POLITECNICO
DI TORINO

Master degree course in Computer Engineering

Master Degree Thesis

Machine Reading of Clinical Notes for Automated ICD Coding

Supervisor
Prof. Maurizio Morisio

Candidate
Stefano Malacrinò

Internship tutors
Dr. Giuseppe Rizzo (LINKS Foundation)
Dr. Carmelo Velardo (University of Oxford)

Academic Year 2018 - 2019

Abstract

A common task in processing of medical records is the assignment of standardized codes for diagnoses and procedures to free-text documents such as clinical notes. With the rising popularity of electronic health records (EHRs) systems for the automated reading of the clinical notes and codes assignment have been proposed in the recent years. This is a hard natural language processing task that requires parsing long documents written in a domain-specific language and selecting a set of appropriate codes among the thousands existent.

We present a hierarhical neural model for automated coding of clinical notes composed by an embedding layer, a recurrent layer and a convolutional layer, with attentional layers at word and sentence level. The sentence level attentional matching is computed using embedded representation of the codes based on their textual descriptions.

Our proposed model outperforms the previous state of the art model (Mullenbach *et al.* , 2018) according to several quantitative metrics. We provide a qualitative evaluation of the results, which provides some insight on what parts of the text are identified as most relevant for each code by the attention mechanism. Finally through this evaluation we show the strengths and the limits of this family of models and we provide suggestions for future improvements based on our observations.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The International Classification of Diseases (ICD) is a healthcare classification system maintained by the World Health Organization. It provides a set of diagnostic codes for classifying diseases (including a wide range of signs, symptoms and external causes of injury or disease) and surgical procedures. ICD codes are used for a number of different tasks, such as reporting health conditions and carrying out medical billing.

The standard coding procedure consists in assigning one or more ICD codes to a patient's hospital visit: this operation is performed by medical coders, who read and review the clinical notes written by physicians and then assign the appropriate ICD codes according to the coding guidelines. This process can be time consuming and error-prone.

With the rising popularity of electronic health records (EHRs) systems for the automated reading of the clinical notes and codes assignment have been proposed in the recent years.

In this work we present a hierarhical neural model for automated coding of clinical notes. The architecture is composed by an embedding layer, a recurrent layer and a convolutional layer, with attentional layers at word and sentence level. The sentence level attentional matching is computed using embedded representation of the codes based on their textual descriptions.

Our proposed model outperforms the previous state of the art model (Mullenbach *et al.* , 2018) according to several quantitative metrics. We provide a qualitative evaluation of the results by examining the n-grams highlighted by the attention mechanism for each label, which provide some insight on what parts of the text are most relevant for each code. Finally through this evaluation we show the strengths and the limits of this family of models and we provide suggestions for future improvements based on our observations.

Chapter 2

ICD coding system

In order to assign the correct ICD codes to a patient's hospital stay medical coders must read the clinical notes written by physicians and then select the appropriate codes according to the guidelines. Such guidelines differ from country to country. In this study we refer to the guidelines dictated by the United States government for ICD-9-CM, a related classification system based on the ICD which provides additional morbidity detail. ICD9-CM was updated annually until 2015, when it was officially deprecated in favor of its successor ICD-10.

There are 3 volumes of ICD-9 published each year. Volume 1 is a tabular index with a numerical list of disease code numbers, Volume 2 is an alphabetical index to diseases entries, Volume 3 provides an alphabetic index and a tabular list for surgical, diagnostic and therapeutic procedures. Furthermore there are five appendices to Volume 1: Appendix A – Morphology of Neoplasms, Appendix B – Glossary of Mental Disorders, Appendix C – Classification of Drugs, Appendix D – Classification of Industrial Accidents, Appendix E – Listing of 3-digit codes.

ICD-9-CM diagnosis codes are composed of 3, 4, or 5 characters. The three-characters codes are included in ICD-9-CM as headings of a category of codes that might be further subdivided by the use of a fourth and/or fifth digits, which provided greater detail (E codes have four-characters headings, subdivided by a fifth character). The coarser codes must be used only when no further subdivision is present, otherwise where fourth-digit sub-categories or fifth-digit sub-classifications are provided, they must be assigned.

Procedure codes are organized in a similar way, but are composed of either 3 or 4 characters. The two-digit codes are the headings of a category of codes that might be further subdivided by the use of third and fourth characters, which provide greater detail. As for diagnosis codes, the most specific procedure codes available must be assigned.

Given the ICD-9-CM structure the codes headings can be further aggregated into groups of codes in order to form a hierarchy tree, where top level categories

refer to generic concepts and lower level codes indicate specific conditions or procedures. Information on the hierarchy is retrieved from an ontology available on NCBO BioPortal [1]. The codes contained in the ontology are ICD-9-CM version 2012.

The hierarchy is based on semantics (codes related to similar concepts are grouped together), but the granularity of these groupings varies from case to case. For example at level 3 diseases are grouped according as intervals of code headers (e.g. 001-009), while procedures are not grouped and nodes represent single codes of 3 characters. Moreover, the hierarchy tree is imbalanced: considering the "Diseases and injuries" node, some of its children have an additional level of granularity with respect to their siblings. A sample of the hierarchy tree is shown in Figure2.1.

To assign the codes for a patient, a coder must first identify the reason for the patient's visit by looking at signs, symptoms, diagnosis and conditions reported in the physician's notes. Conditions that are referred to as "possible" must not be coded and only defined symptoms and illnesses must be taken into account. One the reason for a patient's visit is identified, Volume 2 index is used to look up the relevant terms. Once the appropriate code in the Volume 2 is identified then Volume 1 is used to review that code. In particular, Volume 2 provides only generic codes, Volume 1 must be used to select the related code with the highest level of specificity. Volume 1 provides also additional coding instructions specific to a code, such as precedence ("code first") or co-occurrence ("code also"). For example, for a patient with severe sepsis, code 995.92 states "code first the underlying infection".

level	# nodes
0	1
1	4
2	73
3	526
4	3585
5	10038
6	7313
7	867

Table 2.1: Number of codes by level in the ICD-9-CM hierarchy

[1] https://bioportal.bioontology.org/ontologies/ICD9CM

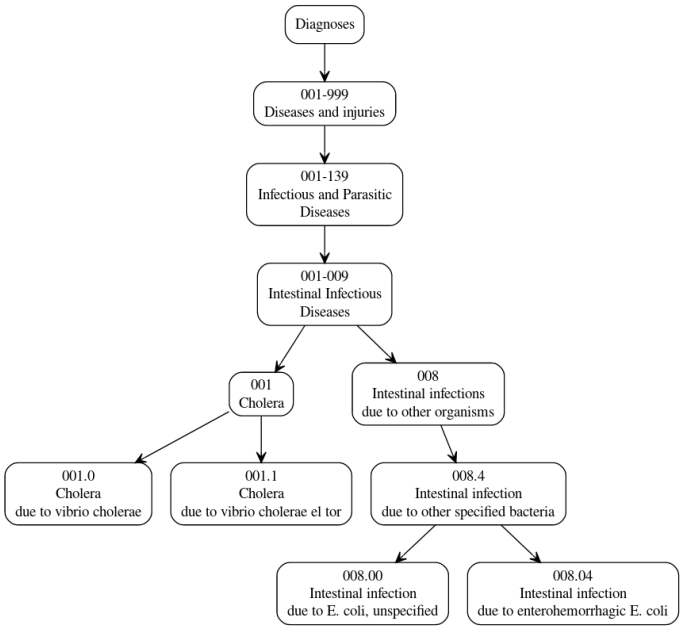


Figure 2.1: A sample from the ICD-9-CM hierarchy tree

Chapter 3

Natural language processing for document classification

Document classification is a natural language processing task whose purpose is to automatically classify text documents into one or more defined categories. Some examples of text classification are audience sentiment analysis from social media, spam detection, categorization of news articles into topics.

