Interpretable Deep Learning to Map Diagnostic Texts to ICD10 Codes

Aitziber Atutxa, Arantza Díaz de Ilarraza, Koldo Gojenola, Maite Oronoz, Olatz Perez-de-Viñaspre

Department of Languages and Computer Systems.
IXA Research Group: `http://ixa.eus`.
University of the Basque Country (UPV-EHU)
{*aitziber.atucha, a.diazdeilarraza, koldo.gojenola, maite.oronoz, olatz.perezdevinaspre*}*@ehu.eus*

Abstract

Background. Automatic extraction of morbid disease or conditions contained in Death Certificates is a critical process, useful for billing, epidemiological studies and comparison across countries. The fact that these clinical documents are written in regular natural language makes the automatic coding process difficult because, often, spontaneous terms diverge strongly from standard reference terminology such as the International Classification of Diseases (ICD).

Objective. Our aim is to propose a general and multilingual approach to render Diagnostic Terms into the standard framework provided by the ICD. We have evaluated our proposal on a set of clinical texts written in French, Hungarian and Italian.

Methods. ICD-10 encoding is a multi-class classification problem with an extensive (thousands) number of classes. After considering several approaches, we tackle our objective as a sequence-to-sequence task. According to current trends, we opted to use neural networks. We tested different types of neural architectures on three datasets in which Diagnostic Terms (DTs) have their ICD-10 codes associated.

Results and conclusions. Our results give a new state-of-the art on multilingual ICD-10 coding, outperforming several alternative approaches, and showing the feasibility of automatic ICD-10 prediction obtaining an F-measure of 0.838, 0.963 and 0.952 for French, Hungarian and Italian, respectively. Additionally, the results are interpretable, providing experts with supporting evidence when confronted with coding decisions, as the model is able to show the alignments between the original text and each output code.

Keywords: International Classification of Diseases, Electronic Health Records, Sequence-to-Sequence mapping, Neural Machine Translation

1. Introduction

Death Certificates are clinical text documents written by clinicians. They are typically accompanied by numerical codes to describe the morbid disease or conditions that led to the death of an individual. These codes originate from the International Classification of Diseases (ICD)[1]. They ensure normalization upon the ways different clinicians and countries employ for writing Diagnostic Terms (DTs) to describe the same disease. For instance, the English standard description of the E141 code is *Unspecified diabetes mellitus: With ketoacidosis*, but many clinicians use different variants. Table 1 presents several variants taken from Electronic Death Certificates in different languages.

[1]In this paper the 10th revision of the ICD classification will be referred to as ICD-10.

ICD10 code	French	Italian	Hungarian
E141	diabete acidocetosique	chetoacidosi diabetic scompensat	cukorbetegseg ketoacidosissal
	decompensation diabeto-cetosique	acidosi diabetic	diab versavanyodas
	diabete acidosique	chetoacidosi diabetic	diabetes ketoacidosis
	diabete cetosique insuline	acetonemia diabetic	diabeteses ketoacidozis
	acido-cetose diabetique	acidosi metabolic diabetic	diabeteses acidosis
	acidocetose diabetique	chetonemia diabetic	ketoacidotikus diab

Table 1: Example of different DTs to express the E141 code in French, Italian and Hungarian.

The coding is useful for billing, epidemiological studies or comparison across countries. The process of assigning diagnosis codes is labor intensive, costly and error prone due to the fact that it is carried out by human coders.

The training of experts to encode the ICD is expensive and time-consuming. According to Lang [1], about $25 billion per year is spent in the USA in encoding records with ICD codes. In addition, the ICD is evolving and, therefore, experts have to adapt to the most recent revision available. Even if standard diagnostic terminology is well defined and widely internationalized (the ICD exists in 43 languages and is used in around 117 countries), for obvious reasons, physicians usually use their own non-standard expressions that seldom agree with the standard. Hence, automatic ICD encoding is a very useful clinical application based on information extracted from text. Natural language processing methods are specially suitable to satisfactorily solve this task.

There have been many attempts to automatically extract ICD codes from clinical documents. This paper presents the application of different sequence-to-sequence models to map DTs to ICD-10 codes [2], a large-scale multi-class classification task.

Neural network models have revolutionized AI, especially in sequence-to-sequence mapping [3, 4]. Recently, a number of open systems have emerged [5, 6, 7], allowing experimentation of different approaches in a flexible way. Following this paradigm, the task can be viewed as a special type of machine translation (MT). In this work, we consider both input (plain text for DTs) and output texts (codes) as if they were two different languages, and we try to translate from one to another.

Our approach is general and has been applied to a varied set of languages, namely, French, Italian and Hungarian, obtaining the best results in the CLEF eHealth 2018 Task 1: "Multilingual Information Extraction - ICD10 coding". CLEF (Conference and Labs of the Evaluation Forum) is a well-known international initiative that, since 2000, has run campaigns for the systematic evaluation of information access systems, playing a leading role in stimulating investigation and research in a wide range of key areas in the information retrieval domain. CLEF is especially interesting for the comparison of approaches dealing with a specific task which is considered of special interest for the community. Besides CLEF, there have been other main venues related to the automatic evaluation of tasks on medical texts in the last years, like the Clinical Natural Language Processing Challenges (*i2b2, n2c2*) [8, 9] and the BioNLP-Shared Tasks [10]. The CLEF 2017 and 2018 eHealth shared tasks [11, 12] have been the reference forum on multilingual ICD coding. The task consisted of mapping Death Certificate lines containing DTs to the relevant ICD-10 codes, with eleven and fourteen teams participating in 2017 and 2018, respectively. In the 2017 edition, the languages under study were English and French while, in 2018, the languages were Hungarian, Italian and French.

2. Related work

~~The rationale behind this review is to summarize the main approaches to automatic ICD coding and their relation to the present work. Besides, we present how the scientific community deals with this problem in devoted shared tasks.~~

The first attempts to automatically extract ICD codes from clinical documents date back to the 1990s (more precisely, ICD-9). For many years software for ICD-coding used Dictionary Matching and Pattern Matching methods to search keywords or clusters of keywords to identify ICD codes [13]. These methods were not powerful enough because the task presents a complex characterization: large-scale multi-class classification, treatment of non-standard language, and alignment issues between spontaneous writing and ICD codes. Nowadays, there are more sophisticated approaches to tackle this task, ranging from knowledge-based solutions to statistical [14, 15] and deep learning ones.

Rule-based systems are still used with good accuracy when the terms to be coded follow regular patterns, the task is limited to one DT one ICD assignment, and the number of ICDs is quite small. Unfortunately, these conditions seldom apply. An example is described in [16] where in order to classify mental severity symptoms in psychiatric records, knowledge-based methods outperform those obtained by neural networks. In the first shared task related to ICD coding, the Computational Medicine Challenge (CMC) [8], Farkas and Szarvas [17] started to address this task by replacing some steps in the construction of hand-crafted-systems with machine learning (ML). They realized that manually building rules was not straightforward and it was time consuming. They worked with 45 classes and not with the entire scope of the ICD-10 catalog (thousands of classes). Pérez et al. [18] extracted the encoding rules in the form of inferred Weighted Finite-State Transducers (WFST) from the corpus to produce normalized alternatives to a given spontaneous string and applied text similarity to select the standard string in the ICD framework that best matched the normalized alternatives. In the current work, a first step of normalization is not required, as neural systems learn how to manage non-standard language and, in addition, instead of a one-to-one (1:1) alignment, a much harder N:M term-code alignment is necessary.

When the range of classes to be tagged is high and the corpus is big enough, machine learning based techniques have been successful [19, 20]. Koopman et al. [21] trained Support Vector Machine (SVM) classifiers to identify cancers in a cascaded architecture, first identifying the presence of a cancer and later classifying its type according to the ICD-10 classification. In another work, Koopman et al. [22] used a machine learning approach and keyword matching rules to identify the presence of diabetes, influenza, pneumonia and HIV in Death Certificates. An SVM classifier with term-based and concept-based features (i.e. SNOMED CT concepts) was trained for each of the four diseases. A single classifier model was also trained for each ICD-10 code representing each of these diseases. In the rule-based approach a set of keywords provided by experts was used to indicate whether a Death

Certificate was a positive or a negative match for a particular disease. These works approach the problem as document classification (Death Certificate classification), instead of term encoding. In [23], separate machine learning models were trained with data from unstructured text, semi-structured text and structured tabular data to create a multimodal machine learning model that predicts ICD-10 diagnostic codes. Recently, neural networks boosted the results in many NLP tasks [24], also in the medical domain [25] including ICD coding [26, 27]. Transfer learning [28] has also been used combined with deep learning, as in [29] where MeSH [30] domain knowledge is transferred to a deep learning model by pre-training it with MeSH datasets and fine-tuning the neural network on ICD-9 datasets. The authors state that transfer learning is the key element to improve ICD-9 encoding.

