A study of Machine Learning models for Clinical Coding of Medical Reports at CodiEsp 2020

Marco Polignano, Vincenzo Suriano, Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro

University of Bari Aldo Moro, Dept. Computer Science, Bari , Italy.
marco.polignano@uniba.it, v.suriano10@studenti.uniba.it, pasquale.lops@uniba.it, marco.degemmis@uniba.it, giovanni.semeraro@uniba.it

Abstract. The task of identifying one or more diseases associated with a patient's clinical condition is often very complex, even for doctors and specialists. This process is usually time-consuming and has to take into account different aspects of what has occurred, including symptoms elicited and previous healthcare situations. The medical diagnosis is often provided to patients in the form of written paper without any correlation with a national or international standard. Even if the WHO (World Health Organization) released the ICD10 international glossary of diseases, almost no doctor has enough time to manually associate the patient's clinical history with international codes. The CodiEsp task at CLEF 2020 addressed this issue by proposing the development of an automatic system to deal with this task. Our solution investigated different machine learning strategies in order to identify an approach to face that challenge. The main outcomes of the experiments showed that a strategy based on BERT for pre-filtering and one based on BiLSTM-CNN-SelfAttention for classification provide valuable results. We carried out several experiments on a subset of the training set for tuning the final model submitted to the challenge. In particular, we analyzed the impact of the algorithm, the input encoding strategy, and the thresholds for multi-label classification. A set of experiments has been carried out also during a post hoc analysis. The experiments confirmed that the strategy submitted to the CodiEsp task is the best performing one among those evaluated, and it allowed us to obtain a final mean average error value on the test set equal to 0.202. To support future developments of the proposed approach and the replicability of the experiments we decided to make the source code publicly accessible.

Keywords: BERT, CNN, BiLSTM, Self Attention, Machine Learning, ICD-10, Medical Diagnosis, Deep Learning, Clinical Coding

1 Introduction

Clinical coding [30] is the task of associating unique identification codes with a clinical diagnosis, or sometimes with a portion of it. Doctors and specialists associate the diagnosis given to the patients with the corresponding international classification only in rare cases. One of the most widely adopted standards is ICD10 [22], the tenth version of the international medical glossary released by WHO (World Health Organization). Although this annotation task may not seem very useful for medical purposes, it is extremely relevant for statistical purposes, automatic diagnosis analysis of clinical records, and data interoperability across healthcare systems in different countries. Indeed, if each diagnosis is fully digitized with a worldwide standard, every doctor in the world who is visiting us could uniquely interpret our medical records and provide us with the appropriate treatment. In addition, diagnostic patterns used by clinicians could be identified to improve automatic disease prediction strategies and provide automatic specialist support for decision making. These observations strongly support the need for automated systems to support clinicians to perform this task quickly and without human intervention. From the technical point of view, this task is very challenging because it requires the development of an artificial intelligence system able to not only assign more than one class label to the medical response choosing from a very high number of choices, but also to identify the fragment of text associated to that choice. The CodiEsp task at CLEF 2020 [30, 13, 21] tries to face this problem by releasing a corpus of 1,000 clinical case studies manually selected by practicing physicians and clinical documentalists. Using that dataset, we carried out an experimental study by testing several machine learning approaches, and we submitted the best performing one to the competition. In the following, we first analyze the state of the art concerning the reference topic (Sect. 2), then we provide the details of the proposed models (Sect. 3). In Sect. 4, we thoroughly present the performed experiments, and we finally present the main outcomes and possible future work.

2 Related Work

The research community has addressed clinical coding tasks for a long time, and numerous scientific contributions have been proposed about the topic. Indeed, CLEF (Conference and Labs of the Evaluation Forum) conference has been working on eHealth and Information Extraction since 2013, but the oldest corpus on the subject dates back to 1973 [30]. Chapman et al. [9], already in 1999, stated that a computer algorithm could solve the clinical coding task better than a human being. This assertion is intuitive because even for an expert in the field, it can be very complex to assign a specific code to the result of a medical diagnosis choosing it from more than 70,000 currently available ICD-10 codes. However, in 2006, Kukafka et al. [18], confirmed that when the identification of the right code is not obvious, also Natural Language Processing (NLP) tools could easily lack accuracy. Today, NLP and machine learning techniques are widespread and

are receiving substantial attention from research communities. Among the best performing systems, the one proposed by Miftahutdinov [19] at CLEF eHealth 2017 uses an LSTM on a TF-IDF representation of the text to identify the most suitable ICD-10 code for the input sequence. This allows to obtain an F1 score equal to 0.85, considering a classification on 1,256 distinct classes. During the same competition, Cabot et al. [7] used an NLP pipeline to obtain the highest F1 score of the competition on the data provided in French (i.e., 0.764). Several preprocessing steps were performed, including stop words filtering, then a method based on the Double Metaphone phonetic encoding algorithm was used to operate a first approximate term search. Finally, a Weighted Distance Score algorithm has been developed to rank the list of candidate terms. The most likely term having the highest score is retained as the matching ICD-10 code for the phrase. In 2018, Atutxa et al. [3], proposed a three-level sequence-to-sequence neural network-based approach. The first neural network tries to assign one set of ICD-10 codes to the whole document, then they are refined to assign one set of codes to the line, and finally one specific code. This strategy allowed the model to obtain an F1 score between 0.7086 and 0.9610, depending on the language of the dataset on which the system has been evaluated. Almagro et al. [1] proposed a supervised learning system based on a multilayer perceptron, SVMs, and a One-vs-Rest strategy. The approach allows to train a binary model for each of the target ICD-10 codes, indicating the presence or absence of the code. The model was able to obtain an F1 score of 0.910. At CLEF eHealth 2019, the best system was proposed by Sänger et al. [28], obtaining an F1 score of 0.80. The proposed model utilized a multilingual BERT [10] text encoding model, fine-tuned on additional training data of German clinical trials also annotated with ICD-10 codes. The model is extended by a single output layer to produce probabilities for specific ICD-10 codes. Amin et al. [2] participated in the same task obtaining the second place. They evaluated various approaches, such as Convolutional Neural Networks (CNN) and Attention models, among others. They obtained the best results when relying on Bidirectional Encoder Representations from Transformers (BERT) and, more specifically, on BioBERT, which was trained on biomedical documents. Considering the successful models presented as state of the art, we decided to use a machine learning approach that combines CNNs, Bidirectional LSTMs, Attention Layers, and BERT. Details of the proposed architecture are provided in Sect. 3.