Text Classification is an example of supervised machine learning task since a labelled dataset containing text documents and their labels is used for train a classifier.

An end-to-end text classification pipeline consists of the following stages:

- Document preprocessing and tokenization
- Document vectorization
- Model training
- Model validation

In the following sections we present the main text vectorization approaches and the most common models used for document classification.

3.1 Text vectorization

Machine learning algorithms take numeric feature vectors as input. Thus, when working with text documents, each document must be transformed into a numeric vector. This process is known as text vectorization and can be performed using different techniques.

3.1.1 Bag-of-words models

In the bag-of-words (BOW) vectorization approach, every document in the corpus is represented as a vector whose length is equal to the vocabulary of the corpus. The value of each element of the document vector is computed according to the BOW model chosen. In the next sections we will show the most common models of this family, each of which extends or modifies the base model to describe semantic space in a different way.

We will use the following dictionary for our examples.

0	*and*	1000000
1	*bird*	0100000
2	*cat*	0010000
3	*dog*	0001000
4	*eats*	0000100
5	*play*	0000010
6	*the*	0000001

Frequency vector

The simplest encoding model consists in filling the vector index corresponding to each token in the vocabulary with the frequency of that word in the document.

$$the \quad cat \quad and \quad the \quad dog \quad play \implies 1011012$$

Frequency vectors represent similarity at the document level, but information on word order is lost and since all word vectors are orthogonal they don't provide any information on similarity between words.

One-Hot Encoding

Because frequency-based encoding methods disregard grammar and word order in documents, they suffer from the long tail, or Zipfian, distribution: tokens that occur very frequently are orders of magnitude greater than the less frequent ones. This can have a significant impact on models that expect normally distributed features.

A solution to this problem is one-hot encoding, which represents a document as a binary vector: the index of this vector corresponding to each token in the vocabulary is set to a value that indicates either the presence (1) or absence (0) of that token in the document.

An example of one-hot encoding is shown below.

$$the \quad cat \quad and \quad the \quad dog \quad play \implies 1011011$$

One-hot encoding reduces the imbalance in the tokens distribution and is most effective for very small documents that don't contain very many repeated elements.

TF-IDF Transform

The bag-of-words models listed above only describe a document without taking into account the context of the whole corpus. TF-IDF considers the relative frequency of tokens in the document against their frequency in other documents of the corpus. The intuition is that terms that are frequent in a document but rare in the corpus are more meaningful than the ones that are frequent across the whole corpus.

TF–IDF (term frequency–inverse document frequency) encoding normalizes the frequency of tokens in a document with respect to the their frequency in the rest of the corpus. This approach puts emphasis on tokens that are relevant to a specific instance of the corpus.

TF–IDF is computed on per-token, such that the relevance of a token to a document is measured by the scaled frequency of the appearance of the term in the document, normalized by the inverse of the scaled of its frequency in the entire corpus.

TF-IDF is the product of two statistics, term frequency tf and inverse document frequency idf. The term frequency of a token t given a document d, $tf(t, d)$, can be the boolean frequency (as in one-hot encoding, 1 if t is present in d, 0 otherwise), or the raw count. Generally term frequency is scaled logarithmically to prevent bias caused by longer documents or terms that appear much more frequently with respect to other terms: $tf(t, d) = 1 + \log(f_{t,d})$.

Similarly, the inverse document frequency of a term given the set of documents D is logarithmically scaled and is computed as $idf(t, D) = \log 1 + \frac{N}{n_t}$, where N is the number of documents and n_t is the number of occurrences of the term t in the corpus. TF–IDF is then computed as

$$tf - idf(t, d, D) = tf(t, d) \cdot idf(t, D) \tag{3.1}$$

An example of one-hot encoding is shown below.

the	*cat*	*and*	*the*	*dog*	*play*	$\Longrightarrow$	0.42	0.00	0.30	0.42	0.00	0.42	0.60
the	*cat*	*eats*	*the*	*bird*			0.00	0.47	0.33	0.00	0.47	0.00	0.67

The TF–IDF score is always in the range [0,1]. The closer the TF–IDF score of a token is to 1, the more informative that term is to that document. The closer the score to zero, the less informative the token.

3.1.2 Distributed models

Bag-of-words models have a very high dimensionality, resulting in an extremely sparse space. Word order, grammar, and other structural features are not represented. When using these encodings no meaningful comparison can be computed

between word vectors other than equality testing.

The distributed representation models overcome these issues by training a model that maps each word of the vocabulary to a vector of fixed size: each vector identifies a point in the representation space, allowing to use to use vector arithmetic on words to determine their semantic properties and relationships. Formally, for an arbitrary word w in the dictionary D a real number vector e_w with fixed size d_e is assigned, called the word vector of w.

In order to deduce the meaning of words, their use in language can be analyzed: this is a fundamental idea of distributional semantics called the "distributional hypothesis". This intuition was first captured by John Rupert Firth, an English linguist working on language patterns in the 1950s, who said:

> You shall know a word by the company it keeps

Firth was referring to the principle that the meaning of a word is captured by its use with the surrounding words, that is to say the context for any word is useful to define the meaning of that word. This is the concept behind many models for learning word vectors, also called word embeddings.

As shown in Figure3.1 when word vectors are trained with distributed models similar words converge to similar locations in the representation space. Similarity between two word vectors can be defined as Euclidean distance (the actual distance between points in space), or cosine similarity (the angle between two vectors in space).

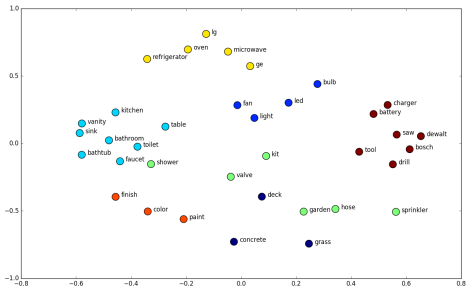


Figure 3.1: Natural clustering exhibited by word vectors

A direct effect of this property is that word vectors can effectively capture the semantic relationships of the words. For instance, words that are synonyms tend to have similar vectors in terms of cosine similarity and antonyms tend to have dissimilar vectors. Furthermore, word vectors tend to obey the laws of analogy. For example, for the analogy "woman is to queen as man is to king" the following equality holds

$$v_{man} - v_{woman} \approx v_{king} - v_{queen}$$

where v_{man}, v_{woman}, v_{king} and v_{queen} are the word vectors for man, woman, king and queen respectively. These observations strongly suggest that word vectors encode valuable semantic information about the words that they represent.