In the last CLEF eHealth ICD encoding shared tasks [11, 12], dictionaries and corpora were the main resources and n-gram patterns, machine learning and neural approaches the most employed methods. In the CLEF 2017 edition, Ebersbach et al. [31] and Zweigenbaum and Lavergne [32], along with many other teams, relied on lexical resources and made use of different ML methods, dictionary projections from medical ontologies, expansion of synonyms and edit distance calculation. In this CLEF eHealth 2017 edition, only Miftahutdinov and Tutubalina [33] implemented Recurrent Neural Networks to assign ICD-10 codes to fragments of English Death Certificates. Their system used a LSTM to map the input sequence into a vector representation, and then another LSTM to decode the target sequence from the vector, obtaining an F-measure of 0.850.

In the latest CLEF 2018 edition, 12 teams out of 14 used the dictionaries supplied by the organization. Cossin et al. [34] applied a dictionary-based approach, including a module for the detection of typos and another module for synonym expansion. Regarding machine learning approaches, Almagro et al. [35] implemented a supervised learning system using multilayer perceptrons and a One-vs-Rest (OVR) strategy, also experimenting with IR methods. In Gobeill and Ruch [36] a purely statistical instance-based approach was used by indexing all training sentences to feed a k-Nearest Neighbors algorithm. Neural architectures have been used to approximate the ICD encoding task. Jeblee et al. [37] used an ensemble model for prediction which includes n-gram matching followed by an ensemble of a convolutional neural network and a recurrent neural network (RNN) encoder-decoder. Their system employed embeddings learned on the data provided, as well as on language-specific Wikipedia corpora.

Ševa et al. [38] focused on the setup and evaluation of a language-independent neural architecture using a multi-language word embedding space. The authors obtained these word representations by concatenating Italian, French and Hungarian Fastext[2] pre-trained embeddings. Their approach builds on two RNNs, modelling the extraction and classification based on Long Short-Term Memories (LSTM) with an attention mechanism. Another idea to initialize the networks in a recurrent neural model, in this case the weights of the final network, is given in [39] where co-ocurrences between ICD classes in the training data and the hierarchical structure of ICD-10 are used. Note that in the latter work the whole Death Certificates including the events

[2]https://github.com/facebookresearch/fastText/blob/master/docs/crawl-vectors.md

leading to death (diagnoses with ICD-10 codes) and the main cause of death are employed. In our system co-occurrences are modeled implicitly by the sequence to sequence approach.

While many CLEF systems [38] use external general domain pre-trained word-embeddings, we decided to initialize the embeddings randomly, after a previous attempt initializing the embedding layer with external pre-trained embeddings resulted in a performance decrease. This decay can be due to the fact that external embeddings (using Word2Vec, GloVe or FastText) are calculated as the result of a general learning task (the embeddings correspond to the hidden states of the system), that might not be the kind of representation needed for the task at hand. Therefore, our system obtains the embeddings from scratch, recalculating them during the training process of the ICD classification task itself.

Some of the issues we considered when we chose the method to approach the ICD encoding task were: i) the large scale of the ICD classification (more than 5,000 codes from around 500,000 coding items in French and Hungarian and 2,500 codes from 75,000 coding items in Italian), ii) the lexical variability in the texts and, iii) the need of a multilingual approach. These requirements led us to approach the problem with a neural architecture, a sequence-to-sequence approach which, combined with an attention mechanism, tackles alignment problems, a special issue in the ICD encoding task.

3. Materials and methods

3.1. Corpora

We employed three datasets provided for the *Multilingual Information Extraction* CLEF-2018 Task 1 [12]. The French dataset (CépiDc[3], [40]) comprises 135,000 Death Certificates collected from 2006 to 2015. The Italian dataset (ISTAT[4]) stores around 18,000 synthetic Death Certificates. They were artificially built from original Death Certificates, generated in 2016, with the aim of preserving confidentiality. The lines of each synthetic Death Certificate were obtained from different original Certificates, always ensuring topical coherence and preserving the sequence of death causes (line 1 of a synthetic certificate was created using line 1 of a real certificate). The age and sex of the patient were also maintained. Hence, this synthetic corpus provides a realistic simulation of language and terminology found in Italian Death Certificates, together with the official coding. Finally, the Hungarian dataset consists of 100,000 Death Certificates randomly extracted from a set of non-electronic Death Certificates for the year 2016 electronically transcribed afterwards. This corpus was provided by the Hungarian central statistical office (KSH[5]). See Table 2 for more details.

One of the sections in the Death Certificates [41] is a piece of text containing one or more lines that describe the morbid diseases or conditions that led to the death of an individual (an example is provided in the last column of Table 3). The job of nosologists or human mortality medical coders is to

[3]http://www.cepic.inserm.fr
[4]http://www.istat.it
[5]http://www.ksh.hu

	French		Italian		Hungarian	
	Train	Test	Train	Test	Train	Test
Death Certificates	125,384	11,931	14,502	3,618	84,703	21,176
Avg. DTs per ICD10	17.59	-	7.03	-	12.74	-
#ICD10 Codes	509,103	48,948	60,955	15,789	392,020	98,264
#Uniq. ICD10s	3,723	1,806	1,443	903	3,124	2,011

Table 2: Short Description of relevant aspects of the corpus.

assign an ICD code to each of these conditions. In the cases where multiple conditions are coded, the nosologists decide the corresponding code for each DT after having seen how each condition modifies or relates to one another. The dataset provided for the shared task only contains the set of lines that describe those diagnoses of medical conditions. Some lines contain a single diagnostic term, and its corresponding code, while others contain various spontaneous DTs that correspond to a set of ICD codes.

All datasets were preprocessed to align the DTs at document line level as expressed in the original Death Certificate, along with their corresponding ICD-10 codes, as both pieces of information were contained in different files. Note that a 1:1 correspondence between a DT and the ICD-10 could not be assured because DTs were encoded by humans at document line level, not at diagnostic term level. Therefore, only when a document line contained a unique DT, then there was a 1:1 DT-ICD10 correspondence. However, when a line contained several DTs, 1:1 alignment was not guaranteed (see Table 3). This poses serious problems for text similarity or rule based strategies but not for sequence-to-sequence systems, especially when they use attention mechanisms, which allow the right alignments to be learned (see section

3.2.2).

ICD-10 codes	Standard Diagnostic-Term	Original Text
N179	insuffisance renale aigue	insuffisance renale aigue , masse tumorale de la tete du pancreas responsable d 'une compression duodenale avec nausees / vomissements - pouvant correspondre a un 2eme primitif ou metastase
C250	tumeur maligne tete pancreas	
K566	compression duodenale	
R11	nausees vomissements	

Table 3: Example of multiple DTs within the same document line. The *Original Text* column contains several Spontaneous Diagnostic Terms. The column named *Standard Diagnostic Term* shows the standard term describing each of the multiple ICD-10 codes manually assigned to the Original Text.

3.2. Architecture

3.2.1. Baseline: Levenshtein Edit Distance

Edit distance is used to quantify similarity between two strings, counting the minimum number of operations required to transform one string into another. The most common metric is the Levenshtein Distance [42] in where the basic edit operations are removal, insertion and substitution of a single character. This metric finds the minimum distance for each spontaneous diagnostic term (SpoDT) with respect to all standard Diagnostic Terms (DictDT), obtaining the best candidate match (see equation 1).

$$minLev(SpoDT, DictDTs) \tag{1}$$

To identify each Spontaneous DT in a text containing several DTs, we made certain assumptions. We considered colon, comma, semicolon, coordination (*and or*), and certain prepositions, such as *with*, as DT separators.

