

End-to-End Neural Coder for Tumor Named Entity Recognition

Mohammed Jabreel^a

^a*Department of Computer Science, Hodeidah University, Hodeidah 1821, Yemen*

Abstract

This paper describes E2ENC, the system that we have developed to participate in CANTEMIST (CANCer TExt Mining Shared Task – tumor named entity recognition). E2ENC is a data-driven and end-to-end neural network-based system. It does not rely on external resources such as part-of-speech tagger. It proposes to solve two problems jointly; the first problem is to automatically extract the tumor morphology mentions that can be found in medical documents written in Spanish. The second task is to find the corresponding eCIE-O-3.1 codes for each extracted entity. E2ENC shows promising results, comparing to the baseline system. The reported results show that the proposed system achieve a micro-F1 of 84.9% and 77.7% on the test set for the first and second sub-tasks, respectively, and a MAP of 73.7% on the third sub-task.

Keywords

Information Extraction, Deep Learning, Text Mining, Medical Documents

1. Introduction

Recently, cancer has become one of the first causes of death in the Globe, surpassing cardiovascular diseases. To perform research and improve standards of healthcare and to evaluate cancer treatment outcomes easy—and ideally, in an automated way—access to a variety of data sources is required. The knowledge embedded in unstructured textual documents (e.g., pathology reports, clinical notes), is crucial to achieve all these goals. Hence, there is an imperative need to take advantage of natural language processing (NLP) and text mining technologies to develop Information Extraction (IE) systems that can automatically process unstructured medical resources, i.e., pathology reports and clinical notes, and extract critical information that leads to better clinical decision-making.

There are multiple approaches to building such IE systems. In general, such systems leverage NLP tools, such as tokenizers, part-of-speech taggers and parsers to pre-process the documents. After that, modules, that may be rule-based or machine learning-based, are designed, to solve IE related tasks.

Starting by a seed collection of entities, the idea of rule-based systems is to manually engineer some rules based on regular expressions, syntactic, or dependency structures to expand the collection iteratively [1, 2, 3].

Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2020)


EMAIL: mhjabreel@gmail.com (M. Jabreel)

URL: <https://mhjabreel.com> (M. Jabreel)

ORCID: 0000-0001-9774-9066 (M. Jabreel)



© 2020 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

The feature-engineering-based systems aim to train a sequence tagger with rich, hand-crafted features based on linguistic or syntactic information from annotated corpus to predict a label (e.g., *O*, *B-* < *entity* > or *I-* < *entity* >) on each token in a sentence [4].

Rule-based and feature-engineering-based approaches are labor-intensive for constructing rules or features using linguistic and syntactic information. Despite some promising results, there are two main issues with these approaches. First, the engineering of rules and features is a time-consuming task. Moreover, rules always need to be updated. Second, the systems of these two categories are dependent on some external requirements like a parser analyzing the syntactic and dependency structure of sentences. Therefore, the performances of these systems rely on the quality of the parsing results [5, 2]. To avoid these issues, deep-learning is used to develop systems learn high-level representations for each token, on which a classifier or sequence tagger can be trained [6, 7, 8].

CANTEMIST (CANcer TExt Mining Shared Task – tumor named entity recognition) [9] is the first shared task specifically focusing on named entity recognition of a critical type of concept related to cancer, namely tumor morphology, in the Spanish language. There are three structured sub-tasks: "NER", "NORM" and "CODING". The first sub-task aims to identify automatically tumor morphology mentions. All tumor morphology mentions are defined by their corresponding character offsets in UTF-8 plain text medical documents. The second sub-task aims at returning all tumor morphology entity mentions together with their corresponding eCIE-O-3.1 codes i.e. finding and normalizing tumor morphology mentions. The goal of the third sub-task is to returning for each document a ranked list of its corresponding ICD-O-3 codes (Spanish version: eCIE-O-3.1).