3 Resources and Model Architectures

3.1 CodiEsp 2020 corpora

The CodiEsp track at CLEF 2020 [30, 13, 21] contains three sub-tracks (2 main and 1 exploratory) about analysis of clinical reports:

- CodiEsp Diagnosis Coding main sub-task (CodiEsp-D): it requires automatic ICD10-CM [CIE10 Diagnóstico] code assignment. This sub-track evaluates systems that predict ICD10-CM codes (in the Spanish translation, CIE10-Diagnóstico codes).

Table 1. Statistics on the training datasets.

	ICD10-CM	ICD10-PCS
Number of code sections	21	16
Total number of possible codes	71,486	72,081
Unique codes used in the training set	1,788	546
Number of total annotations in the training set	8,494	2,587

- CodiEsp Procedure Coding main sub-task (CodiEsp-P): it requires automatic ICD10-PCS [CIE10 Procedimiento] code assignment. This sub-track evaluates systems that predict ICD10-PCS codes (in the Spanish translation, CIE10-Procedimiento codes).
- CodiEsp Explainable AI exploratory sub-task (CodiEsp-X). Systems are required to submit the reference to the predicted codes (both ICD10-CM and ICD10-PCS). The correctness of the provided reference is assessed in this sub-track, in addition to the code prediction.

For each task a dataset for training, development and test has been released. Generally speaking, the CodiEsp corpora contain manually annotated clinical reports with corresponding clinical codes. The clinical reports are written in Spanish, and they are annotated with the CIE10 glossary (the Spanish version of ICD10-CM and ICD10-PCS). The training set contains 500 clinical cases, while the development and the test set provide 250 clinical cases each. The CodiEsp corpus format is plain text with UTF8 encoding, where each clinical case is stored in a single file whose name is the clinical unique case identifier. The final collection of the 1,000 clinical cases of the corpus contains 16,504 sentences, with 16.5 sentences per clinical case on average. It contains 396,988 words, with 396.2 words per clinical report on average. For sub-task 1 and 2 of the CodiEsp task, (CodiEspD and CodiEsp-P), the training files contain the following fields: [articleID, label, ICD10-code, text-reference].

- **ArticleID**: it contains the identifier of the clinical text that corresponds to the name of the file.
- **Label**: it contains the diagnostic or procedimiento code.
- **ICD10-code**: it contains the ICD10 code.
- **Text-reference**: it contains the word or phrase in the clinical text.

In Fig. 1 and 2, it is possible to observe how annotations provided with the training data of the two tasks are differently distributed between the different sections of the two ICD10 vocabularies. It is immediately clear that the distribution is not uniform, and some classes are more represented than others. For example, for the training set of task CodiEspD, class 18 is the most represented one, with 2,214 annotations, while class 16 is the least represented with only 23 examples. In Table 1, it is possible to observe that considering all the possible codes of ICD10, the training dataset covers only 1,788 unique codes for

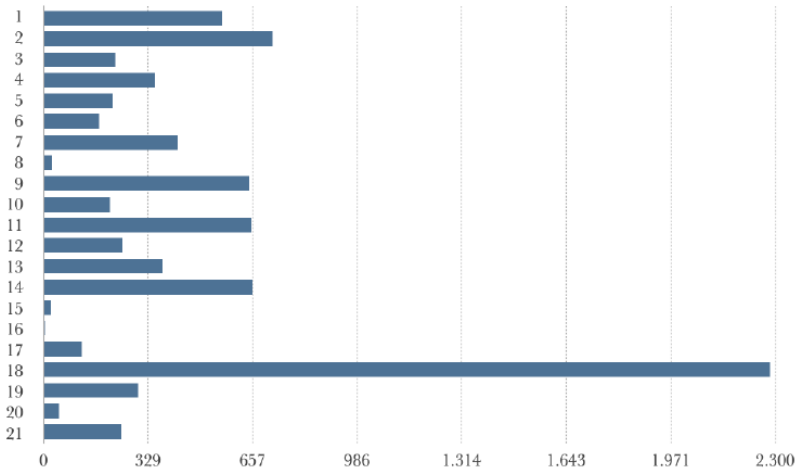

Fig. 1. Distribution of annotations in the training dataset among the ICD10-CM sections.

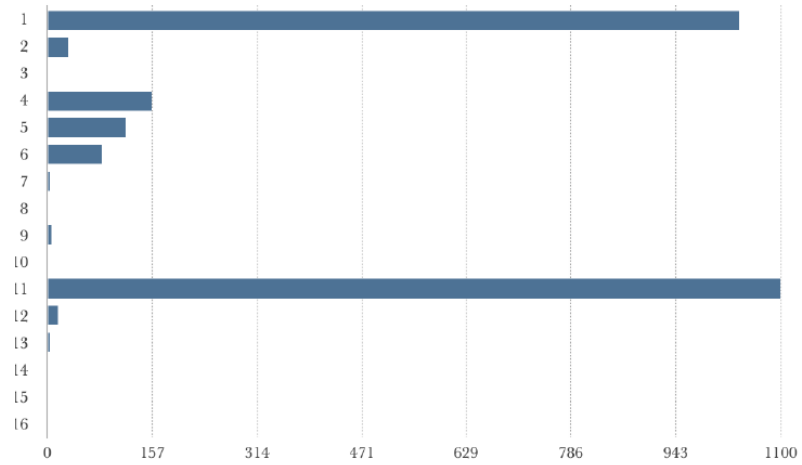

Fig. 2. Distribution of annotations in the training dataset among the ICD10-PCS sections.