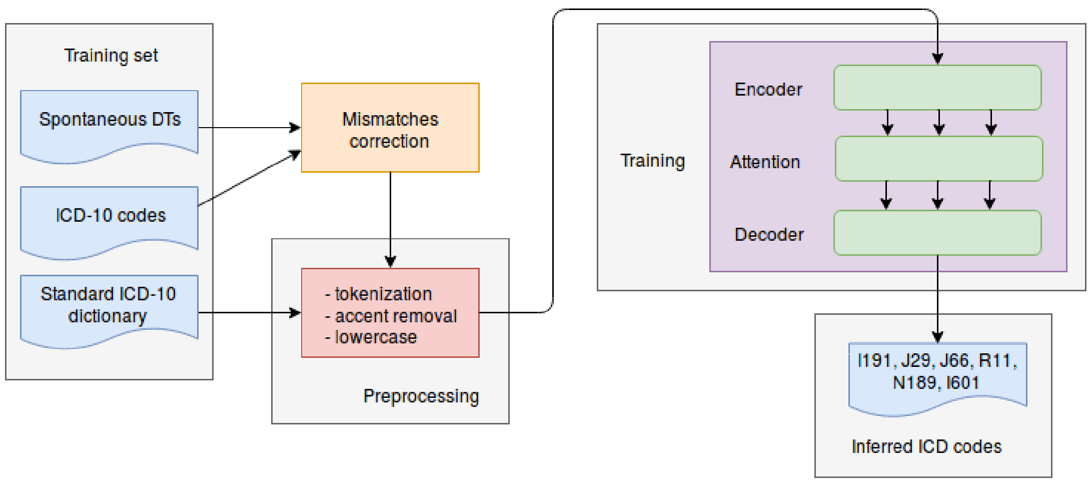


Figure 1: General architecture of the system.

3.2.2. ICD-10 coding as machine translation

In the present work we adopted a sequence-to-sequence neural machine translation (NMT) solution. Formally, having an input text $X = x_1, x_2 ... x_n$, and an output sequence $Y = y_1 ... y_m$, the goal is to model $P(Y|X)$. X corresponds to one document line (only those lines containing one or several DTs) and Y corresponds to one or more ICD-10 codes associated to the input DTs. Sequence-to-sequence systems calculate $P(Y|X)$ by modeling it as a step sequence where $P(Y|X) = \prod_{t=1}^{m} P(y_t|y_{1..t-1}, X)$. In a neural sequence-to-sequence model, the previous formula would be stated as $P(Y|X; \theta) = softmax(W_o s_t + b_o)$, where θ represents the neural network parameters that are automatically tuned to make the neural network minimize the error. W_o

corresponds to the output layer weight matrix, b_o is the bias term of this layer, and s_t represents the hidden state of the neural network at step t.

Figure 1 presents the main components of the system. The training set is composed of a set of 1:1 or N:M pairs of the form (spontaneous DT terms, ICD codes). Initial preprocessing tries to minimize the mismatches in alignments of DTs and codes that appear in the relatively noisy input examples. The text containing the Diagnostic Terms of the Death Certificate and the corresponding codes were supplied disaggregated in different files; one file stored the text containing the Diagnostic Terms as appearing in the Death Certificate (Spontaneous DTs) and the location of the text occurrence (document index and line index). Another file stored the corresponding ICD-10 codes, the standard text description of each code and the location (document index and line index). In order to obtain the set of examples to train the system, these two pieces of information had to be related through their location in the Death Certificate. Along the process, we fixed, when possible, all the location mismatches in order to improve the quality of the training set. For example, there were several erroneous instances where the location for an ICD code is the n^{th} line and there was no DT in the n^{th} line. In fact, the spontaneous DT associated to it appeared in the $(n+1)^{th}$ line. The corrected pairs (1:1 or N:M) together with the standard ICD dictionary containing *(standard term, ICD code)* pairs (1:1) were then tokenized, converted to lower case and accents were removed.

The use of pre-trained word embeddings is generalized for many NLP neural models [43, 44, 45]. Regarding NMT, when source-target data is not big enough, those pre-trained embeddings have proven to be useful [46, 47].

The task at hand shows two differences with respect to other general NMT systems. First, it is domain specific, in particular, medical domain and, second, it consists in translating from words directly to codes, that is, mapping between two very different spaces. We think that in this case the origin of the pre-trained embeddings plays an important role on how positive their impact in the task might be. As we mentioned in section 2, word embeddings are the vectors generated in a hidden layer as a side-effect of maximizing a given probability, for example the probability of predicting the center word given context words (CBOW) or predicting context words given a word (Skip-Gram) [48]. These tasks are general enough to make these embeddings useful in many NLP tasks, but the task at hand for CLEF, namely, translate a DT in an ICD, is very different in nature and too specific. The first layer of the neural networks (depending on the architecture, a different neuron layer is applied) is used to learn the word representations or word-embeddings. We initialized it with random values, and these values were updated during the training process. Therefore we can say that they were tailored for the DT-ICD translation. We also experimented with pre-trained embeddings but we did not obtain good results.

The encoder-decoder architecture using RNNs is currently the most popular solution for NMT, especially when combined with attention mechanisms to tackle alignment problems and their limitation on long sequences by concentrating the attention on the relevant input parts. Recurrent Neural Networks or RNNs are specially useful when working with sequential data (as text), thanks to its ability to maintain information about previous inputs using an internal memory. RNNs carry input information across neurons by means of recursive looping through an internal hidden state, thus trying to maintain information about the whole sequence up to each input word. In theory, RNNs can maintain information from the beginning of the sentence but, in practice, this does not always happen and information from remote words becomes insignificant. Long distance context is what Long Short Term Memory (or LSTM) units are good at keeping inside the neural network. Using a LSTM unit is like adding a memory unit that can remember context from the very beginning of the input.

Encoder-decoder architectures present two distinct blocks, the encoder and the decoder. Generally speaking, an RNN encoder-decoder approach is an extension of a language model. The encoder block reads the input string word by word and obtains a fixed length vector representation. Then, the decoder learns to sequentially predict an output code (icd_m) at position m in the input string context ($DTs = w_0..w_j$) encoded by the encoder and the codes predicted so far ($icd_0..icd_{m-1}$).

As mentioned before, and to avoid the gradient vanishing problem, the RNN employed makes use of long short-term memory units (LSTM) [49]. When training a classic RNN using back-propagation, due to the computation involved in the process, the gradients can tend to zero or infinity. With LSTM units, gradients can flow unchanged. Using a gate-based system, LSTMs are able to automatically regulate how much of the "previous history" context should persist and how much should be renewed. Equations 2 to 6 represent a basic LSTM cell as represented in Figure 2.a.

$$
\begin{aligned}
i_t &= \sigma(W_{x_i} x_t + W_{h_i} h_{t-1} + W_{c_i} c_{t-1} + b_i) && (2)\\
\tilde{c}_t &= \tanh(W_{x_c} x_t + W_{h_c} h_{t-1} + W_{c_i} c_{t-1} + b_c) && (3)\\
c_t &= (1 - i_t) \odot c_{t-1} + i_t \odot \tilde{c}_t && (4)\\
o_t &= \sigma(W_{x_o} x_t + W_{h_o} h_{t-1} + W_{c_o} c_t + b_o) && (5)\\
h_t &= o_t \odot \tanh(c_t) && (6)
\end{aligned}
$$

- x_i corresponds to the embedding representation of w_i.
- σ and tanh represent the sigmoid and hyperbolic tangent, introducing non-linearity in the network.
- t and $t - 1$ correspond to the current and previous time steps, respectively.
- c_t defines the current state of the memory cell controlling how much of the previous context ($(1 - i_t) \odot c_{t-1}$) should be forgotten and how it is updated ($i_t \odot \tilde{c}_t$.)
- i_t represents which values will get updated and $\tilde{c}_t$ represents which new candidates could be added.
- o_t defines, through the *sigmoid* function (σ), which part of the information stored in the cell will go to the output.
- h_t corresponds to the hidden state. In Bi-LSTMs h_t gets calculated as the concatenation of right to left $\overrightarrow{h_t}$ and left to right $\overleftarrow{h_t}$ hidden states.

Besides the RNN, we also experimented with other alternatives that have proven to be successful for translation, as fully Convolutional Neural Networks (CNN) and self-attention (Transformer) networks.

A CNN network intends to identify the most relevant aspects of a Spontaneous $DT (DT = w_0..w_n)$ and represent these aspects in a fixed length vector. The process consists of moving a sliding-window of k words over the text obtaining several overlapping subsequences $(s_1, s_2...s_{n-k+1})$[6]. Then a filter is applied to each subsequence with the aim of capturing some major aspect of the subsequence. A filter is simply a dot-product of a given vector representing the subsequence[7]. These weight values will be updated repeatedly over the training process using gradient descent. The result will be a scalar value $(v_i = s_i.u)$. Different u $(u_1, u_2..u_j)$ filters can be applied over the same subsequence s_i obtaining a vector l_i $(l_i = v_{i,1}, v_{i,2}...v_{i,j})$. Ideally each $v_{i,x}$ vector will identify a different aspect of the subsequence s_i. Next, the "pooling" operation applies over the $l_{1:n-k+1}$ vectors to combine them reducing the dimension to represent the initial Spontaneous DT. The reduction consists in finding the maximum value across each position in the vector l, which indicates the most important signal in that position; this helps to eliminate noise and also ensures that all sequences no matter how long are represented as a fixed length vector. This reduction is graphically represented as a triangle as in Figure 2.b. The CNN decoder is similar to the encoder but it has an additional attention mechanism at every layer and a fully connected layer with a softmax to perform the actual ICD prediction. The fully

[6] Assuming that the window slides in steps of one word.

