-decoder model [21]. Other approaches from the 2017 challenge included rule-based systems (IMSUNIPD [15], LITL [4]), SVM classifiers (LIMSI [24]), and query-based algorithms (SIBM [1], WBI [22], LITL).

3 Data and pre-processing

The provided training data consists of cause-of-death texts, ICD-10 codes, and demographic information in the French, Hungarian, and Italian languages. The data is given in either *raw* or *aligned* format. *Raw* data consists of the following three files:

- A *CausesBrutes* file containing values for `DocID` (the death certificate ID), `YearCoded` (the year the death certificate was processed), `LineID` (the line number within the death certificate), `RawText` (the raw text entered by the physician in the death certificate), `IntType` (the type of time interval the patient had been suffering from coded cause), and `IntValue` (the time interval of `IntType`).
- A *CausesCalculees* file containing the ICD-10 codes. Similar to *CausesBrutes*, this file contains the `DocID`, `YearCoded`, and `LineID` fields. Additionally, this file includes values for `CauseRank` (the rank of the ICD-10 code assigned by the coder), `StandardText` (the dictionary entry or excerpt of `RawText` that supports the assigned code), and `ICD10` (the gold standard ICD-10 code).
- A *Ident* file containing demographic information. This file contains the `DocID`, `YearCoded`, and `LineID` fields, as well as the following fields: `Gender` (the gender of the deceased), `PrimCauseCode` (the code of the primary cause of death), `Age` (age at the time of death, rounded to the nearest five-year age group), and `LocationOfDeath`.

Files are provided in the *raw* format for the French, Hungarian, and Italian datasets. Additionally, the French data is also provided in an *aligned* format, consisting of one file in which the information from the *CausesBrutes*, *CausesCalculees*, and *Ident* files is already combined.

For each language, ICD-10 dictionaries are supplied. From these dictionaries, we retain the `DiagnosisText` (the text description of the given code) and `Icd1` (the ICD-10 code) fields.

For this task we discover that, with the exception of n-gram matching, model performance improves in proportion to the volume and quality of reference data. To this end, we pre-process both training and supplementary reference data to maximize the model's coverage of terms in the text by standardizing the corpora and reducing noise.

3.1 Training Data

We combine the raw format files with the demographic data to create an aligned-like file for each language, since preliminary experiments reveal that using the demographic information greatly improves classification results. We create a concatenated input stream by appending each row of `RawText` with the `YearCoded`, `Gender`, `Age`, `IntType`, `IntValue`, and `LocationOfDeath` that match the row's `DocID` and `LineID`. If a document has multiple rows, each row has identical appended data.

After initial experiments, we determine that commas (`,`) in the `RawText` field often indicate multiple ICD-10 codes. Our results improve by splitting each unaligned training row by commas prior to processing for the n-gram and the CNN models, thus assigning $n+1$ codes given a row with n commas. Removing accents from the text does not improve performance on the training data, so we keep all accents for all languages. We lowercase and strip the text from the `RawText` field of all punctuation symbols for all of our models.

3.2 Supplementary data

For our word embedding models, we use publicly available monolingual Wikipedia data from Linguatools [7]. The supplementary data is pre-processed to reduce the content to only alphabetical characters (including accented characters), since the input `RawText` does not contain significant numeric data. In addition to removing web URLs and extra whitespace, we remove any characters not in the language-specific lists in Table 1, and convert all text to lower case. After pre-processing, we have over 9M lines of Hungarian (120M tokens), 21M lines of Italian (370M tokens), and 32M lines of French (540M tokens) supplemental data.

Table 1. Allowed characters in supplementary data. We pre-process the Wikipedia data by removing web URLs, whitespace, and any characters not in these lists.