CodiEspD and 546 codes for CodiEspP. Based on this observation, we decided to use models that only provide codes with at least one example in the training set.

The organizers of the CodiEsp task also released an additional resource that extends the previous datasets. It contains the description of the codes in the

ICD10 vocabulary, both "Diagnosis" and "Procedures" (Fig. 3). This resource contains two files:

- "codiesp-D_codes.tsv": it contains all the 98,288 ICD10-CM codes, along with their description in Spanish and English.
- "codiesp-P_codes.tsv": it contains all the 87,170 ICD10-PCS codes, along with the their description in Spanish and English. It contains the codes up to the fourth nesting lever its of hierarchy.

16075	Derivación de ventrículo cerebral a intestino, con sustituto de tejido autólogo, abordaje abierto	Bypass Cerebral Ventricle to Intestine with Autologous Tissue Substitute, Open Approach
16076	Derivación de ventrículo cerebral a cavidad peritoneal, con sustituto de tejido autólogo, abordaje abierto	Bypass Cerebral Ventricle to Peritoneal Cavity with Autologous Tissue Substitute, Open Approach
16077	Derivación de ventrículo cerebral a tracto urinario, con sustituto de tejido autólogo, abordaje abierto	Bypass Cerebral Ventricle to Urinary Tract with Autologous Tissue Substitute, Open Approach
16078	Derivación de ventrículo cerebral a médula ósea, con sustituto de tejido autólogo, abordaje abierto	Bypass Cerebral Ventricle to Bone Marrow with Autologous Tissue Substitute, Open Approach
001607B	Derivación de ventrículo cerebral a cisternas cerebrales, con sustituto de tejido autólogo, abordaje abierto	Bypass Cerebral Ventricle to Cerebral Cisterns with Autologous Tissue Substitute, Open Approach
00160J0	Derivación de ventrículo cerebral a nasofaringe, con sustituto sintético, abordaje abierto	Bypass Cerebral Ventricle to Nasopharynx with Synthetic Substitute, Open Approach
00160J1	Derivación de ventrículo cerebral a seno mastoideo, con sustituto sintético, abordaje abierto	Bypass Cerebral Ventricle to Mastoid Sinus with Synthetic Substitute, Open Approach
00160J2	Derivación de ventrículo cerebral a aurícula, con sustituto sintético, abordaje abierto	Bypass Cerebral Ventricle to Atrium with Synthetic Substitute, Open Approach
00160J3	Derivación de ventrículo cerebral a vaso sanguíneo, con sustituto sintético, abordaje abierto	Bypass Cerebral Ventricle to Blood Vessel with Synthetic Substitute, Open Approach

Fig. 3. Example of code description provided as addition task resource.

3.2 Classification model proposed to the CodiEsp task

The Clinical Coding task has been approached using different machine learning strategies that are commonly used to deal with the classification task in the field of Natural Language Processing (NLP). In particular we focused on the use of deep learning techniques such as LSTM [15], CNN [17], Attention Layers [32] and Bidirectional Encoder Representations from Transformers (BERT) [10].

Long-short term memory model. The neural network model based on long-short term memory (LSTM) was proposed in 1997 [15]. Since then, it has been widely used with data that have an inherent sequential structure, such as text. LSTMs are part of the family of sequential models based on recurring neurons [12]. In particular, an architecture of this type is based on the idea that the state of the specific neuron depends on that of the previous $t-1$ state. The natural evolution of a model based on recurring neurons introduces a memory. This structure allows the recurring neuron to depend not only on the state of the single one at step $t-1$, but also on the state of the different neurons at step $t-n$. This idea is the base of architectures such as RNN, LSTM, and GRU. Among them, LSTM has the peculiarity of having also a forget gate able to manage the amount of information to be kept in memory. At each step a portion of the memory is deleted and another one is added. These features allow the model to be state-of-the-art for many NLP applications, such as machine translation [35], automatic summarization [29], parsing, and sentiment analysis [33, 5]. In our architecture we used LSTM in its bidirectional variant.

Convolutional neural network. Convolutional neural networks (CNN) are born from an accurate study of how the portion of the brain cortex works for vision [16]. This has promoted their wide use in the computer vision task for image recognition since 1980 [11]. Recently, they are also widely used in text analysis tasks, thanks to the fast increase in computational power available to everyone. A convolutional neuron is able to concentrate only on a portion of the input data [12], e.g., a set of pixels in an image. A layer full of neurons using the same filter (or convolution kernels) gives a feature map, which highlights the areas in an input that are most similar to the filter. During the training, a CNN finds the most useful filters for its task, and learns to combine them into more complex patterns. A CNN is usually followed by a Pooling Layer. Its goal is to subsample the input image in order to reduce the computational load, the memory usage, and the number of parameters. A pooling layer typically works on every input channel independently, so the output depth is the same as the input depth. A neural model using CNN usually alternating CNN layers with Pooling layers with a dense final layer aimed at prediction (classification or regression).