We participated in the CANTEMIST challenge by developing E2ENC, an end-to-end neural coder system for the three sub-tasks. The proposed system provides an end-to-end solution and does not require any parsers or other linguistic resources. Specifically, the proposed system is a multilayer neural network, where the first seven layers aim to learn high representation for a sequence of tokens, then we pass, jointly, the output of these layers to four Conditional Random Field (CRF) models that are learned jointly. One is for extracting the tumor morphology mentions, while the others is for identifying their ICD-O-3 codes.

The rest of the paper structured as follows: Section 2 presents the Methodology; Section 3 explains the experimental settings and discusses the results; finally, Section 4 concludes this paper.

2. System Description

In this section, we introduce our system, i.e., E2ENC, and its implementation steps in detail. We first describe the problem and the corresponding sub-tasks. Given a clinical document written in Spanish, our goal is to build a system that can i) automatically extract tumor morphology mentions. The output of this step is the start and end positions of the text spans that represent the character corresponding offsets of the tumor morphology mentions. ii) Normalising the extracted entities by returning all tumor morphology entity mentions together with their corresponding eCIE-O-3.1 codes. iii) Finally, given the list of eCIE-O-3.1 codes extracted from the document, find a ranked list of ICD-O-3 codes (Spanish version: eCIE-O-3.1).

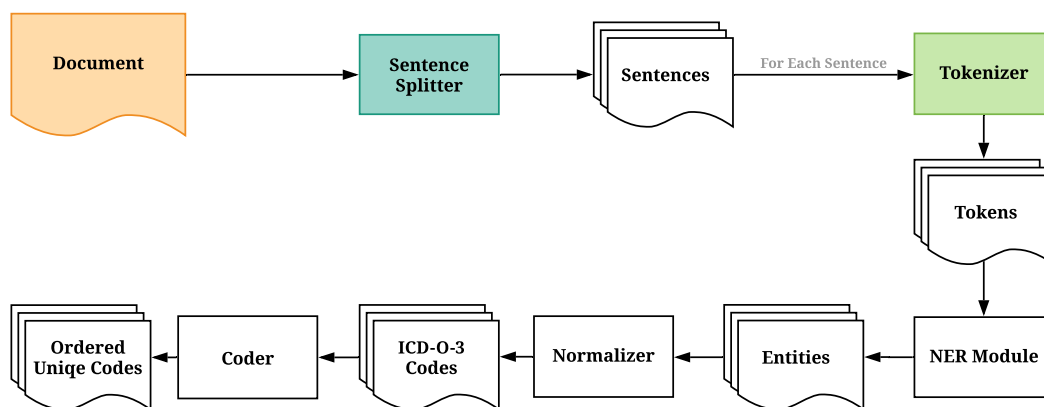


Figure 1: The overall architecture of the E2ENC system.

We propose a deep learning-based system to solve the tasks defined above. Figure 1 depicts the overall architecture of the system. First, we use a sentence splitter to find all the sentences in the document. After that, we tokenize each sentence and find the list of tokens. As soon as we get the tokens, we pass them into the Name-Entity-Recognizer (NER) model to extract the tumor morphology entity mentions. Then, we use a Normalizer (NORM) model to find the code of each extracted tumor morphology. Finally, we take the set of the codes, i.e., removing the duplicates, and use that set as an output for the the third task.

Our system is based on a deep learning-based model that solves the first two tasks, i.e., NER and NORM, jointly. The idea is to regard the problem as a multi-label multi-tags token classification. Given a sequence of tokens $W = \{w_1, w_2, \dots, w_n\}$, the model must produce for each token $w_i, 1 \leq i \leq n$ a vector of labels $\mathcal{L}_i = \{l_{1j}, l_{2k}, \dots, l_{mr}\}$. Where n refers to the sequence length, i.e., the number of tokens and m is the number of labelling tasks. The symbol l_{tj} is one of a set of tags in BIOES tagging schema. That is $l_{tj} \in \{B - label, I - label, O - label, E - label, S - label\}$, where *label* is one of the set of labels in the task t .