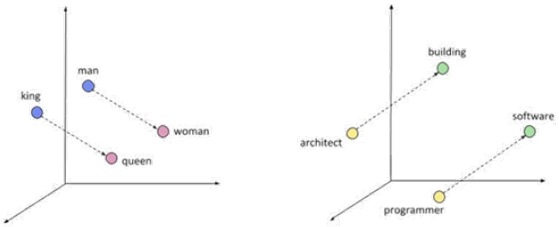


Figure 3.2: Analogies representation with word vectors

In the following sections we illustrate two of the main models used to train word embeddings: Word2ved and GloVe.

Word2vec

Word2Vec, introduced by Mikolov *et al.* (2013), is based on a shallow, two-layer neural network that is trained to reconstruct linguistic contexts of words. After the training the weight matrices of the network are used to build the word vectors.

There are two main word2vec models: Continuous Bag of Words (CBOW) and Skip-Gram. In the CBOW model, we predict a word given a context window. Skip-Gram is the opposite: the context is predicted given an input word.

In the model CBOW model, shown in Figure3.5, the input layer $\mathbf{x}$ is a vector of one-hot encoded words $\mathbf{x} = \{x_1, ..., x_C\}, x_i \in \mathbb{R}^V$, where C is the context window size and V is the vocabulary size. The input vectors are connected to the hidden layer $\mathbf{h} \in \mathbb{R}^N$ through a weight matrix $\mathbf{W} \in \mathbb{R}^{V \times N}$. The output layer is output word y in the training example which is also one-hot encoded. The hidden layer is connected to the output layer through a weight matrix $\mathbf{W}' \in \mathbb{R}^{N \times V}$.

The first step is to evaluate the output of the hidden layer $\mathbf{h}$, computed by

$$\mathbf{h} = \frac{1}{C}(\sum_{i=1}^{C} \mathbf{x_i})\mathbf{W} \tag{3.2}$$

This operation takes the rows of $\mathbf{W}$ corresponding to the vocabulary indexes of the words in the context window (this is a property of the one-hot encoding of $\mathbf{x}$) and averages them.

Next the inputs to each node in the output layer are computed

$$u_j = \mathbf{h}\mathbf{v}'_{\mathbf{w_j}} \tag{3.3}$$

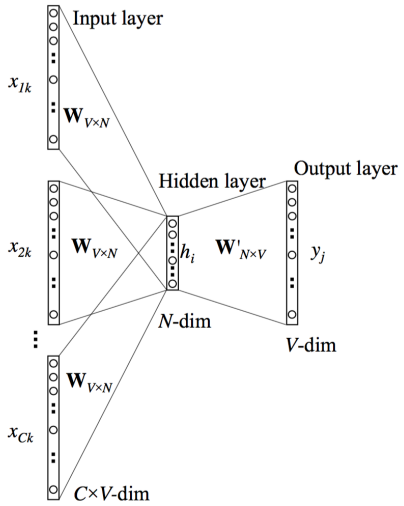


Figure 3.3: Word2vec CBOW model

where $\mathbf{v}'_{\mathbf{w_j}}$ is the j-th column of the matrix $\mathbf{W}'$. The final output y_j is computed by passing u_j through the softmax function.

$$y_j = p(w_{y_j}|w_1, ..., w_C) = \frac{exp(u_j)}{\sum_{j'=1}^{V} exp(u_{j'})} \tag{3.4}$$

The training objective is to maximize equation 3.9, the conditional probability of observing the word w_O (its index in the output layer is denoted with j^*) given the input context w_I. Therefore the loss function to minimize will be

$$\mathcal{L} = -\log(p(w_O|w_I)) \tag{3.5}$$

$$= -u_{j^*} - \log(\sum_{j'=1}^{V} exp(u_{j'})) \tag{3.6}$$

The Skip-Gram model, shown in Figure 3.4 is the opposite of CBOW; in the former we predict the context C given an input word, where in the latter we predict the word from C.

The hidden layer is simply computed as

$$\mathbf{h} = \mathbf{xW} \tag{3.7}$$

while the inputs to each node in the output layer are computed as

$$u_{c,j} = u_c = \mathbf{hv}'_{\mathbf{w_j}} \forall j \in \{1,2, ..., C\} \tag{3.8}$$

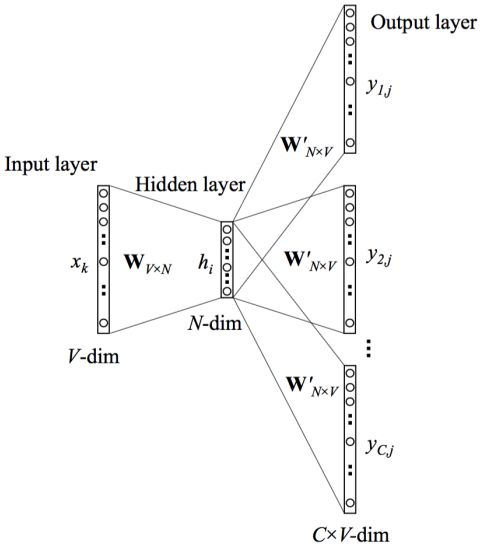


Figure 3.4: Word2vec skipgram model

At the output layer, instead of one multinomial distribution, C multinomial distributions are outputted.

$$y_{c,j} = p(w_{c_j} = w_{O,c}|w_I) = \frac{exp(u_{c,j})}{\sum_{j'=1}^{V} exp(u_{j'})} \tag{3.9}$$

The loss function is changed to

$$\mathcal{L} = -\log(p(w_{O,1}, ..., w_{O,C}|w_I)) \tag{3.10}$$

$$= -\sum_{c=1}^{C} u_{c,j_c^*} + C\log(\sum_{j'=1}^{V} exp(u_{j'})) \tag{3.11}$$

GloVe

Context window-based models such as Word2vec have the disadvantage of ignoring the global corpus statistics. As a result, repetition and large-scale patterns may not be learned during training.

For example, words "the" and "cat" might be used often together, but word2vec doesn't know if this is because "the" is a common word or if this is because the words "the" and "cat" have a strong linkage.

GloVe, introduced by Pennington *et al.* (2014), is a distributed model that takes into account both local context and global corpus information when training the word vectors.

The first step is to build a co-occurrence matrix. Local context is taken into account by computing the matrix using a fixed window size (words are co-occurrent when they appear together within the same window). For example, the sentence "The cat sat on the mat" with a context window size of 4 (2 words to the left and two to the right of the central word) is converted to the following co-occurrence matrix

	the	cat	sat	on	mat
the	0	1	2	1	1
cat	1	0	1	1	0
sat	2	1	0	1	0
on	1	1	1	0	1
mat	1	0	0	1	0

The underlying principle behind GloVe is that the co-occurrence ratios between two words in a context are strongly correlated to their meaning.