Self attention. Similarly to the attention strategy proposed in [4], self-attention, also known as intra-attention, provides the model ability to weigh the vectors of single words of the sentence differently, according to the similarity of the neighboring tokens. It is possible to say that the level of attention can provide us an idea of what features the network is looking at most during learning and subsequent classification. In particular, we consider an additive context-aware self-attention equal to the whole set of words in input (Eq. 1) [34].

$$\begin{aligned} h_{t,t'} &= tanh(x_t^T W_t + x_{t'}^T W_{t'} + b_t) \\ e_{t,t'} &= \sigma(W_e h_{t,t'} + b_e) \\ a_{t,t'} &= softmax(e_{t,t'}) \\ l_t &= \sum_{t'=1}^{n} a_{t,t'} x_{t'} \end{aligned} \tag{1}$$

where, σ is the element-wise sigmoid function, W_t and W'_t are the weight matrices corresponding to the hidden states h_t and $h_{t'}$'; W_e is the weight matrix corresponding to their non-linear combination; b_t and b_e are the bias vectors. The attention-focused hidden state representation l_t of a token at timestamp t is given by the weighted summation of the hidden state representation $h_{t'}$ of all other tokens at timesteps t. We use the last self-attention implementation for Keras [1] .

BERT. The Bidirectional Encoder Representations from Transformers (BERT) [10] is a deep learning model based on the Transformer concept [32]. In particular, a Transformer architecture can be considered as a stack of N input encoding modules and M decoding modules to obtain an output using multi-head attention and feed-forward layers. The encoder and decoder blocks are identical in

[1] https://github.com/CyberZHG/keras-self-attention

numbers, and they are stacked on top of each other. The idea behind a Transformer architecture is to formalize the dependencies between input and output without the use of recurring neural networks. BERT uses a modified version of this architecture in which only encoding layers are present. In particular, the basic version of BERT uses 12 layers, the full version 24. BERT uses two different training strategies "masking" and "next sentence prediction". The first strategy trains the model to recognize certain words in the input sentence that have been appropriately hidden. Usually, the amount of hidden elements is 20% of the words in the sentence. The second mode is to guess the sentence that follows the input sentence. During training, 50% of the inputs are a pair in which the second sentence is the subsequent sentence in the original document, while in the other 50% a random sentence from the corpus is chosen as the second sentence. This training strategy makes BERT an extremely reliable model that can be very accurate in formalizing semantics among words considering their context. As a result, the pre-trained BERT model can be refined with only one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without significant changes to the task-specific architecture. There are currently several versions of BERT, also trained on data in languages other than English, such as BETO [8] for Spanish and AlBERTo [26] for Italian.

The machine learning model we proposed for the CodiEsp challenge uses all the previously described architectures. Specifically, we decided to use a BERT-based classifier to perform a pre-filtering operation in order to select a subset of sentences possibly referring to a clinical state. Later, the candidate sentences are submitted to a classifier based on BiLSTM, CNN, and self-attention to assign them one or more clinical codes. The architecture of the final model proposed for the clinical coding task is shown in Fig. 4. It is worth to note that using the proposed strategy it was not possible to participate in task 3 of the competition (CodiEsp-X), which requires the identification of the fragment of text referring to the code. Indeed, we do not focus on the portion of text that specifically refers to a disease, while we used a classification model working with the whole sentence.

Focusing more on the proposed model in Fig. 4, we observe that the input text is provided to a BERT model. The goal is the data pre-filtering, in order to select generally speaking sentences, from those talking about a symptom, disease, or treatment. This is a mandatory step because BERT accepts as input only pieces of text not exceeding 128 characters. For this reason, we work on the task at the sentence level, splitting the original clinical report into many sentences that could be or not associated with one or more codes. As pre-trained BERT model, we decided to use BETO [8], a Spanish pre-trained version of BERT. The authors trained BETO using 12 self-attention layers with 16 attention-heads each and 1,024 as hidden size. They used all the data from Wikipedia and all of the sources of the OPUS Project [31], having the text in Spanish. This source includes the United Nations and Government journals, TED Talks,