Language	Allowed characters
French	a-zA-ZÀ-ÖØ-öø-ÿ
Hungarian	a-záéúőóüöíA-ZÁÉÚŐÓÜÖÍ
Italian	a-zA-ZàèéìíîòóùúÀÈÉÌÍÎÒÓÙÚ

Table 2. Vocabulary coverage of our word embedding models on the test data, compared to models from Facebook MUSE. We report the dimensions of each word embedding, and the vocabulary coverage as quantified by the percentage of types and of tokens of the test data that can be found in the corresponding word embedding model.

Language	Embeddings	Dimensions	Types	Tokens
French	MUSE	300	33%	75%
	ours	100	**66%**	**96%**
Hungarian	MUSE	300	29%	45%
	ours	100	**82%**	**95%**
Italian	MUSE	300	65%	81%
	ours	100	**93%**	**99%**

4 Word embeddings

We experiment with pre-trained French, Hungarian, and Italian word embedding models from the Facebook MUSE dataset [2], but discover that they have fairly low vocabulary coverage on the training data. In order to get better coverage, we train our own word embeddings using Wikipedia data (Section 3.2), which we augment with the `RawText` field from the training data and the dictionaries. The training and dictionary texts are repeated $N = 4$ times, (N is chosen empirically to increase the overall accuracy). We use the `word2vec` [10] implementation in Gensim [18], with a context window size of 2 and a minimum word occurrence frequency set to 4. We set the dimensions of our word embeddings to 100, chosen for the ease of potentially aligning these vectors to other publicly available vectors trained on medical datasets.

Because we train the word embedding models on text from the provided training data, we get 100% vocabulary coverage of the training data, and very high coverage of the testing data. See Table 2 for coverage on the test set of our word embeddings models compared to pre-trained models from Facebook MUSE. We report vocabulary coverage as the percentage of types (i.e., *distinct* words) and the percentage of tokens (i.e., all words) from the `RawText` field that can be found in the given word vector model.

5 Models

We use n-gram text matching followed by a variety of neural network models. We implement the neural network models in PyTorch [16] with CUDA [14], and train them on the sequence of word embeddings representing the text of each line. We conduct 10-fold cross-validation on the training data in order to choose the final models and parameters, using `scikit-learn`'s GroupKFold function [17].

5.1 *N*-gram matching

The "first pass" of our pipeline checks whether the text string has an exact match in the dictionary. If so, we use the corresponding ICD code as our target. We experiment with spelling correction on the input text before matching it to dictionary text, since we notice that there are spelling errors in the training data. For spelling correction we use the `pyenchant` package [5]. `pyenchant` computes edit distance between a given string and vocabulary; the vocabulary we supply in this case is the words in the dictionary entries. Because spellcheck improves precision only on the French dataset, our French classifier is the only one which performs spellcheck before the exact match procedure. Interestingly, the French, Italian, and Hungarian datasets have notably different F_1 scores when this simple procedure is applied. This perhaps suggests a different model or approach for each language might be necessary for classification rather than one language-agnostic approach. Occasionally, the exact same text might map to more than one ICD code. If this happens, the most frequent ICD code is taken — frequency is computed across all training data in all languages (we assume that the distribution of ICD codes is the same across year and region).

***N*-gram matching on the Hungarian dataset** We note that the Hungarian dataset performs well with pattern matching, thus for one of our test runs, we submit results obtained with a purely rule-based, pattern-matching classifier. Specifically, for the Hungarian data set, a "second pass" is performed in which bigrams of the input text are matched to bigrams of dictionary text. Since there is a many-to-many mapping between bigrams in the training text and bigrams in the dictionary text, our chosen target ICD code is the one that matches the largest number of bigrams from the input text. Again, ties are broken by choosing the most frequent ICD code.

5.2 Encoder-decoder

Similar to work done by [21], we build an encoder-decoder model that takes as input a sequence of words and outputs a sequence of ICD codes. We experiment with one-hot vectors and various word embeddings as described in Section 4, and obtain our best results using our own word embeddings trained on the Wikipedia corpora augmented with training data and dictionary data.