The relation between the ratios can be expressed with the following equation:

$$F(w_i, w_j, \tilde{w_k}) \approx \frac{P_{ij}}{P_{jk}} \tag{3.12}$$

P_{ij} denotes the probability of the word j to appear in the context of i, and can be computed as

$$P_{ij} = \frac{X_{ij}}{X_i} \tag{3.13}$$

where$X$ denotes the co-occurrence matrix, X_{ij} denotes the i, jth element in X (equal to the number of times word j appears in the context of word i) and $X_i = \sum_l X_{il}$ is the total number of words that have appeared in the context of i.

F is some function that takes the embeddings for the words i, k, j as input. Since one of the goals of GloVe is to create word vectors that express meaning using simple arithmetic (vector addition and subtraction), a function F must be chosen so that the resulting vectors meet this property.

Since it's desirable for arithmetic between the vectors to have meaning, the input to the function F can be formulated as the result of arithmetic between vectors. The simplest way to do this is to compute the input to F as the difference between the compared vectors:

$$F(w_i - w_j, \tilde{w_k}) \approx \frac{P_{ij}}{P_{jk}} \tag{3.14}$$

Now, a linear relation between $w_i - w_j$ and $\tilde{w_k}$ must be created. This can be achieved by using the dot product:

$$F((w_i - w_j) \cdot \tilde{w_k}) \approx \frac{P_{ij}}{P_{jk}} \tag{3.15}$$

By taking the logarithm of the probability ratios the ratio can be converted into a subtraction between probabilities, and a bias term is for to each word to take into account the fact that some words occur more frequently than others.

The result of these operations is the following equation:

$$(w_i - w_j) \cdot \tilde{w_k} + b_i - b_j = \log(P_{ik}) - \log(Pjk) \tag{3.16}$$

This equation can be converted into an equation over a single entry in the co-occurrence matrix.

$$w_i \cdot \tilde{w_k} + b_i = \log(P_{ik}) = \log(X_{ik}) - \log(X_i) \tag{3.17}$$

By absorbing the term $-\log(X_i)$ on the right-hand side into the bias term b_i, and adding an output bias $\tilde{b_k}$ for symmetry, the final equation is obtained:

$$w_i \cdot \tilde{w_k} + b_i + \tilde{b_k} = \log(X_{ik}) \tag{3.18}$$

The model is trained by minimizing an objective function J, which evaluates the sum of all squared errors based on Equation 3.18, weighted by a function f

$$J = \sum_{i=1}^{V} \sum_{k=1}^{V} f(X_{ik})(w_i \cdot \tilde{w_k} + b_i + \tilde{b_k} - \log(X_{ik}))^2 \tag{3.19}$$

An f must be chosen such that it helps in preventing common word pairs (i.e. those with large X_{ij} values) from skewing excessively the objective function. The authors of the paper found the following function to perform well:

$$f(X_{ik}) = min\left(1, \left(\frac{X_{ik}}{x_{max}}\right)^{\alpha}\right) \tag{3.20}$$

This function cuts the output of extremely common word pairs (where $X_{ij} > x_{max}$) and simply returns 1. For all other word pairs, some weight is returned in the range [0,1], where the distribution of weights in this range is determined by α.

3.2 Attention Mechanisms

Originally Attention Mechanisms were used primarily in the field of computer vision, beginning in the 1990s. However, they weren't widely adopted until Mnih *et al.* (2014) applied Attention Mechanisms to an RNN model for image classification.

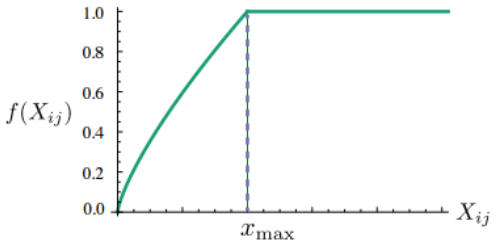


Figure 3.5: Weighting function f with $\alpha = 3/4$

Later on, researchers experimented with Attention Mechanisms for machine translation tasks. Bahdanau *et al.* (2014) used this method to perform translations concurrently. Their work was the first to apply Attention Mechanisms to the field of Natural Language Processing. After this study Attention Mechanisms became common in NLP tasks based on neural networks such as RNN and CNN.

Attention Mechanisms are an approximation of the human sight mechanism. According to the task at hand human visual attention allows us to focus on a specific region and then adjust the focal point accordingly rather then scanning the entire scene end to end. That is, we pay attention to a certain area given the features we detect in that area. Similarly for language, we can discern between patterns that are informative for our task and attend to them, while giving less importance to the other detected patterns.

Attention in deep learning models can be interpreted as a vector of importance weights: in order to infer one element, such as a pixel in an image or a word in a sentence, the attention vector is used to estimate how strongly it is correlated with the input feature vectors and take the sum of their values weighted by the attention vector to obtain a final representation of the input.

Figure 3.6: Attention heatmap for an image detection task

Formally, attention is computed as a function of the input as a set of key-value

pairs K, V and of a query vector Q. First, similarity between each key K_i and the query is computed through a function f to obtain a weight. A typical similarity function used is dot product, but other functions can be used as well. A softmax function is then used to normalize these weights. Finally the attention weights are used to compute a weighted sum of the values and obtain the final representation. In current NLP work, the key and value are often the same, therefore $K = V$.

$$f(Q, K_i) = \begin{cases} Q^\top K_i & \textit{dot product} \\ Q^\top W_a K_i & \textit{generalized dot product} \\ W_a[Q, K_i] & \textit{concatenation} \\ v_a^\top tanh(W_a Q + U_a K_i) & \textit{perceptron} \end{cases}$$

$$a_i = softmax(f(Q, K_i)) = \frac{exp(f(Q, K_i))}{\sum_j exp(f(Q, K_j))}$$

$$Attention(Q, K, V) = \sum_i a_i V_i$$

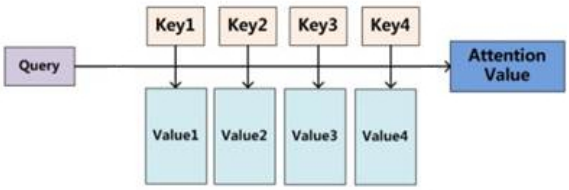


Figure 3.7: Diagram for attention computation

3.3 Hierarchical Attention Networks

This model, introduced by Yang *et al.* (2016), is based on the assumption that textual documents have a hierarchical structure: words form sentences, sentences form a document. The goal is to represent sentence meaning from the words and then represent the meaning of the document from the sentences. However not all parts of a document are equally relevant: different words and sentences in a document have different level of informativeness. Therefore an attention mechanism is used so that the model can focus on important words and sentences.