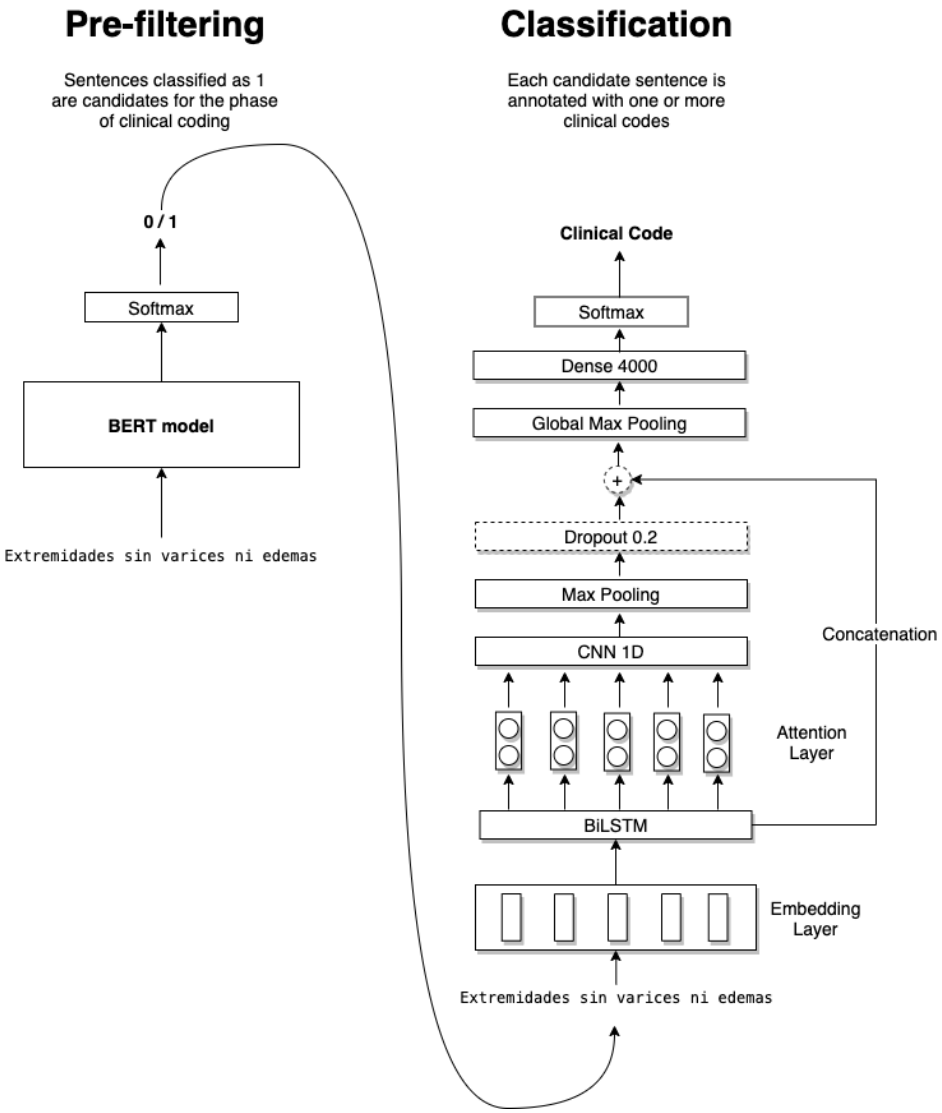

Fig. 4. Architecture of the model used as final submission at the CodiEsp challenge.

Subtitles, News Stories, and more. The total size of the corpora gathered was comparable with the corpora used in the original BERT. We decided not to use the multilingual version of BERT because it has been shown that a version trained on the native language performs much better in many NLP tasks [27]. The sentences classified as possible references of clinical codes are consequently passed to the second part of our model. First of all, the sentences are encoded into

word embeddings. In this step, we decided to use a FastText embedding strategy [6], which proved to be more effective than GLoVE [24] and Word2Vec [20] when many domain-specific words occur in the dataset [25]. For our final configuration of the model, we chose the one released by José Cañete[2] made of 300 dimensions, trained on the Spanish Unannotated Corpora[3] containing more than 3 billion words. We have configured the LSTM network to work with its bidirectional version. We set the value of hidden units to 64 and the internal dropout value to 0.3. This choice was motivated by the need to reduce the dimensionality of the network output, in order to make the operations to be carried out by the following layers not computationally expensive. Moreover, the dropout value was used to reduce, during the learning, the effect of the overfitting on the training data. We have also decided to vary the function of activation used by the net, setting it to the hyperbolic tangent function (tanh). This activation function has an S-Shape and produces values in the -1 and 1 range, making the layer output more centered to the 0. Moreover, it produces a gradient larger than the sigmoid function, helping to speed up the convergence. A level of self-attention is added following the LSTM. We applied the CNN layer on the result of the attention algorithm. Such hidden level has a matrix form due to the vectorial representation supplied by the word embeddings on the tokens in the input. In detail, it has the form of 128x64, which allows us to apply a 1D Convolutional network with 64 filters and 5x5 kernel. We used ReLu as activation function, that unlike the hyperbolic tangent is faster to calculate. On the top of the CNN layer, we added a Max Pooling function for subsampling the values obtained, reducing the computational load and, the number of parameters of the model. In particular, we used a small 2x2 kernel. On the output of the last max-pooling layer, we applied a dropout function. The hidden model obtained until this step has been merged with the output of the previous Bi-LSTM. We apply this operation for letting the model conceptualize both local and long-term features better. After that, we used a max-pooling layer for 'flattening' the results and reduce the model parameters. An analog function of dimensionality reduction is performed by the consequent dense layer and the following dropping function. Finally, another dense layer with a soft-max activation function has been applied for estimating the probability distribution of each clinical code available in the dataset. The source code of the model is publicly available on GitHub[4].

4 Experimental session

The final architecture of the model, proposed in Section 3.2, was obtained after conducting several experiments on 20% of the training dataset released for the **Codiesp-D subtask**. In order to always select the same portion of the dataset, we randomly selected sentences using the value 42 as seed for our random func-

[2] https://github.com/dccuchile/spanish-word-embeddings
[3] http://crscardellino.github.io/SBWCE/
[4] https://github.com/marcopoli/CODIESP-10

tion. During our experimental session, we raised five different experimental questions:

- **RQ1**: Do recent deep learning models, such as LSTMs, outperform classical machine learning approaches?
- **RQ2**: Which is the best strategy for encoding text in the form of word embedding?
- **RQ3**: Which is the best model among those we proposed that allows us to achieve the best performance?
- **RQ4**: Can the use of a sentence pre-filtering classifier help to improve the performance of the model?
- **RQ5**: Is there a class probability threshold that allows us to choose more than one code as a result of the classification?

In order to validate our claims we repeated the experiments also on the annotated test set released by the task organizers after the challenge.

4.1 Metrics and Settings

We trained different classification models in order to understand the best approach to solve the CodiEsp challenge. In particular we developed the following models:

- LSTM, BiLSTM
- CNN, CNN + Self Attention
- BERT
- BiLSTM + CNN, BiLSTM + CNN + Self Attention
- (Pre-filtering) BERT - (Classification) BiLSTM + CNN + Self Attention
- (Pre-filtering) BiLSTM + CNN + Self Attention - (Classification) BERT
- (Pre-filtering) BERT - (Classification) BERT
- (Pre-filtering) BiLSTM + CNN + Self Attention - (Classification) BiLSTM + CNN + Self Attention

For the selection of the word embedding strategy to use for encoding the textual sentences, we have evaluated the following resources:

- FastText Spanish official release [14]
- Fastext Spanish Unannotated Corpora (SUC) by José Cañete [5]
- GloVe Spanish Billion Word Corpus (SBWC) by George Pérez [5]
- Word2Vec Spanish Billion Word Corpus (SBWC) by Cristian Cardellino [5]

Finally, as baselines we chose classic machine learning approaches used for dealing with a classification problem. In particular we implemented them in Python using the scikit learn library [23]:

- Logistic Regression, C=0.1

[5] https://github.com/dccuchile/spanish-word-embeddings

Table 2. Results obtained running the baselines on the 20% of the training set.