In our case, the number of labelling tasks, i.e., m , is four. The first task is the NER task and has only one label that is "MORFOLOGIA_NEOPLASIA". The rest three tasks are related to the NORM task. Considering that the format of eCIE-O-3.1 codes which is $\backslash d\{4,\} / \backslash d\{1,\} / [H]$ and by splitting the code by the "/" symbol, we decompose the NORM task into three tagging tasks. Each is corresponding to produce a part of the the code.

We designed a deep learning-based model, shown in Figure 2 that is composed by eight consecutive layers to jointly solve the four tagging tasks. The following subsections explain in details the model and the training procedure that we followed to train the model.

First, we start by explaining the Embedding layer in Subsection 2.1. After that, in Subsection 2.2, we describe the structure of the encoder layer. Subsection 2.3 explains the output layer. Finally, in Subsection 2.4, we define the objective function and detail the training routine.

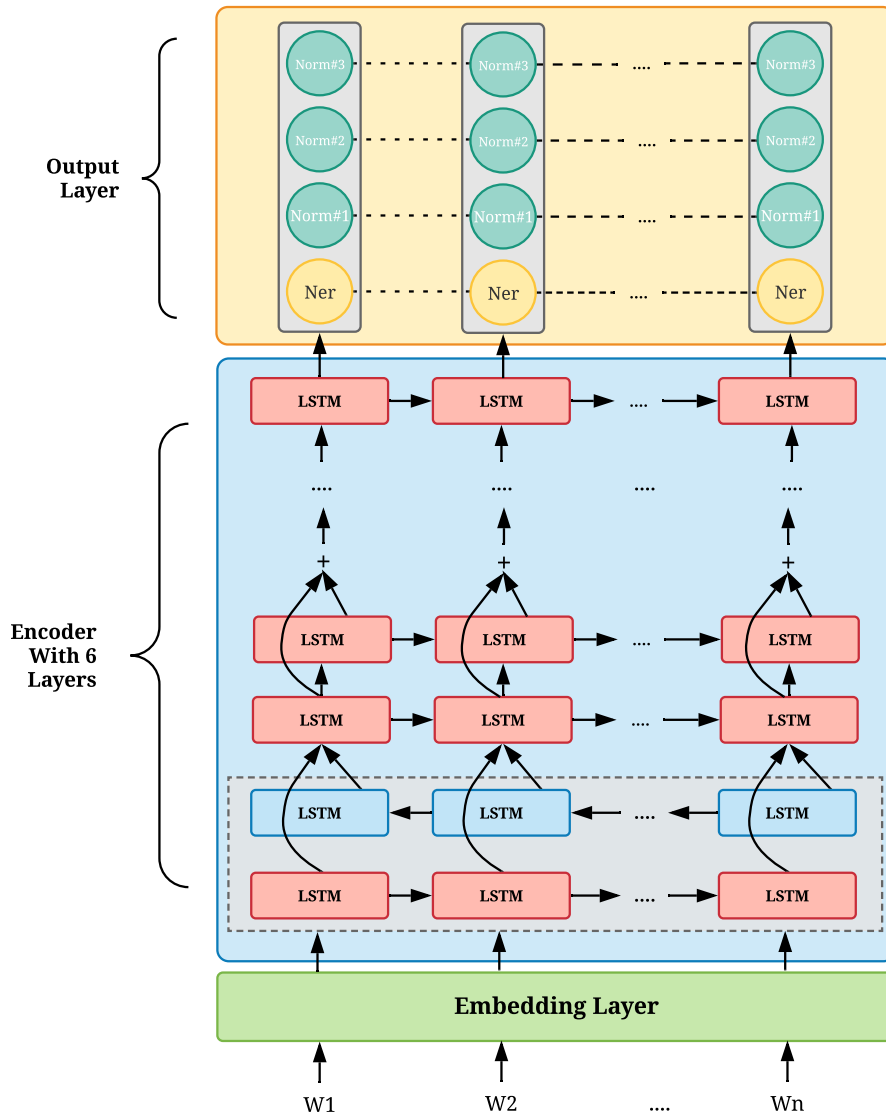


Figure 2: The architecture of E2ENC system.