Determining the relevant sections however involves modeling the interaction between the words, not just their presence in isolation: the importance of a document

section is highly context dependent, i.e. the same word or sentence can have different meanings in different contexts.

Figure3.9 shows the attention allocated at word level (blue) and sentence level (red) for a sentiment analysis task on a review from Yelp. The highlighted sentences are more meaningful compared to the others, while words such as *delicious* and *amazing* are relevant in attributing the positive sentiment expressed by the review.

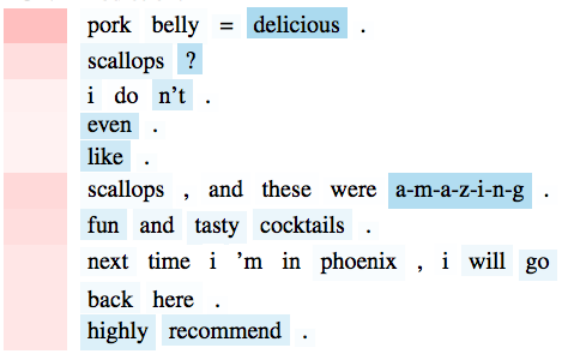


Figure 3.8: A classification example on a review from the Yelp dataset

3.3.1 Model characteristics

The model consists of two main components: recurrent bidirectional layers and attention layers. A first recurrent bidirectional layer learns the meaning of the words forming each sentence and returns a vector for each word in the input sentence. Then an attention layer gets the weights corresponding to each word and scale each word vector according to its corresponding weight. Then it aggregates the words representations into a sentence representation by doing a sum of the weighted vectors.

The same procedure is applied to sentence vectors, using another recurrent bidirectional layer and another attention layer, in order to generate a final vector that represents the whole document.

3.3.2 Basic component: Gated Recurrent Unit (GRU)

A Gated Recurrent Unit (Cho *et al.* , 2014) is a type of gated Recurrent Neural Network, a variation of the standard RNN model. This family of models is designed to produce a representation of a sequential input by maintaining an internal status s which is updated at every step t. The general formula of a recurrent network is shown below:

$$h_t = f(x_t, h_{t-1} : \theta) \tag{3.21}$$

With respect to simple RNNs Gated Recurrent Units are designed to address the problem of long-term dependencies in documents. Two gates control how information is updated at the state at each step t.

- Update gate $z_t \in [0,1]$ decides how much of the new information to add
- Reset gate $r_t \in [0,1]$ decides how much of the past information to retain

At each step t the state h_t is updated as follows

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{3.22a}$$
$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{3.22b}$$
$$\hat{h} = tanh(W x_t + r_t \odot (U h_{t-1})) \tag{3.22c}$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t \tag{3.22d}$$

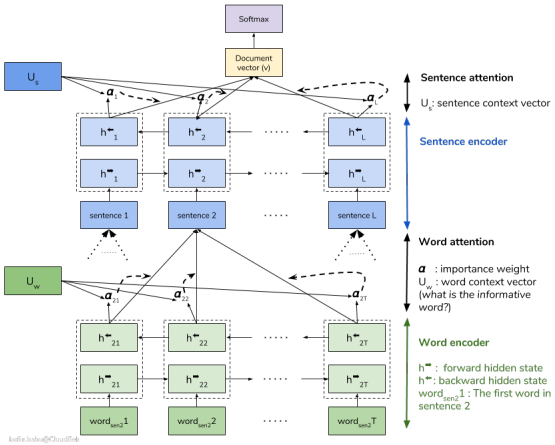


Figure 3.9: Hierarchical attention network overview

3.3.3 Model overview

A document D has L sentences s_i, with each sentence containing T_i words. $w_{it}, i \in [1, L], t \in [1, Ti]$ represents the tth words in the ith sentence.

1. **Word embeddings** Each word w_{it} is mapped to a vector through an embedding matrix W_e

$$x_{it} = W_e w_{it} \tag{3.23}$$

2. **Word encoder** A bidirectional GRU summarizes information from both directions for words, therefore incorporating the contextual information in the representation

$$h_{it}^{fw} = GRU^{fw}(x_{it}), t \in [1, T] \tag{3.24}$$
$$h_{it}^{bw} = GRU^{bw}(x_{it}), t \in [T,1] \tag{3.25}$$
$$h_{it} = [h_{it}^{fw}, h_{it}^{bw}] \tag{3.26}$$

3. **Word attention** Each word representation h_{it} goes through a fully connected layer and a non-linear activation function

$$u_{it} = tanh(W_w h_{it} + b_w) \tag{3.27}$$

The importance of each word u_{it} is measured through its similarity with a vector u_w normalized through a softmax function, obtaining the normalized importance weight αit.

u_w can be interpreted as a high level representation of a highly informative word, it's initialized randomly and jointly learned during the training

$$\alpha_{it} = \frac{exp(u_{it}^{\top} u_w)}{\sum_{\tau}^{T} exp(u_{i\tau}^{\top} u_w)} \tag{3.28}$$

Each sentence embedding s_i is calculated as a weighted sum of the word representations scaled by their attention weights

$$s_i = \sum_{t}^{T} \alpha_{it} h_{it} \tag{3.29}$$

4. **Sentence encoder**

A second bidirectional GRU summarizes information from both directions for sentences

$$h_i^{fw} = GRU^{fw}(s_i), i \in [1, L] \tag{3.30}$$
$$h_i^{bw} = GRU^{bw}(s_i), i \in [L,1] \tag{3.31}$$
$$h_i = [h_i^{fw}, h_i^{bw}] \tag{3.32}$$

5. **Sentence attention** Attention is computed for sentences representations in the same way as words representations

$$u_i = tanh(W_s h_i + b_s) \tag{3.33}$$

$$\alpha_i = \frac{exp(u_i^\top u_s)}{\sum_l^L exp(u_l^\top u_s)} \tag{3.34}$$

$$v = \sum_i^L \alpha_i h_i \tag{3.35}$$

6. **Document classification**

   The document vector v is a high level representation of the document and can be used as features vector for the document classification task by feeding it to a fully connected layer and a final non-linear activation function f.

$$p = f(W_c v + b_c) \tag{3.36}$$

3.4 Convolutional Neural Networks

Convolutional Neural Networks (LeCun *et al.* , 1998) are a class of deep, feed-forward artificial neural networks and use a variation of multilayer perceptrons designed to require minimal preprocessing. These models are inspired by animal visual cortex.

CNNs were conceived to be used computer vision, however they've recently been applied to various NLP tasks with promising results. When a CNN is applied on text data the result of each convolution will fire when a linguistic pattern is detected. By varying the size of the kernels and concatenating their outputs, patterns of multiples sizes can be detected. The patterns are word n-grams (with n equal to the filter size) that can be detected everywhere in the document regardless of their position.

A strong argument in favour of CNNs is that they are fast, as convolutions are a fundamental operation in computer graphics and they implemented on a hardware level on GPUs. As a consequence CNNs are usually faster to train than other models commonly used for NLP such as RNNs.