	Marco-P	**Macro-R**	**Macro-F1**
Logistic Regression	0.03367	0.03412	0.03071
SVC - rbf	0.00106	0.00128	0.00116
Decision Tree Classifier	0.00729	0.00868	0.00771
Random Forest Classifier	0.00531	0.00272	0.00275
Ada Boost Classifier	0.00106	0.00127	0.00116
LSTM - no pre-trained word embeddings	**0.08473**	**0.09060**	**0.07923**

Table 3. Results obtained varying the pre-trained word embeddings on the 20% of the training set.

	Macro-P	**Macro-R**	**Macro-F1**
LSTM - FastText Spanish official	0.09299	0.09629	0.08589
LSTM - FastText SUC	**0.09357**	**0.09903**	**0.08845**
LSTM - GloVe SBWC	0.09252	0.09722	0.08715
LSTM - Word2Vec SBWC	0.09202	0.09672	0.08685

- SVC (SVM Classifier) with RBF kernel, C=0.1
- Decision Tree Classifier
- Random Forests Classifier, n_estimators= 500
- ADA Boost, n_estimators= 100, learning_rate= 0.01

The model performance has been evaluated using the standard metrics of precision, recall, F1 in their macro-average version and on test set also the Mean Average Precision (MAP) [12]. We trained all the models for 30 epochs with a fixed random value of 42 a batch size of 256 when needed.

4.2 Discussion of results

Looking at results in Tab. 2, it is worth to note that the strategy based on deep learning, i.e. the LSTM model, outperforms those based on classical machine learning approaches. Specifically, the logistic regression is the best model among the "classic" ones, with a F1 score equal to 0.03071. The LSTM strategy here proposed, based on a layer of word embeddings trained only data provided as input, i.e. no pretrained weights are used, achieves a F1 score performance which is higher than twice that of logistic regression, i.e. 0.07923. This result allows us to provide a positive answer to **RQ1**.

Tab. 3 reports the results obtained by evaluating different pre-trained word embedding weights. As expected, the differences in the final F1 outcome are

Table 4. Results obtained holding the FastText SUC word embedding, varying the model, using an evaluation on the 20% of the training set.

	Macro-P	**Macro-R**	**Macro-F1**
LSTM	0.09357	0.09903	0.08845
BiLSTM	0.10354	0.10909	0.09789
BiLSTM + CNN	0.09995	0.11552	0.09831
BiLSTM + CNN + SelfAtt.	**0.10629**	**0.11887**	**0.10410**
CNN	0.09511	0.10095	0.09100
CNN + SelfAtt.	0.09279	0.09484	0.08706
BERT (BETO)	0.10381	0.10821	0.10294

Table 5. Results obtained holding the FastText SUC word embedding, the two models (BERT, BiLSTM + CNN + SelfAttention) and varying their combinations for pre-filtering and classification, using an evaluation on the 20% of the training set.

	Macro-P	**Macro-R**	**Macro-F1**
(Pre-filtering) BiLSTM + CNN + SelfAtt. — (Classification) BiLSTM + CNN + SelfAtt.	0.13241	0.10934	0.11534
(Pre-filtering) BiLSTM + CNN + SelfAtt. — (Classification) BERT (BETO multi-class)	0.09180	0.08871	0.10022
(Pre-filtering) BERT (BETO) — (Classification) BERT (BETO multi-class)	0.09704	0.10092	0.11734
(Pre-filtering) **BERT (BETO)** — **(Classification)** **BiLSTM + CNN + SelfAtt.**	**0.13823**	**0.12053**	**0.13632**

not significant, but the best score is obtained using the FastText approach pre-trained on the Spanish Unannotated Corpora (SUC). We decided not to evaluate word embeddings weights released for English, because we would like to focus on the portion of data released in Spanish rather than its translated version. In this new experiment, we increased the F1 score previously claimed, reaching a value of 0.10410. These results allow us to answer properly to **RQ2**.

In Tab. 4, we reported results obtained by varying the architecture of our model, by holding the pre-trained word embeddings choice at the previous experimental step. An unexpected result is obtained by observing the strategy based

Table 6. Results obtained keeping fixed the FastText SUC word embedding, the two models (BERT, BiLSTM + CNN + SelfAttention) and varying the threshold on the probability returned by the architecture for each class. We use it for selecting multiple labels for the specific sentence. Also in this case the evaluation has been performed on the 20% of the training set.

BERT (BETO) — **BiLSTM + CNN + SelfAtt.**	**Macro-P**	**Macro-R**	**Macro-F1**
— Threshold: 0.05	0.09271	0.27911	0.12479
— **Threshold: 0.10**	**0.14951**	**0.24802**	**0.16011**
— **Threshold: 0.25**	**0.22644**	**0.18310**	**0.16188**
— Threshold: 0.50	0.27531	0.09811	0.13210
— Threshold: maxProb - (maxProb*0.10)	0.15545	0.13324	0.13584
— Threshold: maxProb - (maxProb*0.25)	0.12746	0.13943	0.12802
— Threshold maxProb-0.10	0.00434	0.29032	0.01028

on BERT as a simple classifier of codes. It performs quite worst than the one that uses BiLSTM, CNN, and self-attention layers. The difference between the two approaches is tiny, and, from our point of view, it is not very relevant. For this reason, we decided to go over using both the approaches. In this step, we obtained the best F1 score of 0.10410, i.e., around 0.02 points greater than the previous result. The results allow us to provide a valid answer to **RQ3**.

The following evaluation step is about splitting the clinical coding task into two independent steps: pre-filtering and classification. We reported the results obtained by this evaluation step in Tab. 5. It is possible to note that, among the different combinations of classifiers used for the two steps, the best configuration is that using BERT as a pre-filtering strategy and BiLSTM + CNN + Self Attention as a classification approach. We were able to increase F1 score by around 0.03 points from the previous step, reaching the value of 0.13632. The behavior we observed in results allows us to answer at **RQ4** positively.