2.1. Embedding Layer

The goal of the embedding layer is to represent each word $w_i \in W$ by a low-dimensional vector space $v_i \in \mathbb{R}^d$. Here, d , which is 400d in our case, is the size of the embedding layer. As shown in Figure 3, we use two levels of embedding: word-level and character-level. For the word-level embedding, we replace w_i with its pre-trained word embedding vector v_i^w [10]. We use a single-layer 1-Dimensional Convolutional Neural Networks (Conv1D) with max-over-time

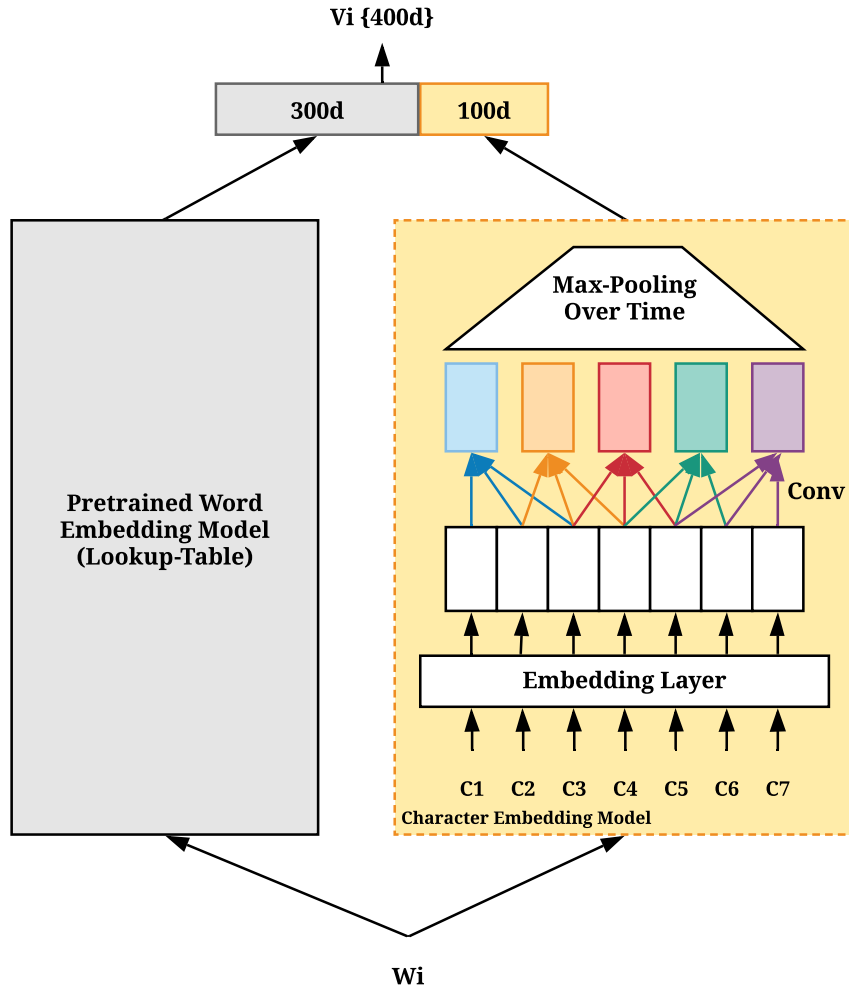


Figure 3: The embedding layer. The solid Gray part is none-trainable.