[7] Note that the representation of each subsequence is usually the concatenation of all the word vectors contained in the subsequence with a weight vector (u)

connected layer will use information of both the input string (one or several DTs) encoded by the encoder and the ICD codes predicted so far. For more details see [50] and [51]. Equation 7 corresponds to the generalization for a system with multiple convolutional layers (cl) of the basic convolutional filter operation.

$$h_i^{cl} = v(W^{cl}[h_{i-[k/2]}^{cl-1}; ...; h_{i+[k/2]}^{cl-1}] + b^{cl}) + h_i^{cl-1} \tag{7}$$

In the Transformer architecture, a small constant step number is applied. Each step consists of a self-attention mechanism which directly models relationships among all words in a source text by tuning weights. For both the encoder and the decoder the basic operation is presented in equation 8.

$$C = softmax\Big(\frac{QW^Q(KW^K)^T}{\sqrt{d}}\Big)VW^V \tag{8}$$

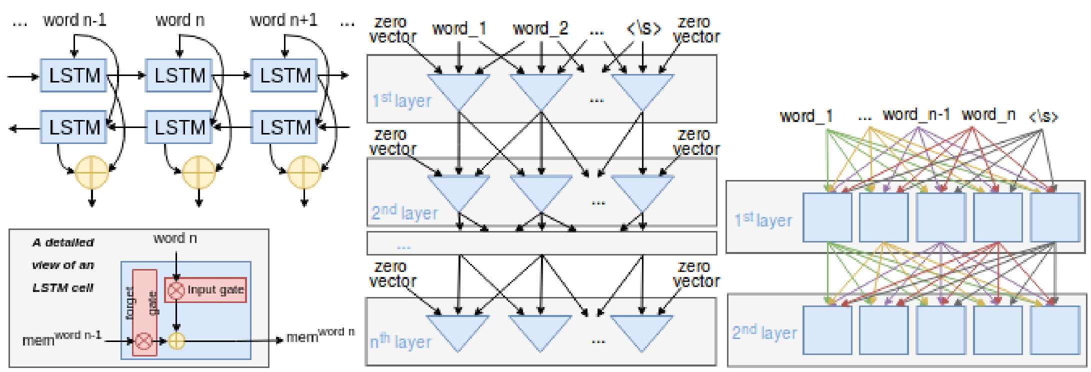
Q is called the Query and it is usually the last hidden state of the decoder, so it represents the target ICD codes. K is called the Key and represents the source Spontaneous DT terms (through the encoder's last hidden state). So, the softmax, as stated, will assign bigger values to sources that are "closer" to targets, and V is referred to as the values and is equal to K, so the result of the softmax multiplied by V gives the most probable value from the source[8].

In general terms, in CNN and Transformer architectures, position is not intrinsically part of the system, as opposed to RNNs[9]. Therefore, words are augmented with positional information by adding the word embedding and the positional embedding.

[8]see http://jalammar.github.io/illustrated-transformer/ for further reading

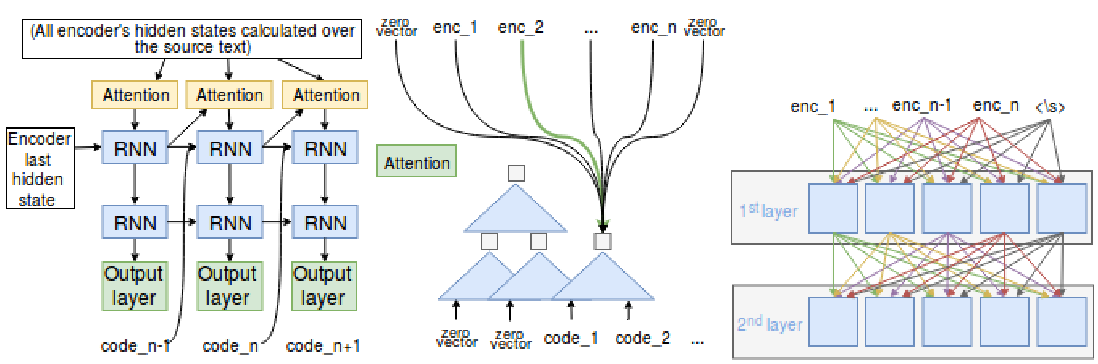
[9]Sequentiality intrinsically encodes the position of the source items.

Figure 2 graphically shows the different algorithms. From the multiple toolkits available we chose Sockeye [7] because of its flexibility regarding the available architectures, including Deep Recurrent Neural Networks with Attention [52], Transformer Models with self-attention [53] and fully convolutional sequence-to-sequence models [50].



(a) biLSTM encoder (b) CNN encoder (c) Transformer encoder

Figure 2: biLSTM, CNN and self-Transformer encoders



(a) RNN decoder (b) CNN decoder (c) Transformer decoder

Figure 3: RNN, CNN and self-Transformer decoders

3.3. Evaluation

For evaluation, system performance was assessed by the usual information extraction metrics: recall, precision and F-measure over the set of predicted and gold standard codes (specifically, we used β=1). Matches (true positives) were counted for each ICD-10 code supplied, matching the reference for the associated document.

$$Precision = \frac{true\ positives}{true\ positives + false\ positives} \tag{9}$$

$$Recall = \frac{true\ positives}{true\ positives + false\ negatives} \tag{10}$$

$$F - measure = \frac{(1+\beta^2) \times precision \times recall}{\beta \times precision \times recall} \tag{11}$$

4. Results

Table 4 presents the results of our system on the CLEF 2018 Shared Task 1 data [12]. For the sake of comparison, we also show the results for the best systems that competed in the shared task and which were described in the related work section (section 2). For the French dataset, there were 18 official runs (or systems) from 12 teams. For the Hungarian dataset, there were 9 runs from 5 teams and, for the Italian raw dataset, 12 runs from 7 teams.

In our case, we developed different models on an initial partition of the data into training, validation and the initially provided test (60%-20%-20%

split). The neural network was trained on *batches* of the training set and evaluated repeatedly on the validation set until no improvement was reached. We experimented with different values for the most relevant parameters, including the embedding size for source and target tokens (256, 512 and 1024), the learning rate (from 0.0001 to 0.0005), the number of layers in encoder and decoder, the dropout rate for source and target embeddings, and the optimization algorithm. The final set of parameters uses an embedding size of 512 for source and target embeddings, 0.0003 learning rate, a single layer for encoder and decoder, no dropout and the Adam optimization algorithm. Regarding the transformer, it uses 8 heads (or parallel attention layers) in a multi-head context. We used fixed positional embeddings that are calculated deterministically using sinusoidal functions of different frequencies, related to each position [54], as they better generalize for sequences of lengths not found during training. By default, fixed positional embeddings require a size equal to that of the word embeddings. The best parameter setting was finally applied to the test set. From the built models, we selected the best two systems, which were evaluated on an unseen final test file by the shared task organizers. The final systems used the merged *train+validation* dataset in order to obtain the model parameters, leaving the initial test set for validation.

Using an Intel Xeon 3.00GHz processor with an NVIDIA TITAN X Pascal graphical processing (GPU) unit it took less than one hour for each experiment.

French			
System	**Prec**	**Rec**	**F1**
Baseline (Levenshtein)	0.578	0.543	0.56
Ours RNN-CNN	0.841	**0.835**	**0.838**
Ours RNN-Transformer	**0.846**	0.822	0.834 (-0.004)
Cossin et al. [34] dictionary-based (standard text)	0.794	0.779	0.786 (-0.052)
Cossin et al. [34] dictionary-based (ICD dict.)	0.782	0.772	0.777 (-0.061)
Gobeill and Ruch [36] instance-based learning	0.763	0.764	0.764 (-0.074)
Hungarian			
System	**Prec**	**Rec**	**F1**
Baseline (Levenshtein)	0.935	0.935	0.935
Ours CNN-RNN	**0.970**	**0.955**	**0.963**
Ours RNN-Transformer	0.968	0.954	0.961 (-0.002)
Almagro et al. [35] perceptron + One-vs-Rest	0.946	0.911	0.928 (-0.035)
Almagro et al. [35] IR	0.932	0.922	0.927 (-0.036)
Jeblee et al. [37] n-gram + word embeddings	0.922	0.897	0.910 (-0.053)
Italian			
System	**Prec**	**Rec**	**F1**
Baseline (Levenshtein)	0.822	0.794	0.808
Ours RNN-RNN	**0.960**	**0.945**	**0.952**
Ours Transformer-RNN	0.945	0.922	0.934 (-0.018)
Almagro et al. [35] perceptron + One-vs-Rest	0.917	0.875	0.895 (-0.057)
Almagro et al. [35] IR	0.931	0.861	0.895 (-0.057)
Jeblee et al. [37] n-gram + word embeddings	0.908	0.824	0.864 (-0.088)

Table 4: Performance of the best 5 systems for French, Hungarian and Italian, respectively.