The last evaluation concerns the strategy for selecting many labels for a single sentence. We decided to use a threshold on the result of the softmax function that allowed us to extract the label on which the model is more certain. We decided to vary the thresholds using both fixed and dynamic ones. The results are reported in Tab. 6. It is possible to observe that both the values 0.10 and 0.25 achieve good results. In particular, using a threshold of 0.10, we are maximizing the recall, on the contrary, we observe high values of precision. Due to the consideration that in a real scenario, a tool like this can be a decision support system for the doctors, we decided to use the configuration that uses 0.10 as the final model for the CodiEsp task. The results support our positive answer also for **RQ5**.

The model we implemented, has been used for participating at both CodiEsp subtasks, i.e., CodiEsp-D and CodiEsp-P.

Table 7. Results obtained on the annotated test set of the CodiEsp-D subtask.

BERT — BiLSTM + CNN + SelfAtt.	**MAP**	**Macro-P**	**Macro-R**	**Macro-F1**
Threshold 0.05	0.194	0.12913	0.25016	0.16804
Threshold 0.10	**0.202**	**0.19238**	**0.21361**	**0.19931**
Threshold 0.25	**0.181**	**0.28702**	**0.19935**	**0.2201**
Threshold 0.50	0.15	0.38739	0.11682	0.1765
Threshold 0.75	0.118	0.425	0.08365	0.13742
Threshold maxProb-0.10	0.156	0.00599	0.31974	0.01172
Threshold maxPrb-0.25	0.142	0.00394	0.50282	0.07813
Threshold maxProb - (maxProb*0.10)	0.165	0.21341	0.15787	0.17897
Threshold maxProb - (maxProb*0.25)	0.168	0.19305	0.16649	0.17636
Threshold: MeanValue	0.096	0.00836	0.48201	0.0134
Threshold: MeanValue - MeanValue*0.25	0.093	0.00735	0.49042	0.01187
Threshold: MeanValue - MeanValue*0.35	0.0901	0.00698	0.49313	0.01118
Threshold: MeanValue - MeanValue*0.50	0.089	0.00643	0.49681	0.01022

Table 8. Official CodiEsp task results.

CodiEsp-D							
Run	**MAP**	**MAP _codes**	**MAP30**	**MAP30 _codes**	**P**	**R**	**F1**
BERT (BETO) — BiLSTM + CNN + SelfAtt.	**0.202**	**0.236**	**0.202**	**0.236**	**0.295**	**0.323**	**0.308**
BERT (BETO)	0.117	0.136	0.117	0.136	0.272	0.218	0.242
BiLSTM+CNN+SelfAtt.	0.169	0.195	0.16	0.185	0.135	0.442	0.207
CosiEsp-P							
BERT — BiLSTM + CNN + SelfAtt.	**0.221**	**0.25**	**0.219**	**0.247**	**0.186**	**0.38**	**0.25**
BiLSTM	0.137	0.152	0.127	0.141	0.122	0.399	0.187
BERT	0.141	0.154	0.14	0.153	0.155	0.323	0.209
BiLSTM+CNN+SelfAtt.	0.17	0.191	0.15	0.168	0.097	0.513	0.164

4.3 Post Hoc Analysis

We performed a further investigation of the performance of our model on the gold annotated test set. In particular, in this phase, we take into account the score obtained for the MAP metric, because it is used by the organizers for calculating the final leaderboard. As we can observe in Fig. 7, the configuration of the model using a threshold equal to 0.10 is the best performing one, followed by those obtained using a the threshold 0.05 and 0.25, respectively. The results successfully supported our choice of using a threshold able to maximize the recall more than the precision. The final results obtained by the CodiEsp task are those reported in Tab. 8. Looking at the final scores of precision and recall, it is worth to note that our model is more feasible for real use as a decision support system. Indeed, it is able to obtain a higher recall than precision and, as previously stated, this easily allows to select candidate codes for clinical reports.

5 Conclusion

In this paper, we faced the problem of clinical coding by applying several machine learning methods. We compared traditional classification approaches such as logistic regression, random forests, and SVM, with deep learning models, including LSTM, CNN, and BERT. The experimental analyses allowed us to propose a classification model based on two steps of execution: pre-filtering and classification. In the pre-filtering phase, we use a BERT based classification model to select a set of medical report sentences that we believe can be associated with one or more ICD10 codes. Later, we use a BiLSTM, CNN, and Self-Attention-based classifier to select the specific set of possible codes for the candidate sentence. The results obtained in the post hoc evaluation phase have shown that the approach proposed for the challenge is the best possible among those considered in this study. The results obtained for the challenge showed a MAP score of 0.202 for the CodiEsp-D task and 0.221 for the CodiEsp-P task. These are encouraging results given the difficulty of the task, and there is also the possibility of further improving the figures by better balancing the training data available among the various categories of codes. As future work, we expect to be able to associate the ICD10 code with the corresponding portion of text in order to implement a first strategy for explaining results.