pooling to represent the word at character-level as the following. Suppose that w_i is made up of a sequence of characters $[c_1, c_2, \dots, c_N]$, where N is the length of w_i . First, we pass the sequence of characters of the word w_i to a randomly initialized character embedding layer to get the matrix $C_i \in \mathbb{R}^{r \times d^c}$ - that is the character-level representation of w_i . Here, the j -th column corresponds to the character embedding for c_j . After that, we apply a narrow convolution between C_i and a filter (or kernel) $K \in \mathbb{R}^{r \times k}$ of width k , after which we add a bias and apply a nonlinearity to obtain a feature map $f^i \in \mathbb{R}^{n-k+1}$. Specifically, the m -th element of f^i is given by:

$$f^i[m] = \tanh(\langle C_i[*, m : m + k - 1], K \rangle + b) \quad (1)$$

where $C_i[* , m : m + k - 1]$ is the m -to- $(m + w - 1)$ -th column of C_i and $\langle A, B \rangle$ is the frobenius inner product. Finally, we take the max-over-time

$$v_i^c = \max_m f^i[m] \quad (2)$$

as the feature corresponding to the filter K (when applied to word w_i). A filter is basically picking out a character n -gram, where the size of the n -gram corresponds to the filter width.

The final representation of the word w_i is given by concatenating the word-level vector and the character-level vector.

$$v_i = [v_i^w; v_i^c] \quad (3)$$

The embedding layer is the first layer in xx model. It takes as an input a word w_i and produces a vector v_i that is regarded as the representation of that word.

2.2. Encoder Layer

The main objective of the encoder is to transform the sequence of vectors v_1, v_2, \dots, v_n produced by the embedding layer into a sequence of recurrent states h_1, h_2, \dots, h_n in a high level of representation. The encoder of our system is inspired by the encoder in the translation system GNMT system from google [11].

It is composed by six Long Short-Term Memory (LSTM) [12] recurrent neural layers. The first encoder layer is bi-directional in which the blue nodes process the information from left to right while the red nodes gather information from right to left. The other layers of the encoder are uni-directional. Residual connections start from the layer third from the bottom in the encoder.

2.3. Output Layer

The output layer receives as an input the sequence of states h_1, h_2, \dots, h_n obtained from the encoder and send pass them in parallel to four fully-connected layers each is responsible to solve a single tagging problem as mentioned above. It has been shown that in such tasks, i.e., sequence labelling, it is beneficial to consider the correlation between labels in neighbourhoods, specifically when there are strong dependencies across the output labels. For example, the tag **B** is more likely to be followed by the tag **I**. Thus, instead of modeling tagging decisions independently, we model them jointly using a Conditional Random Field (CRF) [13].

Formally, let $H = \{h_1, h_2, \dots, h_n\}$ be the sequence of vectors to be labeled, which is produced by the encoder layer, and $Y_t = \{y_1, y_2, \dots, y_n\}$ is the corresponding tag sequence for the task t . Each element y_i of Y_t is one of the *B - label*, *I - label*, *O - label*, *E - label* or *S - label* tags. Both H and L_t are assumed to be random variables and they are jointly modeled. The entire model can be represented as an undirected graph $G = (V, E)$ with cliques C .

In this work we employed a linear-chain CRF, where G is a simple chain or line: $G = (V = \{1, 2, \dots, n\}, E = \{(i, i + 1)\})$. It has two different cliques (i.e. $C = \{P, M\}$): a unary clique (P) representing the input-output connection, and a pairwise clique (M) representing the adjacent output connection. We consider P to be the matrix of output scores, where $P_{i,j}$ corresponds to the score of the j^{th} tag of the i^{th} word in a sentence and it is computed as follows:

$$P_{i,j} = W_{i,j}h_i + b_j ; j = 1, 2, 3 \quad (4)$$

In this equation, the parameters are $W_{i,j} \in \mathbb{R}^{2 \times d_h}$ and $b_j \in \mathbb{R}^1$, where d_h is the dimensionality size of the hidden state.

The clique M is considered to be the matrix of transition scores such that $M_{i,j}$ represents the score of a transition from the tag i to the tag j . Given that, we define the score function of the sequence of predictions as follows:

$$s(H, Y_t) = \sum_{i=1}^n P_{i,y_i} + \sum_{i=0}^n M_{y_i,y_{i+1}} \quad (5)$$

In this expression y_0 and y_{n+1} denote the start and the end tags of the sentence, that we add to the set of possible tags.