Instead of image pixels, the input to most NLP tasks are sentences or documents represented as a matrix. Each row of the matrix corresponds to one token. Typically, the row vectors are word embeddings trained with word2vec or GloVe, but they could also be one-hot encoded vectors. For example, for a 10 word document using a 100-dimensional embedding, the corresponding input would be a 10x100 matrix.

Figure 3.11 shows an example of Convolutional Neural Network for sentence classification, taken from Zhang & Wallace (2015). This model uses three filter region sizes: 2, 3 and 4, each of which has 2 filters. Every filter performs convolution

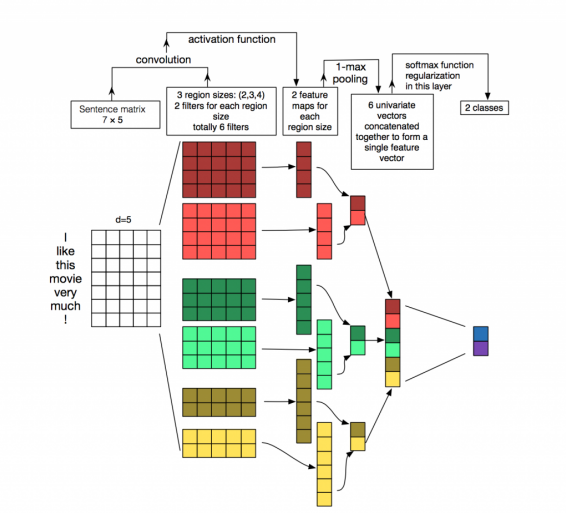


Figure 3.10: Illustration of a CNN architecture for document classification

on the sentence matrix and generates variable-length feature maps. Max pooling is then performed over each map and the resulting 6 features are concatenated to form a feature vector (the sentence representation). The final softmax layer then receives this feature vector as input and uses it to classify the document.

A key element of Convolutional Neural Networks are pooling layers, typically applied after the convolutional layers. Pooling layers subsample their input. The most common way to do pooling is to apply a max operation on the output of each filter. Pooling can be performed over the complete matrix or over a window. In NLP pooling is typically applied over the complete output, in order to yield a fixed size output matrix regardless of the filters size, which is usually a requirement for classification.

Pooling also reduces the output dimensionality but keeps the most salient information. By performing the max operation only the most important features are sampled: as a result global information about locality (where in a sentence a pattern was detected) is lost, but local information captured by the filters (features produced by the convolutions with high values) are preserved.

The most natural fit for CNNs are classifications tasks, such as Sentiment Analysis, Spam Detection or Topic Categorization. Because convolutions and pooling operations lose information about the local order of words sequence tagging as in PoS Tagging or Entity Extraction is a harder to perform with a CNN architecture.

Kim (2014) evaluates a CNN architecture on various classification datasets, mostly comprised of Sentiment Analysis and Topic Categorization tasks. The CNN architecture achieved very good performance across datasets, and set a new state-of-the-art on a few. The input is a sentence matrix where the rows are word2vec word embeddings. That's followed by a convolutional layer with multiple filters, then a

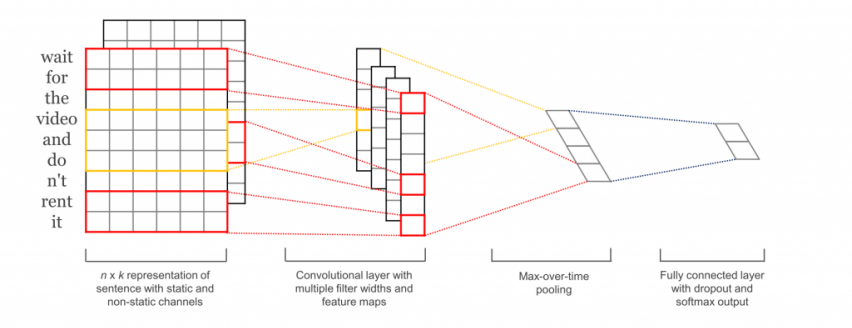


Figure 3.11: Convolutional Neural Networks for sentence classification (Kim, 2014)

max-pooling layer, and finally a softmax classifier.

Alternatively, an attention layer can be used instead of pooling to perform dimensionality reduction after a convolution.

Chapter 4

Related Work

Recently there has been significant work on the topic of automated coding of clinical notes (Perotte *et al.* , 2013; Kavuluru *et al.* , 2015; Shi *et al.* , 2017; Baumel *et al.* , 2017; Karmakar, 2018; Escudié *et al.* , 2018; Mullenbach *et al.* , 2018). We review some of the recent work.

Perotte *et al.* (2013) experimented with flat and hierarchical SVMs using tf-idf features on the MIMIC-II dataset. The hierarchical classifier exploit the ICD-9 hierarchy and makes a prediction only on the child codes with positively predicted parents.

The flat SVM treats each label as an independent binary decision. One disctinct linear SVM classifier is created for each possible ICD9 code, excluding the root, which is always positive. All documents in the training set labeled with that ICD9 code are considered positive, and all others are considered negative. Ancestors of a positive ICD9 code must also be positive, and descendants of a negative ICD9 code must also be negative. This relationship is taken into consideration only during testing setups where single predictions are augmented to include all ancestors.

The hierarchical SVM takes into consideration the structure of the ICD9 code tree also during training. An augmented dataset is created, where each document is labeled with codes from the whole tree, not just the leaf nodes. During training many SVM classifiers are created and trained, one for each code of the hierarchy excluding the root. The classifier associated with a given code in the hierarchy is applied only if its parent code has been classified as positive. Therefore, only documents whit a positive parent code are fed to the child classifier. Whereas in the flat SVM each classifier has the same amount of training data, in the hierarchy-based SVM some classifiers could have a very small amount of training examples. The classifiers are applied from the root downward until a child node is classified as negative. This procedure is repeated for all paths from the root to all possible leaves, resulting in a tree of multi-label positive predictions for a given document.

The results show that the hierarchy-based SVM predicts ICD9 codes with a higher F-measure than the flat SVM (29.3% vs 21.1%), with an improved recall

(23.3% vs 13.0%) but a lower precision (39.4% vs 56.2%). These metrics are computed considering only non-augmented labels as target predictions.

Kavuluru *et al.* (2015) built classifiers for ICD-9 diagnosis codes over three datasets: a dataset of 71,463 EMRs corresponding to in-patient visits from the University of Kentucky (UKY) Medical Center, a smaller subset of this dataset and a third gold standard dataset of radiology reports. They performed feature selection and used a variety of methods such as SVM, naive Bayes, and logistic regression for this problem. They achieved a micro F-score of 0.48 over all codes with at least 50 training examples and a micro F-score of 0.54 on the set of codes that occur at least in 1% of the UKY dataset.