References

1. Almagro, M., Montalvo, S., de Ilarraza, A.D., Pérez, A.: MAMTRA-MED at CLEF ehealth 2018: A combination of information retrieval techniques and neural networks for ICD-10 coding of death certificates. In: Cappellato, L., Ferro, N., Nie, J., Soulier, L. (eds.) Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum, Avignon, France, September 10-14, 2018. CEUR Workshop Proceedings, vol. 2125. CEUR-WS.org (2018), http://ceur-ws.org/Vol-2125/paper_110.pdf

2. Amin, S., Neumann, G., Dunfield, K., Vechkaeva, A., Chapman, K.A., Wixted, M.K.: Mlt-dfki at clef ehealth 2019: Multi-label classification of icd-10 codes with bert. In: CLEF (Working Notes) (2019)
3. Atutxa, A., Casillas, A., Ezeiza, N., Fresno, V., Goenaga, I., Gojenola, K., Martínez, R., Anchordoqui, M.O., Perez-de Viñaspre, O.: Ixamed at clef ehealth 2018 task 1: Icd10 coding with a sequence-to-sequence approach. In: CLEF (Working Notes). p. 1 (2018)
4. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
5. Basile, P., Croce, D., Basile, V., Polignano, M.: O verview of the evalita 2018 aspect-based sentiment analysis task (absita). EVALITA Evaluation of NLP and Speech Tools for Italian **12**, 10 (2018)
6. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics **5**, 135–146 (2017)
7. Cabot, C., Soualmia, L.F., Darmoni, S.J.: Sibm at clef ehealth evaluation lab 2017: Multilingual information extraction with cim-ind. In: CLEF (Working Notes) (2017)
8. Cañete, J., Chaperon, G., Fuentes, R., Pérez, J.: Spanish pre-trained bert model and evaluation data. In: to appear in PML4DC at ICLR 2020 (2020)
9. Chapman, W.W., Haug, P.J.: Comparing expert systems for identifying chest x-ray reports that support pneumonia. In: Proceedings of the AMIA Symposium. p. 216. American Medical Informatics Association (1999)
10. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: NAACL-HLT (1) (2019)
11. Fukushima, K., Miyake, S.: Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In: Competition and cooperation in neural nets, pp. 267–285. Springer (1982)
12. Géron, A.: Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. O'Reilly Media (2019)
13. Goeuriot, L., Suominen, H., Kelly, L., Miranda-Escalada, A., Krallinger, M., Liu, Z., Pasi, G., Saez Gonzales, G., Viviani, M., Xu, C.: Overview of the CLEF eHealth evaluation lab 2020. In: Arampatzis, A., Kanoulas, E., Tsikrika, T., Vrochidis, S., Joho, H., Lioma, C., Eickhoff, C., Névéol, A., andNicola Ferro, L.C. (eds.) Experimental IR Meets Multilinguality, Multimodality, and Interaction: Proceedings of the Eleventh International Conference of the CLEF Association (CLEF 2020) . LNCS Volume number: 12260 (2020)
14. Grave, E., Bojanowski, P., Gupta, P., Joulin, A., Mikolov, T.: Learning word vectors for 157 languages. In: Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018) (2018)
15. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
16. Hubel, D.H., Wiesel, T.N.: Receptive fields of single neurones in the cat's striate cortex. The Journal of physiology **148**(3), 574 (1959)
17. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
18. Kukafka, R., Bales, M.E., Burkhardt, A., Friedman, C.: Human and automated coding of rehabilitation discharge summaries according to the international classification of functioning, disability, and health. Journal of the American Medical Informatics Association **13**(5), 508–515 (2006)

19. Miftahutdinov, Z., Tutubalina, E.: Kfu at clef ehealth 2017 task 1: Icd-10 coding of english death certificates with recurrent neural networks. In: CLEF (Working Notes) (2017)
20. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
21. Miranda-Escalada, A., Gonzalez-Agirre, A., Armengol-Estapé, J., Krallinger, M.: Overview of automatic clinical coding: annotations, guidelines, and solutions for non-english clinical cases at codiesp track of CLEF eHealth 2020. In: Working Notes of Conference and Labs of the Evaluation (CLEF) Forum. CEUR Workshop Proceedings (2020)
22. Organization, W.H., et al.: The ICD-10 classification of mental and behavioural disorders: diagnostic criteria for research, vol. 2. World Health Organization (1993)
23. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in python. the Journal of machine Learning research **12**, 2825–2830 (2011)
24. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014)
25. Polignano, M., Basile, P., de Gemmis, M., Semeraro, G.: A comparison of word-embeddings in emotion detection from text using bilstm, cnn and self-attention. In: Adjunct Publication of the 27th Conference on User Modeling, Adaptation and Personalization. pp. 63–68 (2019)
26. Polignano, M., Basile, P., de Gemmis, M., Semeraro, G., Basile, V.: Alberto: Italian bert language understanding model for nlp challenging tasks based on tweets. In: CLiC-it (2019)
27. Polignano, M., Basile, V., Basile, P., de Gemmis, M., Semeraro, G.: AlBERTo: Modeling Italian Social Media Language with BERT. Italian Journal of Computational Linguistics - IJCOL **-2, n.2** (2019)
28. Sänger, M., Weber, L., Kittner, M., Leser, U.: Classifying german animal experiment summaries with multi-lingual bert at clef ehealth 2019 task 1. In: CLEF (Working Notes) (2019)
29. Song, S., Huang, H., Ruan, T.: Abstractive text summarization using lstm-cnn based deep learning. Multimedia Tools and Applications **78**(1), 857–875 (2019)
30. Stanfill, M.H., Williams, M., Fenton, S.H., Jenders, R.A., Hersh, W.R.: A systematic literature review of automated clinical coding and classification systems. Journal of the American Medical Informatics Association **17**(6), 646–651 (2010)
31. Tiedemann, J.: Parallel data, tools and interfaces in opus. In: Lrec. vol. 2012, pp. 2214–2218 (2012)
32. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
33. Wang, S., Zhu, Y., Gao, W., Cao, M., Li, M.: Emotion-semantic-enhanced bidirectional lstm with multi-head attention mechanism for microblog sentiment analysis. Information **11**(5), 280 (2020)
34. Zheng, G., Mukherjee, S., Dong, X.L., Li, F.: Opentag: Open attribute value extraction from product profiles. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 1049–1058. ACM (2018)
35. Zhou, J., Cao, Y., Wang, X., Li, P., Xu, W.: Deep recurrent models with fast-forward connections for neural machine translation. Transactions of the Association for Computational Linguistics **4**, 371–383 (2016)