A softmax over all possible tag sequences (Y_t^*) on a sequence H yields a probability for the sequence Y_t as follows:

$$p(Y_t|H) = \frac{e^{s(H,Y_t)}}{\sum_{\tilde{y} \in Y_t^*} e^{s(H,\tilde{y})}} \quad (6)$$

During training, we minimize the negative log-probability of the correct tag sequence:

$$J_t = -\log(p(Y_t|H)) = -s(H, Y_t) + \log\left(\sum_{\tilde{y} \in Y_t^*} e^{s(H,\tilde{y})}\right) \quad (7)$$

During inference, we search for the output sequence \widehat{Y}_t that obtains the highest probability given by:

$$\widehat{Y}_t = \arg \max_{\tilde{Y}_t \in Y_t^*} p(\tilde{Y}_t|H) \quad (8)$$

In this model, Eq. 7 and Eq. 8 can be solved efficiently using dynamic programming.

2.4. Training Procedure

Lets t refers to one of the tagging tasks T and J_t the objective function of that task computed by Eq. 7, we define the joint objective function as the following:

$$J = \frac{1}{|T|} \sum_{t \in T} J_t \quad (9)$$

In this expression $|T|$ means the number of tasks.

Table 1

The number of each document in the datasets.

#	Split	Num. of Documents
1	Train	501
2	Dev1	250
3	Dev2	250
4	Test	300

The derivative of the objective function J , Eq. 9, is taken through back-propagation with respect to the whole set of parameters of the model, which are the transition matrix M , the parameters of the BiGRU model and the parameters of the matrix P defined in Eq. 4. The parameters are optimized using Adam optimizer with a learning rate of 0.0001. To reduce the effects of gradient exploding, we set the clipping threshold of the gradient to 5. We apply a dropout [14] between the embedding layer and the recurrent layer with probability of 0.5 to prevent over-fitting.

3. Experiments and Results

In this section, we discuss the dataset used and different experimental settings devised to evaluate our system.

3.1. Datasets

We trained and fine-tuned our system respectively on the training and the development sets provided by the organizers of the CANTEMIST challenge. The statistical description of the datasets is shown in Table 1. After that, we submitted the predicted labels of the test set that are produced by our system to evaluate its performance. The organizers omitted the golden labels of the test for the tree tasks. We used the train split to train the model, the Dev1 split to fine-tune the system and find the best parameters and Dev2 for testing the system performance locally.

3.2. Evaluation Metrics

The official evaluation metrics used to validate the performance of the system are: Precision, Recall and F1-score for Cantemist-NORM and Cantemist-NER. For the Catemist-CODING Mean Average Precision (mAP) was used.

3.3. Results

Table 2 shows the results of our submitted system on the Test and the Dev2 sets. It also shows the results of the baseline system on the Test set, which is simply a dictionary lookup based on Levenshtein distance. It looks for train and development annotations in the test set. From the reported results, we can note that in general, our system outperforms the baseline system and

Table 2

The Performances of our system on the test set compared to the performance on the Dev2 set, and the baseline system.

System	Sub-Task 1 (NER)			Sub-Task 2 (NORM)			Sub-Task 3 (CODING)
	P	R	F1	P	R	F1	mAP
Dev2 (Our System)	84.7	85.2	84.9	77.7	78.2	77.9	75.0
Test (Our System)	83.7	84	83.9	77.5	77.9	77.7	73.7
Test (Baseline)	18.1	73.3	29.1	18.0	73.0	28.8	58.4

gives comparable performance on both the Dev2 and the Test sets. The maximum difference in the performance of the system on the Dev2 test and the Test set is recorded for the Recall metric (with 1% in case of the NER task and 0.3% in case of the NORM task). In terms of the CODING task, the difference is 1.3%. Another remarkable observation is that our system gives a similar performance in all the evaluation metrics, which shows its consistency. It is also clearly shown that the performance of the system in the NORM task affects the outcome in the CODING task (as we only take the distinct codes as the final ranking result for the CODING task). To overcome this issue, we plan to develop a ranking model that takes as an input the extracted entities and the set of coding as a candidate set of codes and produce a ranked set of codes. We leave this extension for the future work.