The encoder and decoder architectures consist of gated recurrent units (GRUs) of hidden size 256, to which we apply a dropout of 0.1. The encoder-decoder model is trained on pairs of sequences of tokens of $\langle$`RawText`, `ICD10`$\rangle$ for each line of each death certificate (i.e., for each `LineID, DocID`). The sequence of `ICD10` codes is ordered from left-to-right by increasing rank. The pre-processed sequence of tokens from the `RawText` field is converted to a sequence of 100-dimensional word vectors, and the corresponding ICD-10 codes are converted to one-hot word vectors. We also make use of the demographic information (i.e., the `Gender, Age, LocationOfDeath, IntType, IntValue` fields) and pass it as input to the decoder as a zero-padded 100-dimensional vector. We train our

model for 10 epochs, using the AdaGrad algorithm [3] and a learning rate of 0.1, where each epoch iterates through the whole training set. The encoder-decoder model training time depends on the size of the input dataset — it takes about 4 hours for the Italian dataset (the smallest one), and up to 12 hours for the Hungarian dataset. Since we order the ICD-10 codes by rank during training, our output sequence of ICD-10 codes is also ordered by rank. For example, for an output of "T293 S299", we assign `Rank` 1 to "T293" and `Rank` 2 to "S299".

We note that unlike the n-gram matching and CNN approaches, we do not split the `RawText` on commas during training, and give as input the entire text to the encoder-decoder. Since this model treats ICD-10 prediction as a sequence-to-sequence model, it is able to "learn" the correct number of ICD-10 codes for each line of text given, as well as the correct rank order.

5.3 Convolutional neural network

We also use a convolutional neural network (CNN). Although CNNs are typically used for image processing, they have also achieved good results on text problems, including tasks in the medical domain [19].

For French and Hungarian data, we apply spelling correction to the text before passing it to the CNN. On the training data, spelling correction does not improve CNN results for Italian, so we do not correct the Italian text. All text is then pre-processed as described in Section 3.

The CNN model consists of filters of size 1 through 4 by the word embedding size (100), a max pooling layer, and a softmax prediction layer. We use Adam [6] as the optimizer, with a learning rate of 0.001, and train for 20 epochs. Compared to the encoder-decoder model, the CNN is very fast to train: 5–20 minutes versus 4–12 hours for the encoder-decoder.

One limitation of the CNN model is that it outputs exactly one ICD code per input line. In order to overcome this limitation, we use the softmax output probabilities from the last layer of the network as the input to the ensemble model, which allows us to choose multiple codes per line for the final output.

5.4 Ensemble model

We combine the outputs from the models discussed in the previous sections using a rule-based ensemble model. This model takes as inputs the CNN softmax probabilities for all the ICD codes, the codes predicted by the encoder-decoder model (including their ranks), and the codes predicted by n-gram matching.

We empirically choose thresholds to optimize the accuracy on the first cross-validation fold of every language. For each ICD code, the ensemble model determines whether they are assigned to the given `RawText` based on the following rules:

$$(\text{ICD} \in \textit{ende\_output} \textbf{ and } ICD_{ende\_rank} < t_{ende\_rank})$$
$$\textbf{or}$$
$$(ICD_{cnn\_prob} > p_{cnn})$$
$$\textbf{or}$$
$$(\text{ICD} \in \textit{ngram\_output} \textbf{ and } ICD_{cnn\_prob} > p_{ngram})$$

For every ICD code, we first check whether it is in the list of ICD-10 codes producible by the encoder-decoder *ende_output*, and then check whether its rank ICD_{ende_rank} falls *below* the threshold t_{ende_rank}. Next, the ensemble model checks whether the ICD code CNN probability ICD_{cnn_prob} is *greater* than the threshold p_{cnn}. Then, the model checks whether the ICD code is in the list of n-gram matched codes *ngram_output* and verifies whether the corresponding CNN probability ICD_{cnn_prob} is greater than the threshold p_{ngram}. Here, the CNN probabilities have two thresholds — one for the probabilities of all the ICD codes, and one only for the n-gram matched codes. Any codes that pass *one* of the checks described above are included in our prediction. The n-gram matched codes are only used for French.