Shi *et al.* (2017) uses a character-aware neural language model to generate hidden representations of the documents and ICD codes, with an attention mechanism to match the documents and the codes descriptions. The model is trained on a subset of MIMIC-III and the ICD coding target is restricted to the 50 most frequent codes. The architecture consists of four modules, which are used for encoding the input documents, encoding the ICD codes based on their textual descriptions, matching encoded documents with encoded ICD descriptions, and assigning the ICD codes. The overall architecture is illustrated in Figure 4.1.

The documents encoder is based on a long short-term memory (LSTM) recurrent network. For each input document, a character-level LSTM and word-level LSTM are used to obtain its hidden representation. Each input document is represented as a sequence words, where every word is a matrix composed of character embeddings. Each word matrix is fed to the fisrt LSTM and the hidden state of the network at the last time step is taken as the hidden representation of the word. In the word-level LSTM the input is the sequence of sentence matrices, each obtained by concatenating the word representations outputted by the previous layer. The last hidden state is the representation of each sentence.

For the ICD codes descriptions embedding, the same two-level LSTM architecture is adopted to obtain the hidden representations. The parameters of the layers for the codes descriptions encoder and the document encoder are independent, in order to better adapt to the different language styles contained in each of the two sets of texts.

The attentional matching mechanism performs the cosine similarity between the hidden representations of each ICD code and the input documents, to extract the most relevant features for each code independently before feeding the obtained vectors (one per code) to the final softmax classifier.

The best model achieves a micro-F1 score of 0.53, however, given the restricted prediction target, this work is not directly comparable to the results of the other studies presented here.

Baumel *et al.* (2017) experimented with both MIMIC-II and MIMIC-III datasets and proposed a model based on hierarchical attention networks (Yang *et al.* , 2016)

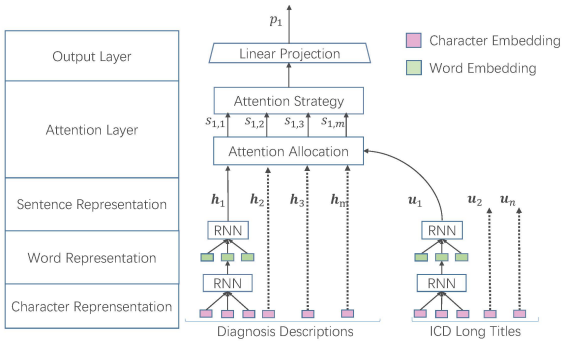


Figure 4.1: Model Architecture by Shi *et al.* (2017)

with per-label attention matching to be able to handle multi-label classification. A hierarchical model with two levels of bidirectional GRU encoders is applied. The first bidirectional GRU operates over tokens and encodes sentences. The second bidirectional GRU encodes the document, applied over all the encoded sentences. In this architecture, each GRU is applied to a much shorter sequence compared with a single GRU. To take advantage of the property that each label is referred to from different parts of the text, an attention mechanism over the second GRU with different weights for each label is used. This allows the model to focus on the relevant sentences for each label (Choi *et al.* , 2016). Attention is also applied over the first GRU using shared weights for all the labels. Each sentence in the input text is encoded to a fixed length vector by applying an embedding layer over all the inputs, applying a bidirectional GRU layer on the embedded words, and using an attention mechanism to encode the bidirectional GRU outputs. After the sentences are encoded into a fixed length vector, a second bidirectional GRU layer over the sentences using different attention layers to generate an encoding specified to each class. Finally a fully connected layer with softmax is applied for each classifier to determine if the corresponding label should be assigned to the document.

To test the impact of training size, both the MIMIC II and MIMIC III datasets were used. The authors created two training sets, one based on MIMIC II and the other based on MIMIC III, and a common test-set comprising summaries of unseen patients. This model achieved a micro-F1 score of 40.52% when trained on MIMIC-III.

Mullenbach *et al.* (2018) presented a model composed of an embedding and a CNN layer and an individual attention layer for each code. They also proposed adding regularization to this model using textual code descriptions. In the embedding layer pre-trained word vectors for each token in the document are concatenated to form a document matrix. This input matrix is then fed to a convolutional layer which performs a 1D convolution with unitary stride and is composed by filters

having all the same width, so that the result of the convolution is a matrix obtained by concatenating the vectors generated by each filter. Additionally the input matrix is padded so that the output matrix has the same length of the input document. After the convolution, the matrix is reduced to a vector by the attention layer. Similarly to Baumel *et al.* (2017) a per-label attention mechanism is applied in order to select the most informative features for each code independently before feeding the obtained vectors (one per code) to the final softmax classifier. Their best model on MIMIC-III obtained a micro F1-score of 52.1%, which was achieved by the convolutional model without regularization.

The survey of previous work indicates the essential role of the attention layer to obtain good performance. Moreover the hierarchical attention model used by Baumel *et al.* (2017), which achieves state of the art performance in the most common document classification benchmarks, is outperformed by the much simpler convolutional model by Mullenbach *et al.* (2018), which achieves the best results on this specific task and therefore represents the baseline for our study. The poor performance of the recurrent models compared to convolutional models is probably caused by the considerable length of the input documents: recurrent models are unable to effectively track very long-term temporal dependencies, whereas convolutional architectures don't suffer from this problem as they only detect pattern across adjacent areas of the input data.

Chapter 5

Methodology

The task of predicting ICD codes from clinical notes can be modeled as a multi-task binary classification problem: for each hospital admission every ICD-9 code can be present (labeled 1) or absent (labeled 0). For each code the models outputs the likelihood (in the range 0-1) for that code to be associated to the input document. Output values greater than 0.5 are assigned as present (1), the other ones are assigned as absent (0).

In the present work we present a neural model that yields state of the art results for the MIMIC-III dataset, outperforming the previous best result of the model that currently achieves the best results on the MIMIC-III dataset (Mullenbach *et al.* , 2018). We achieve these results through different enhancements.

We initialize the embedding layer with word vectors pre-trained on an external common English corpus instead of the MIMIC corpus. We investigate the problem of preprocessing of tokenization of the input documents and we propose an approach that handles more accurately common medical expressions and the protected health information tokens.

We then experiment with a model similar to the one used by Baumel *et al.* (2017), with hierarchical representations on words and sentences and a label-dependend attention layer (also used by Mullenbach *et al.* (2018)). However, their model uses GRU based encoders while ours uses a GRU for the word-level encoder and a CNN for the sentence-level encoder.

Finally, Mullenbach *et al.* (2018) used code descriptions to regularize their model, without improvements in the prediction of the full codes. Similarly to Shi *et al.* (2017) we embed the codes description through a recurrent encoder and we use the obtained representations as context vector to compute the label-dependent attention.

5.1 Preprocessing

As mentioned previously clinical notes contained in EHRs exhibit some textual constructs that make preprocessing necessary in order to perform an effective tokenization. The reason is that the language used in the notes is medical English, which differs from common English for the presence of expressions such as abbreviations, vital signs transcriptions and drugs dispensing reports. Also these notes contains a high amount of misspellings, likely a result of being written by doctors while being under pressure. Moreover the clinical notes contained in the MIMIC dataset contain de-identified protected health information (PHI): elements of the text such as names, dates, locations are replaced by anonymized tokens in the form of generic information enclosed between special symbols ([* ... **]). For example, a patient's name becomes [**Known lastname 101054**]. The number at the end is a unique identifier associated to the information that's being anonymized, so every occurrence of that information in the dataset (e.g. a person's name) will have the same numerical identifier.