4. Conclusion

In this paper, we describe in detail the implementation of End2End Neural Coder (E2ENC) system, including all the techniques that are crucial to its accuracy, and robustness. The system is used to solve the tasks defined in CANTEMIST (CANcer TEXT Mining Shared Task – tumor named entity recognition) shared tasks. E2ENC contains four sub-models that were trained jointly. The first one aims to automatically extract tumor morphology mentions in health reports written in Spanish. The three sub-models are used to perform the normalization task and return all tumor morphology entity mentions with their corresponding eCIE-O-3.1 codes. E2ENC provides an end-to-end solution and does not require any external tools or other linguistic resources. The effectiveness of the proposed system has been evaluated by participating in the CANTEMIST shared task. The reported results show that the proposed system is stable and consistent. In our future work, we plan to perform extensive error analysis and inspect the performance of the system and improve it. For example, we plan to use a transformer-based interpretable model like BERT [15] as a pre-trained embedding model instead of using the Embedding layer.

References

- [1] L. Sweeney, Replacing personally-identifying information in medical records, the scrub system., in: Proceedings of the AMIA annual fall symposium, American Medical Informatics Association, 1996, p. 333.

- [2] I. Neamatullah, M. M. Douglass, H. L. Li-wei, A. Reisner, M. Villarroel, W. J. Long, P. Szolovits, G. B. Moody, R. G. Mark, G. D. Clifford, Automated de-identification of free-text medical records, *BMC medical informatics and decision making* 8 (2008) 32.
- [3] A. Coden, G. Savova, I. Sominsky, M. Tanenblatt, J. Masanz, K. Schuler, J. Cooper, W. Guan, P. C. De Groen, Automatically extracting cancer disease characteristics from pathology reports into a disease knowledge representation model, *Journal of biomedical informatics* 42 (2009) 937–949.
- [4] Z. Liu, Y. Chen, B. Tang, X. Wang, Q. Chen, H. Li, J. Wang, Q. Deng, S. Zhu, Automatic de-identification of electronic medical records using token-level and character-level conditional random fields, *Journal of biomedical informatics* 58 (2015) S47–S52.
- [5] Ö. Uzuner, Y. Luo, P. Szolovits, Evaluating the state-of-the-art in automatic de-identification, *Journal of the American Medical Informatics Association* 14 (2007) 550–563.
- [6] Z. Liu, B. Tang, X. Wang, Q. Chen, De-identification of clinical notes via recurrent neural network and conditional random field, *Journal of biomedical informatics* 75 (2017) S34–S42.
- [7] M. Jabreel, F. Hassan, A. Moreno, Target-dependent sentiment analysis of tweets using bidirectional gated recurrent neural networks, in: *Advances in hybridization of intelligent methods*, Springer, 2018, pp. 39–55.
- [8] M. Jabreel, F. Hassan, D. Sánchez, J. Domingo-Ferrer, A. Moreno, E2ej: Anonymization of spanish medical records using end-to-end joint neural networks., 2019.
- [9] A. Miranda-Escalada, E. Farré, M. Krallinger, Named entity recognition, concept normalization and clinical coding: Overview of the cantemist track for cancer text mining in spanish, corpus, guidelines, methods and results, in: *Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2020)*, CEUR Workshop Proceedings, 2020.
- [10] J. Pennington, R. Socher, C. Manning, Glove: Global vectors for word representation, in: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.
- [11] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al., Google’s neural machine translation system: Bridging the gap between human and machine translation, *arXiv preprint arXiv:1609.08144* (2016).
- [12] S. Hochreiter, J. Schmidhuber, Long short-term memory, *Neural computation* 9 (1997) 1735–1780.
- [13] J. Lafferty, A. McCallum, F. C. Pereira, Conditional random fields: Probabilistic models for segmenting and labeling sequence data (2001).
- [14] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R. R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, *arXiv preprint arXiv:1207.0580* (2012).
- [15] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, *arXiv preprint arXiv:1810.04805* (2018).