The ensemble optimizes the rank threshold t_{ende_rank}, and the probability thresholds p_{cnn} and p_{ngram}. See Table 3 for the threshold values for the three languages.

Table 3. Ensemble thresholds. For each language, the ensemble model optimizes the encoder-decoder rank threshold (t_{ende_rank}), and CNN probability thresholds used to determine inclusion of codes from the CNN (p_{cnn}) and from the n-gram model (p_{ngram}).

Language	**Encoder-decoder**	**CNN**	***N*-gram**
French	14	0.55	0.25
Hungarian	14	0.75	N/A
Italian	3	0.80	N/A

6 Results

Table 4 shows the results of the final n-gram models and of the neural network models on the training set (trained and evaluated on the same data). The precision, recall, and F_1 measures are computed from the provided evaluation script on the training data.

Table 5 shows the results of the final models on the official test set. For French and Italian, we submit runs using our encoder-decoder and ensemble models. For Hungarian, we submit one run using n-gram matching only, and another run using the ensemble model.

Table 4. Results of the final models on the training data. We report the precision, recall, and F_1 scores as given by the evaluation script. Models for which we submit test results are in bold.

Language	Model	Precision	Recall	F_1
French	Exact match	0.879	0.464	0.608
	N-gram match	0.754	0.585	0.659
	Exact match + encoder-decoder	**0.8666**	**0.6508**	**0.7433**
	Exact match + ensemble	**0.8461**	**0.7857**	**0.8148**
Hungarian	Exact match	0.977	0.650	0.780
	N-gram match	**0.875**	**0.865**	**0.870**
	Exact match + encoder-decoder	0.9329	0.9022	0.9173
	Exact match + ensemble	**0.9304**	**0.9043**	**0.9172**
Italian	Exact match	0.988	0.291	0.450
	Exact match + spellcheck	0.937	0.559	0.700
	Longest n-gram match + spellcheck	0.708	0.613	0.657
	Exact match + encoder-decoder	**0.9475**	**0.8719**	**0.9081**
	Exact match + ensemble	**0.9375**	**0.8740**	**0.9046**

Table 5. Results on the test data (official evaluation). Bold indicates the better score.

Language	Model	Precision	Recall	F_1
French (*aligned*)	Exact match + encoder-decoder	0.8152	0.7118	0.7600
	Exact match + ensemble	0.8103	0.7195	**0.7622**
French (*raw*)	Exact match + encoder-decoder	0.8466	0.5151	0.6405
	Exact match + ensemble	0.8418	0.5215	**0.6440**
Hungarian	N-gram match	0.9013	0.8869	0.8940
	Exact match + ensemble	0.9221	0.8972	**0.9095**
Italian	Exact match + encoder-decoder	0.9077	0.8239	**0.8638**
	Exact match + ensemble	0.8995	0.8294	0.8630

7 Discussion

In our final submission, we use the n-gram and ensemble models for the Hungarian dataset, and the encoder-decoder and ensemble models for both the Italian and French datasets.

For Hungarian, we achieve fairly good results using a simple n-gram matching scheme. This suggests that the Hungarian ICD dictionary has better coverage of the words in the death certificate text than the dictionaries of the other two languages. The lines in the Hungarian test data have an average length of 2.9 words (2.0 standard deviation), compared to 2.1 (1.5) for Italian and 3.3 (2.5) for French.

For the encoder-decoder, CNN, and ensemble model, we use the same framework for all three languages. With training data, this model could easily be extended to new languages.

Our test results show low recall values for French, especially for the *raw* data. We suspect that the test data includes ICD codes that our models had not seen during training. A potential way to ensure that our model sees all possible ICD codes would be to include the ICD dictionaries during training. However, in our experiments, combining dictionary data with training data when training the encoder-decoder model actually produced lower results in cross-validation experiments. Augmenting data with the dictionary text skews the distribution of ICD-10 codes, which in turn, affects classification.