5. Discussion

Table 4 shows that the sequence-to-sequence approach considerably outperforms the other systems for all languages, with differences of 5.2, 3.5 and 5.7 absolute points in F-measure with respect to the next best system for French, Hungarian and Italian, respectively.

Notice that the system beats the baseline as expected. Text-similarity approaches (Levenshtein) perform poorly on multiple DT line examples, where splitting each line to find the individual DTs causes alignment problems.

5.1. Influence of the variability in the terms and lists of terms

The use of non-standard language and the variability in the DTs is a source of errors (see Table 6 for the description of several error types). When multiple DTs appear in the same line, besides language variability, alignment issues arise, because identifying individual DTs becomes tricky. For example, two DTs separated by a comma, as in *"evolution terminale , insuffisance cardiaque" (ICD-codes R999 I509)*, might not obtain the same code as when they appear with a preposition, *"evolution terminale d'une insuffisance cardiaque" (ICD-code I509)*. However, having multiple terms can be helpful sometimes. Terms might be interrelated, so a term might thus help to find the right code for another one (see Figure 6).

5.2. Influence of neural architecture

Although we experimented with other combinations of encoder-decoder pairs, Table 4 presents the results of our best two systems for each language. The table shows important differences depending on the neural network architecture employed. There is no unique encoder-decoder combination that performs best for all languages, the best systems being RNN-CNN, CNN-RNN and RNN-RNN for French, Hungarian and Italian, respectively. Regarding the decoders, RNNs seem to work better for Italian, as the best combinations use an RNN decoder. In Hungarian the second best combination uses an RNN as decoder and, finally, none of the best combinations for French employs an RNN. The most plausible explanation for this fact originates from the data itself. RNNs are intrinsically sequential and consequently the order of the ICDs might be an issue in lines with multiple ICDs. It turns out that in Italian only 2% of the data corresponds to multiple ICD lines with an average length of 2.4 ICDs, while in Hungarian it is a 3% and in French it is around a 5% with and average length of 2.9 both. From those multiple ICD lines, Italian is the language showing the lowest order variability, then Hungarian and, finally, French with the highest order variability. As for the encoder, the opposite applies, because the language showing the highest word order freedom is Hungarian, then Italian and finally French. This suggests that the optimal algorithms should be selected after a careful experimentation.

Besides the general architecture, we also tested variations of the different hyperparameters. Our main experience is that the difference comes mostly from varying the architectures (CNN, RNN and transformer), rather than from adjusting the hyperparameters, with minor variations from the standard default values.

We made preliminary experiments using subword units known as byte-pair encodings [6] that try to define smaller segments than the word itself. This is based on the intuition that various word classes are translatable via units smaller than words, for instance names (via character copying or transliteration), compounds (via compositional translation), and cognates and loanwords (via phonological and morphological transformations). Although this can in principle be useful for dealing with unknown words, we did not find any significant improvement.

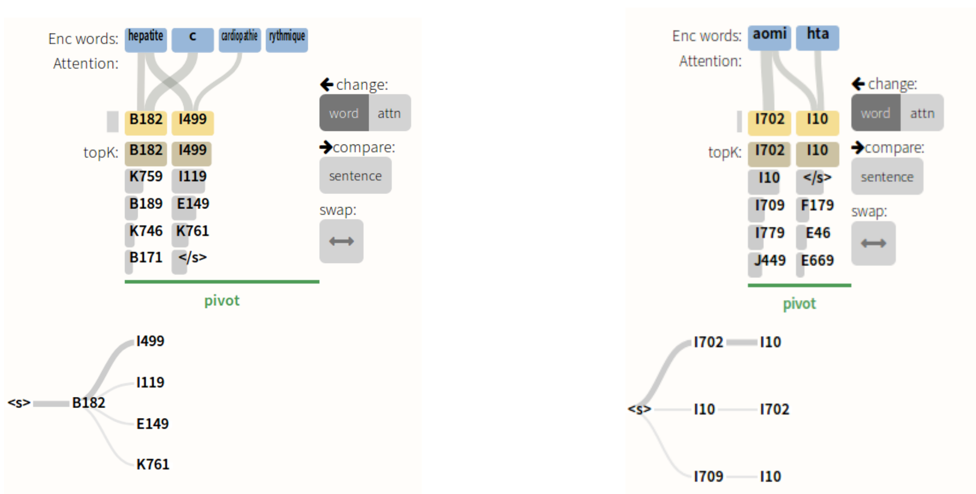
5.3. Interpretability

The examples in Table 3 show that the connection between the codes and the parts of the DTs is not annotated, obscuring the reasons behind specific coding decisions. As Li et al. pointed out [55]: "unlike traditional feature-based classifiers that assign and optimize weights to varieties of human interpretable features (parts-of-speech, named entities, word shapes, syntactic parse features etc) the behavior of deep learning models is much less easily interpreted". Recent works have made an effort towards this direction [56, 57].

Figure 4 shows how the network processes the DT input, using a visualization tool for neural sequence-to-sequence models [58]. The different values calculated in the process get represented graphically allowing users to understand the decisions at each step. The figure presents the three main components of the system: the encoder (in blue), the attention (linking encoder and decoder) and the decoder (in yellow). For instance, in Figure 4.a, for the input *hepatite c cardiopathie rythmique*, the right codes are correctly predicted as *B182* (hepatite C) and *I499* (Cardiopathie rythmique). The width of the attention lines represents the weight of each encoder state (the input) over the decoder predictions (the output). For example, the word "c" plays a relevant role to decide the right code for "hepatite c" (Figure 4.a), as all the codes at the top k candidates refer to hepatic diseases. The candidates, ranked by their probability, are B182 (*Hépatite virale chronique C*), K759 (*Maladie inflammatoire du foie, sans précision*), B189 (*Hépatite virale chronique, sans précision*) and K746 (*Cirrhoses, autres et sans précision*). The attention mechanism is one of the keys that helped to obtain accurate predictions, ~~comparing~~ compared to other participant groups that did not make use of attention. The lower part of Figure 4 shows the alternative paths that are evaluated when searching for the best code assignment, representing the path likelihood by the width of the connecting lines.

Figure 5 shows the encoder contexts analogous to the word "aomi" (Figure 4.b), presenting DTs related to the input. This way, the human coder can inspect similar encoding states which in this case allow to ascertain the linking between "aomi" and "arteriopathie obliterante des membres inferieurs" (as an acronym).

Figure 6 presents the alignment examples of two DTs and ICD-10 codes, showing the result of the sequence-to-sequence alignment. In addition to assigning codes with good accuracy, our system provides an interpretable result, aligning each code with its corresponding piece of text as in [59]. This information is not rendered by most ML approaches which, although accurate, do not provide any helpful information besides the result itself. This is relevant in medical environments where hospitals/clinicians require an understandable way to find the most informative evidence. We provided a physician with a sample of 78 ICDs appearing in 15 lines, with and without the alignment matrices. Overall, the physician found the system very helpful. Finding the right ICD for a given DT from scratch, that is, with-



(a) (b)

Figure 4: Examples of the ICD code generation process.

out having any system prediction at all, ~~implied to start looking~~ required the human annotator to look up the DT in the ICD coding classification as provided by the WHO (World Health Organization). This was hard since DTs do not usually match the standard terms, so when there was no perfect match the expert had to locate in the hierarchy the right chapter (DT generalization, e.g, *insuffisance renale terminale* corresponds to chapter XIV codes from N00N99, for Maladies de l'appareil génito-urinaire) and then dig ~~up~~ for the right code. Having a prediction flips the process~~,~~: the physician started by looking up ~~for~~ the predicted ICD code to obtain the standard term associated to it, and if the standard term was a synonym of the DT to be classified, the process finished successfully. But even when the prediction was not correct, as the ICDs follow a hierarchical structure, a partial prediction narrowed the search space reducing the coding time (e.g., *insuffisance*