Our best model achieves an F_1 measure of 0.91 on the Hungarian language test data using the ensemble model. An ANOVA on the F_1 measures of our test data reveals no significant effect from either *language* ($F = 3.712$, $p = 0.212$), *model* ($F = 1.033$, $p = 0.492$), or between the effect of the two covariates ($F = 0.001$, $p = 0.982$).

We note that the amount of supplementary material is not proportional to a language's vocabulary coverage. The French Wikipedia corpus has over three times the quantity of tokens and sentences of the Hungarian, yet includes 20% fewer terms (see Table 2). This implies that the source of supplemental material has varied efficacy depending on the language. Note that spelling is not corrected in the supplemental data, which may reduce the term coverage.

8 Conclusion

We have shown that using in-domain data in addition to a large corpus of general language-specific data for word embeddings, along with neural network models, can achieve fairly high performance on ICD coding in multiple languages. This kind of model does not require any expert knowledge or feature engineering, and therefore can be implemented for any language for which we have training data.

The performance of this model could be improved in several ways. Our word embedding models are trained on a large corpus of language-specific Wikipedia data, but we would expect a better representation if the embedding models were trained on large corpora of text in the respective languages from the medical domain, such as clinical notes or medical textbooks.

A multilingual ICD classification model could also benefit from cross-lingual representations such as word embeddings aligned in the same vector space [8]. We were unable to get a good alignment for this task, but such an alignment would allow us to train models on multiple languages, and apply them to any language for which we have aligned word embeddings.

This model could also potentially benefit from character-based embeddings and RNN models, especially for agglutinative languages such as Hungarian, which have complex morphology. For such languages, more pre-processing such as morphological analysis could also help.

This challenge exposes several aspects of disease prediction that must be overcome if machine learning is to be applied globally to the electronic medical record. Not least of these aspects includes differences that occur between languages – both in terms of their own structure and semantics but also in terms

of the availability of data resources from which models are to be trained. Although we have made progress in terms of overall precision and recall, more work remains.

References

1. Cabot, C., Soualmia, L.F., Darmoni, S.J.: SIBM at CLEF eHealth Evaluation Lab 2017: Multilingual information extraction with CIM-IND. In: CLEF 2017 Evaluation Labs and Workshop: Online Working Notes, CEUR-WS (2017)
2. Conneau, A., Lample, G., Ranzato, M., Denoyer, L., Jégou, H.: Word translation without parallel data. arXiv preprint arXiv:1710.04087 (2017)
3. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research **12**(Jul), 2121–2159 (2011)
4. Ho-Dac, L.M., Fabre, C., Birski, A., Boudraa, I., Bourriot, A., Cassier, M., Delvenne, L., Garcia-Gonzalez, C., Kang, E.B., Piccinini, E., Rohrbacher, C., Séguier, A.: LITL at CLEF eHealth2017: Automatic classification of death reports. In: CLEF 2017 Evaluation Labs and Workshop: Online Working Notes, CEUR-WS (2017)
5. Kelly, R.: Python bindings for the enchant spellchecker (2017), https://github.com/rfk/pyenchant, Last accessed on 2018-05-24
6. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (dec 2014)
7. Kolb, P.: Linguatools; Wikipedia monolingual corpora (2018), http://linguatools.org/tools/corpora/wikipedia-monolingual-corpora/, Last accessed on 2018-05-24
8. Lample, G., Denoyer, L., Ranzato, M.: Unsupervised machine translation using monolingual corpora only. arXiv preprint arXiv:1711.00043 (2017)
9. Liang, Z., Zhang, G., Huang, J.X., Hu, Q.V.: Deep learning for healthcare decision making with EMRs. In: 2014 IEEE International Conference on Bioinformatics and Biomedicine (BIBM). pp. 556–559. IEEE (November 2014). https://doi.org/10.1109/BIBM.2014.6999219, http://ieeexplore.ieee.org/document/6999219/
10. Mikolov, T., Sutskever, I., Chen, K., Corado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems 26. pp. 3111–3119 (2013)
11. Névéol, A., Anderson, R.N., Cohen, K.B., Grouin, C., Lavergne, T., Rey, G., Robert, A., Rondet, C., Zweigenbaum, P.: CLEF eHealth 2017 Multilingual Information Extraction task Overview: ICD10 Coding of Death Certificates in English and French. In: CLEF 2017 Evaluation Labs and Workshop: Online Working Notes, CEUR-WS (2017)
12. Névéol, A., Robert, A., Grippo, F., Morgand, C., Orsi, C., Pelikán, L., Ramadier, L., Rey, G., Zweigenbaum, P.: CLEF eHealth 2018 Multilingual Information Extraction task Overview: ICD10 coding of death certificates in French, Hungarian and Italian. In: CLEF 2018 Evaluation Labs and Workshop: Online Working Notes, CEUR-WS (2018)
13. Nguyen, P., Tran, T., Wickramasinghe, N., Venkatesh, S.: Deepr: A convolutional net for medical records. IEEE Journal of Biomedical and Health Informatics **21**(1), 22–30 (July 2017)