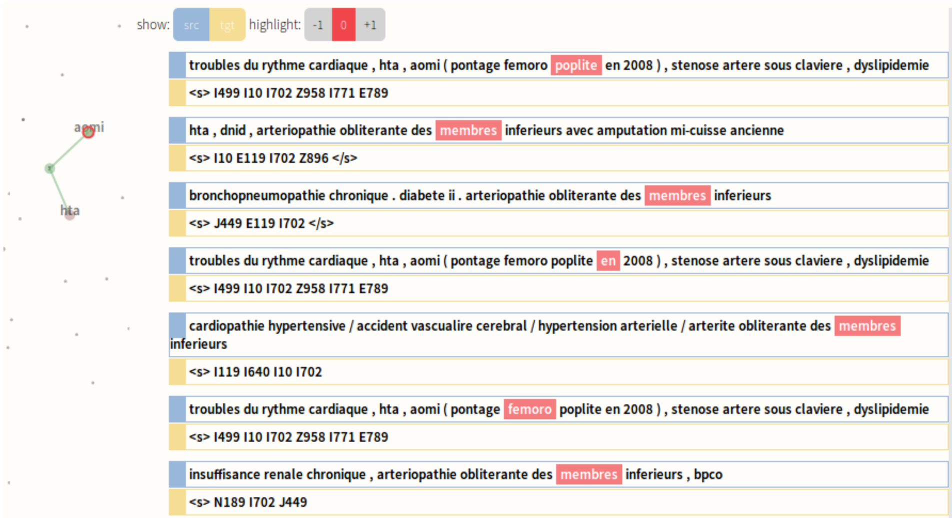
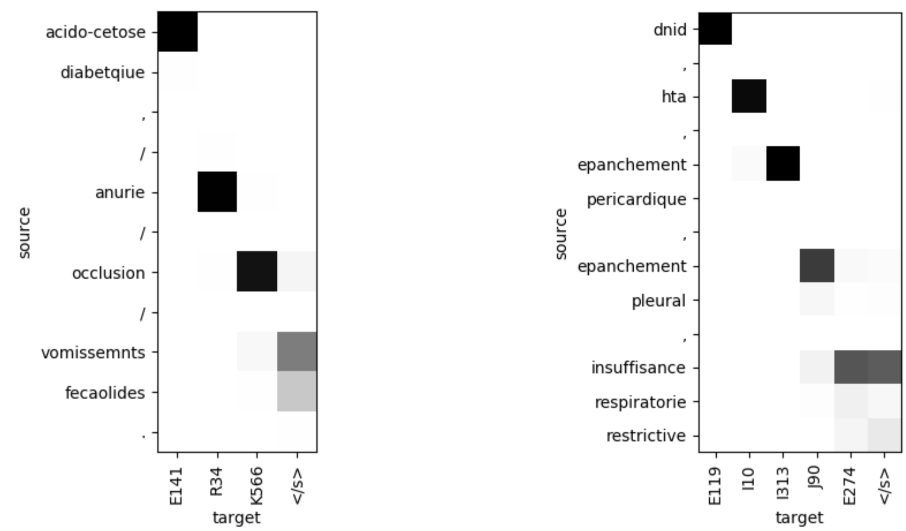


Figure 5: Related encoder contexts of the word "aomi" (Figure 4.b), showing DTs related to the input (e.g. "arteriopathie obliterante des membres inferieurs").

renale terminale was assigned the ICD code N180 which, although incorrect, is close to the correct code, N185). The alignment matrices, in particular, were useful when errors occur, ~~specially~~ especially in long lines that contain several DTs, as the alignments showed a one-to-one correspondence between a term and the predicted ICD, allowing the physician to focus on that exact pair and, if the prediction was wrong, the physician did not need to look for any other DT that could match the ICD.

In figure 6.a, we see how the DT *acido-cetose diabetqiue*, containing a spelling error, is paired with the code E141 that corresponds to the standard term *acidocetose diabetique*. Figure 6.b depicts how the system finds different types of evidence assigned to each DT word with respect to an ICD-10 code, illustrated by the DT *insuffisance respiratorie restrictive*, where each word adds value to the alignment with the term E274, and how some words are more relevant than others.



(a) (b)

Figure 6: Alignment examples.

Additionally, word embeddings calculated in the translation process capture similarities between elements, such as semantically similar equivalents, variants or spelling errors. Table 5 presents the closest words in the embedding space to a set of selected words. For example, given the French word *bronchique*, the set of close word embeddings includes variants (*bronchiques*), spelling errors (*brochique, bronchqiue*), or semantically related terms (*tracheobronchique*).

5.4. Effect of algorithms and training corpus size

We found it interesting to explore the relation between the types of algorithms and certain characteristics of the corpora, such as size and number of

Term (English)	French	Italian
	bronchique	bronchit
	brochique	bronchite
bronchial	bronchqiue	broncopat
	bronchiques	broncopneumopat
	bronchioque	tracheobronchite
	tracheobronchique	broncopneumopatia
	deshydratation	disidratazione
	hypovolemie	ipovolemia
dehydration	deshydrataion	idratazione
	deshydratee	deidratato
	dehydratation	disidratativa

Table 5: Example of close embeddings for the terms "bronchial" and "dehydration" in French and Italian.

codes. To do so, we artificially built several ~~size~~ differently-sized training corpora. Figure 7 shows how Transformer for encoding and CNN for decoding present the best results for small datasets, with an F-score difference of ~~10~~ 0.10 ~~absolute points with~~ from other approaches. However, it also shows the worst performance on the full dataset. Conversely, architectures based on CNN and RNN do not obtain a good accuracy with small training sizes, but achieve the best results with bigger datasets. This could be due to the fact that Transformer is less constrained to positional information, generalizes better, and suffers less from sparsity. The attentional Transformer-RNN pair is the most regular architecture for both small and big datasets. These results present interesting ideas for the implementation of neural architectures, depending on the language or the amount of data.

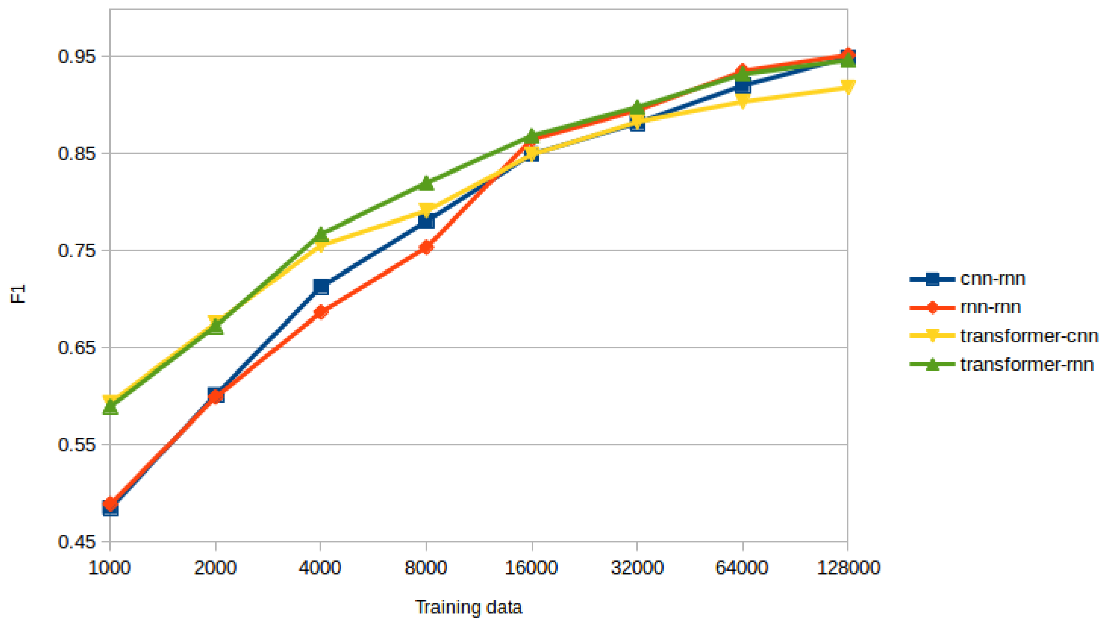


Figure 7: Effect of increasing the training size (Italian) for different encoder and decoders.

5.5. Error analysis

In order to understand the results and with the aim of improving them in future developments, we analyzed the errors manually. Table 6 exemplifies several error sources. This analysis was divided into two groups: i) document lines with a single DT and ii) lines with multiple DTs. These are a general source of error in both cases:

- Abbreviation: the standard form appears abbreviated and so the information is misunderstood. In some cases it carries an incorrect code and in others the addition of a new code.

- Spelling error: the spontaneous DT is misspelled and so the system obtains an incorrect code.

- Superclass: the system gives the spontaneous DT the code corresponding to its direct parent in the ICD-10-CM hierarchy (that means a more general code).

- Information inclusion/exclusion: the system does not identify part of the DT, so it generates a code mismatch. The same thing happens whenever the system considers more tokens for the code assignment.

However, other types of problems, like the omission of commas, or the variations in the use of coordination or certain prepositions only appear when DT lists are used, as this may cause alignment problems. In fact, the system performs better with the one-to-one (1:1) cases which are not prompted to alignment issues. This difference, as expected, was even bigger when using Levenshtein.

6. Conclusion

This work tackles medical record classification following the ICD-10 standard in a multilingual setting. The classification problem is hard for several reasons: 1) the gap between spontaneous and standard language; 2) a large-scale classification task, with thousands of possible classes and, 3) in real data, in most cases, there is no 1:1 alignment between DTs and ICD-10 codes.

We present and evaluate different neural network architectures for multi-class document classification as a sequence-to-sequence problem. The system is also able to learn to identify highly-predictive locations for each label, providing satisfactory explanations for its predictions. The system is also able

Error analysis with examples	
1:1	
Abbreviation	G: oap asphyxique I501 S: **oedeme aigu pulmonaire** asphyxique I501 R090
Multiword **Spelling error**	G: arret cardio-circulatoire I469 G: detresse respiratoire J960 S: detresse respiratoi**o**re R092
Superclass	G: acfa I489 S: acfa I48
Info. missing	G: plaie cardiaque **operatoire** Y600 S: plaie cardiaque S269
Info. inserted	G: operee Z924 S: **neoplasie analse** operee C169 Z924
Prep. inserted	G: syndrome glissement R453 S: symdrome **de** glissement R54
...	
1:N **All the cases in 1:1 and in addition the following:**	
Term unification **by comma omission**	G: deshydratation , demence type alzheimer , tumeur sein E86 G309 D486 S: deshydratation demence type alzheimer tumeur du sein E86 G309 C509
Term unification **by comma substitution**	G: hemorragie encephalique , hemorragie ventriculaire I619 I615 S: hemorragie encephalique **et** ventriculaire S062
...	

Table 6: Error cases detected (French). G refers to the standard form and S to the spontaneous form. The sequence of one or more DTs is followed by the corresponding ICD codes, eiteher manual or automatic.

to deal with real non-aligned data which is difficult for some other approaches, such as text similarity based models.

Our best model showed high-quality results, establishing a new state-of-the-art, and this fact opens a promising avenue for the task of automatically assigning ICD-10 codes to medical documents. The method is language independent, allowing efficient training, given only a set of annotated documents, and does not require complex feature engineering.

Acknowledgements

This work was supported by the Spanish Ministry of Science and Innovation (TUNER: TIN2015-65308-C5-1-R, PROSA-MED: TIN2016-77820-C3-1-R, DOMINO: PGC2018-102041-B-I00), and the Basque Government (MODENA: KK-2018/00087).
We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X Pascal GPU used for this research.

References

[1] D. Lang, Consultant Report-Natural Language Processing in the Health Care Industry, Cincinnati Children's Hospital Medical Center, Winter (2007).

[2] W. H. Organization, et al., International Classification of Diseases ICD 10 Classification of Mental and Behavioural Disorders: Clinical Descriptions and Diagnostic Guidelines, World Health Organisation, 1993.

[3] K. Cho, B. van Merrienboer, G. Gulcehre, F. Bougares, H. Schwenk, Y. Bengio, Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, in: Conference on Empirical Methods in Natural Language Processing (EMNLP) (2014).

[4] I. Sutskever, O. Vinyals, Q. Le, Sequence to Sequence Learning with Neural Networks, in: Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems (NIPS), pp. 3104–3112 (2014).

[5] G. Klein, Y. Kim, Y. Deng, J. Senellart, A. Rush, OpenNMT: Open-Source Toolkit for Neural Machine Translation, in: Proceedings of ACL 2017, System Demonstrations, Association for Computational Linguistics, Vancouver, Canada, 2017, pp. 67–72.

[6] R. Sennrich, O. Firat, K. Cho, A. Birch, B. Haddow, J. Hitschler, M. Junczys-Dowmunt, S. Läubli, A. V. Miceli Barone, J. Mokry, M. Nadejde, Nematus: a Toolkit for Neural Machine Translation, in: Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics, Association for Computational Linguistics, Valencia, Spain, 2017, pp. 65–68.

[7] F. Hieber, T. Domhan, M. Denkowski, D. Vilar, A. Sokolov, A. Clifton, M. Post, The SOCKEYE Neural Machine Translation Toolkit at AMTA 2018, in: Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers),

Association for Machine Translation in the Americas, 2018, pp. 200–207.

[8] J. P. Pestian, C. Brew, P. Matykiewicz, D. J. Hovermale, N. Johnson, K. B. Cohen, W. Duch, A Shared Task Involving Multi-label Classification of Clinical Free Text, in: Workshop on BioNLP'07: Biological, Translational, and Clinical Language Processing, 2007.

[9] Ö. Uzuner, B. R. South, S. Shen, S. L. DuVall, 2010 i2b2/VA challenge on concepts, assertions, and relations in clinical text, JAMIA 18 (2011) 552–556.

[10] R. Bossy, W. Golik, Z. Ratkovic, D. Valsamou, P. Bessières, C. Nédellec, Overview of the gene regulation network and the bacteria biotope tasks in BioNLP'13 shared task, BMC Bioinformatics 16 (2015) S1.

[11] A. Névéol, R. N. Anderson, K. B. Cohen, C. Grouin, T. Lavergne, G. Rey, A. Robert, C. Rondet, P. Zweigenbaum, 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.

[12] A. Névéol, A. Robert, F. Grippo, C. Morgand, C. Orsi, L. Pelikán, L. Ramadier, G. Rey, P. Zweigenbaum, 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, September (2018).

[13] G. Schmidt, ICD-10 coding using machine learning, `http://www.gregoryschmidt.ca/writing/icd-coding`, 2016.

[14] S. V. Pakhomov, J. D. Buntrock, C. G. Chute, Automating the Assignment of Diagnosis Codes to Patient Encounters Using Example-based and Machine Learning Techniques, Journal of the American Medical Informatics Association 13 (2006) 516–525.

[15] S. Boytcheva, I. Nikolova, E. Paskaleva, G. Angelova, D. Tcharaktchiev, N. Dimitrova, Structuring of status descriptions in hospital patient records, in: the Proceedings 2nd International Workshop on Building and Evaluating Resources for BioMedical Text Mining, associated to the 7th Int. Conf. on Language Resources and Evaluation (LREC-2010), ELRA, pp. 31–36.

[16] G. Karystianis, A. J. Nevado, C.-H. Kim, A. Dehghan, J. A. Keane, G. Nenadic, Automatic mining of symptom severity from psychiatric evaluation notes, in: International journal of methods in psychiatric research.

[17] R. Farkas, G. Szarvas, Automatic construction of rule-based ICD-9-CM coding systems, BMC bioinformatics 9 (2008) 1–9.

[18] A. Pérez, A. Atutxa, A. Casillas, S. A. Gojenola, K., Inferred joint multigram models for medical term normalization according to ICD, International Journal of Medical Informatics, **110**, pp. 111–117 (2018).

[19] R. A. Kavuluru A, L. Y., An empirical evaluation of supervised learning

approaches in assigning diagnosis codes to electronic medical records, Artif Intell Med. 65 (2015) 155–166.

[20] M. Almagro, R. Martínez, V. Fresno, S. Montalvo, Estudio preliminar de la anotación automática de códigos CIE-10 en informes de alta hospitalarios, Procesamiento del Lenguaje Natural 60 (2018) 45–52.

[21] B. Koopman, G. Zuccon, A. Nguyen, A. Bergheim, N. Grayson, Automatic ICD-10 classification of cancers from free-text death certificates, International Journal of Medical Informatics 84 (2015) 956 – 965.

[22] B. Koopman, S. Karimi, A. Nguyen, R. McGuire, D. Muscatello, M. Kemp, D. Truran, M. Zhang, S. Thackway, Automatic classification of diseases from free-text death certificates for real-time surveillance, BMC Medical Informatics and Decision Making 15 (2015) 53.

[23] K. Xu, M. Lam, J. Pang, X. Gao, C. Band, P. Mathur, F. Papay, A. K. Khanna, J. B. Cywinski, K. Maheshwari, P. Xie, E. Xing, Multimodal Machine Learning for Automated ICD Coding, CoRR abs/1810.13348 (2018).