14. Nickolls, J., Buck, I., Garland, M., Skadron, K.: Scalable parallel programming with CUDA. Queue **6**(2), 40–53 (2008). https://doi.org/10.1145/1365490.1365500, http://doi.acm.org/10.1145/1365490.1365500
15. Nunzio, G.M.D., Beghini, F., Vezzani, F., Henrot, G.: A lexicon based approach to classification of ICD10 codes. IMS Unipd at CLEF eHealth Task 1. In: CLEF 2017 Evaluation Labs and Workshop: Online Working Notes, CEUR-WS (2017)
16. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch. In: Autodiff Workshop at Neural Information Processing Systems (NIPS) 2017 (2017)
17. 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, É.: Journal of Machine Learning Research, vol. 12. MIT Press (2001), https://dl.acm.org/citation.cfm?id=2078195
18. Řehůřek, R., Sojka, P.: Software framework for topic modelling with large corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. pp. 45–50. ELRA, Valletta, Malta (May 2010), http://is.muni.cz/publication/884893/en
19. Shickel, B., Tighe, P., Bihorac, A., Rashidi, P.: Deep EHR: A survey of recent advances in deep learning techniques for electronic health record (EHR) analysis (June 2017). https://doi.org/10.1109/JBHI.2017.2767063
20. Suominen, H., Kelly, L., Goeuriot, L., Kanoulas, E., Azzopardi, L., Spijker, R., Li, D., Névéol, A., Ramadier, L., Robert, A., Palotti, J., Jimmy, Zuccon, G.: Overview of the CLEF eHealth Evaluation Lab 2018. In: CLEF 2018 - 8th Conference and Labs of the Evaluation Forum, Lecture Notes in Computer Science (LNCS). Springer (2018)
21. Tutubalina, E., Miftahutdinov, Z.: An Encoder-Decoder Model for ICD-10 Coding of Death Certificates. In: Machine Learning for Health Workshop at Neural Information Processing Systems (NIPS) 2017 (dec 2017), http://arxiv.org/abs/1712.01213
22. Ševa, J., Kittner, M., Roller, R., Leser, U.: Multi-lingual ICD-10 coding using a hybrid rule-based and supervised classification approach at CLEF eHealth 2017. In: CLEF 2017 Evaluation Labs and Workshop: Online Working Notes, CEUR-WS (2017)
23. World Health Organization: International statistical classifications of diseases and related health problems. 10th rev, vol. 1. World Health Organization, Geneva, Switzerland (2008)
24. Zweigenbaum, P., Lavergne, T.: Multiple methods for multi-class, multi-label ICD-10 coding of multi-granularity, multilingual death certificates. In: CLEF 2017 Evaluation Labs and Workshop: Online Working Notes, CEUR-WS (2017)