[24] T. Young, D. Hazarika, S. Poria, E. Cambria, Recent Trends in Deep Learning Based Natural Language Processing, CoRR abs/1708.02709 (2017).

[25] R. Miotto, F. Wang, S. Wang, X. Jiang, J. T. Dudley, Deep learning for healthcare: review, opportunities and challenges, Briefings in bioinformatics 19 (2017) 1236–1246.

[26] J. Mullenbach, S. Wiegreffe, J. Duke, J. Sun, J. Eisenstein, Explainable Prediction of Medical Codes from Clinical Text, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), Association for Computational Linguistics, 2018, pp. 1101–1111.

[27] H. Shi, P. Xie, Z. Hu, M. Zhang, E. P. Xing, Towards automated ICD coding using deep learning, arXiv preprint arXiv:1711.04075 (2017).

[28] S. J. Pan, Q. Yang, A survey on transfer learning, IEEE Transactions on knowledge and data engineering 22 (2010) 1345–1359.

[29] M. Zeng, M. Li, Z. Fei, Y. Yu, Y. Pan, J. Wang, Automatic ICD-9 coding via deep transfer learning, Neurocomputing 324 (2019) 43–50.

[30] K. Liu, S. Peng, J. Wu, C. Zhai, H. Mamitsuka, S. Zhu, MeSHLabeler: improving the accuracy of large-scale MeSH indexing by integrating diverse evidence, Bioinformatics 31 (2015) i339–i347.

[31] M. Ebersbach, R. Herms, M. Eibl, Fusion Methods for ICD10 Code Classification of Death Certificates in Multilingual Corpora, in: CLEF 2017 Conference and Labs of the Evaluation Forum, Online Working Notes, CEUR-WS, September (2017).

[32] P. Zweigenbaum, T. Lavergne, Multiple Methods for Multi-class, Multi-label ICD-10 Coding of Multi-granularity, Multilingual Death Certificates, in: CLEF 2017 Conference and Labs of the Evaluation Forum, Online Working Notes, CEUR-WS, September (2017).

[33] Z. Miftahutdinov, E. Tutubalina, KFU at CLEF eHealth 2017 Task 1: ICD-10 Coding of English Death Certificates with Recurrent Neural Networks, in: CLEF 2017 Conference and Labs of the Evaluation Forum, Online Working Notes, CEUR-WS, September (2017).

[34] S. Cossin, V. Jouhet, F. Mougin, G. Diallo, F. Thiessard, IAM at CLEF eHealth 2018 : Concept Annotation and Coding in French Death Certificates, in: CLEF 2018 Evaluation Labs and Workshop: Online Working Notes, CEUR-WS, September (2018).

[35] M. Almagro, S. Montalvo, A. Diaz de Ilarraza, A. Pérez, MAMTRA-MED at CLEF eHealth 2018: A Combination of Information Retrieval Techniques and Neural Networks for ICD-10 Coding of Death Certificates, in: CLEF 2018 Evaluation Labs and Workshop: Online Working Notes, CEUR-WS, September (2018).

[36] J. Gobeill, P. Ruch, Instance-based Learning for ICD10 Categorization, in: CLEF 2018 Evaluation Labs and Workshop: Online Working Notes, CEUR-WS, September (2018).

[37] S. Jeblee, A. Budhkar, S. Milić, J. Pinto, C. Pou-Prom, K. Vishnubhotla, G. Hirst, F. Rudzicz, Toronto CL at the CLEF 2018 eHealth Challenge Task 1, in: CLEF 2018 Evaluation Labs and Workshop: Online Working Notes, CEUR-WS, September (2018).

[38] J. Ševa, M. Sänger, U. Leser, WBI at CLEF eHealth 2018 Task 1: Language-independent ICD-10 Coding using Multi-lingual Embeddings

and Recurrent Neural Networks, in: CLEF 2018 Evaluation Labs and Workshop: Online Working Notes, CEUR-WS, September (2018).

[39] F. Duarte, B. Martins, C. S. Pinto, M. J. Silva, Deep neural models for ICD-10 coding of death certificates and autopsy reports in free-text, Journal of biomedical informatics 80 (2018) 64–77.

[40] T. Lavergne, A. Névéol, A. Robert, C. Grouin, G. Rey, P. Zweigenbaum, Dataset for ICD-10 Coding of Death Certificates: Creation and Usage, in: Proceedings of the Fifth Workshop on Building and Evaluating Resources for Biomedical Text Mining (BioTxtM 2016).

[41] G. Pavillon, F. Laurent, Certification et codification des causes médicales de décès, Bull Epidemiol Hebd 30 (2003) 134–138.

[42] V. I. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals, in: Soviet physics doklady, volume 10, pp. 707–710.

[43] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, C. Dyer, Neural architectures for named entity recognition, arXiv preprint arXiv:1603.01360 (2016).

[44] X. Ma, E. Hovy, End-to-end sequence labeling via bi-directional lstm-cnns-crf, arXiv preprint arXiv:1603.01354 (2016).

[45] Y. Kim, Convolutional neural networks for sentence classification, arXiv preprint arXiv:1408.5882 (2014).

[46] M. Artetxe, G. Labaka, E. Agirre, K. Cho, Unsupervised neural machine translation, arXiv preprint arXiv:1710.11041 (2017).

[47] M. Neishi, J. Sakuma, S. Tohda, S. Ishiwatari, N. Yoshinaga, M. Toyoda, A bag of useful tricks for practical neural machine translation: Embedding layer initialization and large batch size, in: Proceedings of the 4th Workshop on Asian Translation (WAT2017), pp. 99–109.

[48] T. Mikolov, K. Chen, G. S. Corrado, J. Dean, Efficient Estimation of Word Representations in Vector Space, CoRR abs/1301.3781 (2013).

[49] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory, Neural computation 9 (1997) 1735–1780.

[50] J. Gehring, M. Auli, D. Grangier, D. Yarats, Y. Dauphin, Convolutional Sequence to Sequence Learning, in: Proceedings of the 34th International Conference on Machine Learning.

[51] J. Gehring, M. Auli, A novel approach to neural machine translation, `https://code.fb.com/ml-applications/a-novel-approach-to-neural-machine-translation/`, 2017.

[52] D. Bahdanau, K. Cho, Y. Bengio, Neural Machine Translation by Jointly Learning to Align and Translate, in: arXiv, arXiv:1409.0473.

[53] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, I. Polosukhin, Attention Is All You Need, in: Advances in Neural Information Processing Systems, pp. 6000–6010.

[54] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett

(Eds.), Advances in Neural Information Processing Systems 30, Curran Associates, Inc., 2017, pp. 5998–6008.

[55] J. Li, X. Chen, E. Hovy, D. Jurafsky, Visualizing and Understanding Neural Models in NLP, in: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, 2016, pp. 681–691.

[56] Y. Ding, Y. Liu, H. Luan, M. Sun, Visualizing and Understanding Neural Machine Translation, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, 2017, pp. 1150–1159.

[57] B. C. Kwon, M. Choi, J. T. Kim, E. Choi, Y. B. Kim, S. Kwon, J. Sun, J. Choo, RetainVis: Visual Analytics with Interpretable and Interactive Recurrent Neural Networks on Electronic Medical Records, CoRR abs/1805.10724 (2018).

[58] H. Strobelt, S. Gehrmann, M. Behrisch, A. Perer, H. Pfister, A. Rush, Seq2Seq-Vis: A Visual Debugging Tool for Sequence-to-Sequence Models, ArXiv e-prints (2018).

[59] I. Lopez-Gazpio, M. Maritxalar, A. Gonzalez-Agirre, G. Rigau, L. Uria, E. Agirre, Interpretable semantic textual similarity: Finding and explaining differences between sentences, Knowledge-Based Systems 119 (2017) 186 – 199.

7. Summary Points

What was already known on the topic?

- The problem presents a complex characterization due to non-standard language variation, spontaneous writing, large-scale multi-class classification or DT-ICD10 alignment issues.
- There are varied approaches, ranging from knowledge-based solutions to statistical and deep learning ones.
- Most Machine Learning approaches, although accurate, do not offer any helpful clue about the encoding decision besides the result itself.

What does this work add?

- Sequence-to-sequence deep learning approaches outperform other systems by a considerable margin for all languages.
- We have performed an exhaustive study of different sequence-to-sequence architectures, showing that there is no unique encoder-decoder combination that performs best for all languages, as we show important differences with respect to the neural network architecture employed.
- Apart from assigning the codes with good accuracy, the system also provides an interpretable result, aligning each code with its corresponding piece of text.