In the previous studies the preprocessing approach taken was to tokenize the input text and then convert all non-alphabetical characters to pseudo-tokens (e.g. "11/2/1986" to "dd/d/dddd") (Baumel *et al.* , 2017) or to remove tokens that don't contain any alphabetic characters (e.g. removing "500" but keeping "250mg") and to split the tokens on symbols and spaces (Mullenbach *et al.* , 2018).

In our approach we use the spaCy [1] library to tokenize the text, using custom rules to handle correctly the most frequent medical abbreviations, as those are often splitted incorrectly by the tokenizer. Then we remove tokens that don't contain any alphabetic characters: this rule removes the numbers and spurious tokens entirely made of symbols, but it retains words that contain numbers. We also take spacial care of the anonymized expressions, that would otherwise be split in several tokens: we map them to special tokens, e.g. [**Last Name 1234**] becomes "lastnametoken". This mapping preserves the structure of the text, is easily interpretable when reading the processed notes and allows correct tokenization of these expressions. We also performed experiments with the numerical identifiers associated to the anonymized tokens: we tried to preserve the identifiers, e.g. mapping [**Last Name 1234**] to "lastnametoken_1234". We observed that this approach caused a noticeable growth of the number of unique tokens in the corpus and this led to worse results.

The vocabulary is built by taking all the tokens that appear at least 3 times in the training set; any out-of-vocabulary word is replaced by a special token that represents unknown words. We also tried to map every unknown word to the word in the vocabulary with the shortest Levenshtein distance (similar to the approach

[1] https://spacy.io

taken by Baumel *et al.* (2017)) but this approach didn't yield any performance improvement.

Here is show a sample of clinical note before and after our preprocessing and tokenization. This example illustrates several of the tokenization problems discussed above: anonymized date and name tokens, numbered lists, drug dispensing information, vital signs, etc. The processed note, formatted with one sentence per line, is overall clearer and we can observe how the majority of the problematic expressions are handled correctly. The main remaining issue is the sentence boundary detection, with many sentences incorrectly split int two or more smaller sentences.

```
History of Present Illness:
44 yo female with a h/o left frontal AVM in the supplementary
motor area. The AVM was treated with stereotactic radiosurgery
(Gamma Knife)in [**2114**]. In [**2116**], the patient developed a seizure
disorder. [**2118-5-27**] she developed
headaches and after an MRI and a digital angiogram showed no
residual pathological vessels, a contrast enhancing lesion
with massive focal residual edema was diagnosed- very
likely represents radionecrosis. The patient had midline
shift and mass effect. On [**2118-8-10**] she had a left craniotomy for
resection of the radionecrosis. She then presented to the office
in [**2118-8-27**] with increased left facial swelling and incision
drainage, she was taken to the OR for a wound washout and
craniectomy. She now returns for a cranioplasty after a long
course of outpatient IV antibiotic therapy.

sic

[**2118-12-7**] 03:13PM WBC-13.8*# RBC-4.76 HGB-12.8 HCT-37.6 MCV-79*
MCH-27.0 MCHC-34.2 RDW-14.4
[**2118-12-7**] 03:13PM PLT COUNT-555*
[**2118-12-7**] 03:13PM CALCIUM-9.2 PHOSPHATE-3.4 MAGNESIUM-2.3
[**2118-12-7**] 03:13PM estGFR-Using this
[**2118-12-7**] 03:13PM GLUCOSE-128* CREAT-0.9 SODIUM-141
POTASSIUM-4.1 CHLORIDE-102 TOTAL CO2-30 ANION GAP-13

Discharge Medications:
1. Acetaminophen 325 mg Tablet Sig: 1-2 Tablets PO Q4H (every 4
hours) as needed for pain/t>100/HA.
2. Bisacodyl 5 mg Tablet, Delayed Release (E.C.) Sig: Two (2)
Tablet, Delayed Release (E.C.) PO DAILY (Daily) as needed for
constipation.
3. Docusate Sodium 100 mg Capsule Sig: One (1) Capsule PO BID (2
times a day).
Disp:*60 Capsule(s)* Refills:*2*
4. Hydromorphone 2 mg Tablet Sig: 1-2 Tablets PO Q4H (every 4
hours) as needed for pain.
Disp:*30 Tablet(s)* Refills:*0*
5. Venlafaxine 25 mg Tablet Sig: Two (2) Tablet PO BID (2 times
a day).
6. Fluticasone 50 mcg/Actuation Spray, Suspension Sig: Two (2)
Spray Nasal DAILY (Daily) as needed for nasal congestion.
7. Dexamethasone 2 mg Tablet Sig: One (1) Tablet PO Q6H (every 6
hours) for 6 days: Take 2mg Q6hrs [**Date range (1) 1855**], take 2mg Q12
[**Date range (1) 1856**], Take 2mg Q24 [**12-14**], then stop.
Disp:*16 Tablet(s)* Refills:*0*
8. Levetiracetam 500 mg Tablet Sig: 2.5 Tablets PO BID (2 times
a day).

sic

?????? Have a friend/family member check your incision daily for
signs of infection.
?????? Take your pain medicine as prescribed.
?????? Exercise should be limited to walking; no lifting, straining,
or excessive bending.
?????? You may wash your hair only after sutures and/or staples have
been removed.
?????? You may shower before this time using a shower cap to cover
your head.
?????? Increase your intake of fluids and fiber, as narcotic pain
medicine can cause constipation. We generally recommend taking
an over the counter stool softener, such as Docusate (Colace)
while taking narcotic pain medication.
?????? Unless directed by your doctor, do not take any
anti-inflammatory medicines such as Motrin, Aspirin, Advil, and
Ibuprofen etc.
?????? Clearance to drive and return to work will be addressed at
your post-operative office visit.
```

Listing 5.1: A sample from a discharge summary from MIMIC-III

```
history of present illness
yo female with a h/o left frontal avm in the supplementary motor area
the avm was treated with stereotactic radiosurgery
gamma knife in
in the patient developed a seizure disorder
may
she developed headaches and after an mri and a digital angiogram showed no residual pathological vessels a contrast enhancing
       lesion with massive focal residual edema was diagnosed very likely represents radionecrosis
the patient had midline shift and mass effect
on august
she had a left craniotomy for resection of the radionecrosis
she then presented to the office in
august with increased left facial swelling and incision drainage she was taken to the or for a wound washout and craniectomy
she now returns for a cranioplasty after a long course of outpatient iv antibiotic therapy

sic

december
13pm wbc
rbc
hgb hct mcv
mch mchc rdw
december
13pm plt count
december
13pm calcium phosphate magnesium
december
13pm
estgfr using this
december
13pm
glucose creat sodium
potassium chloride total co2 anion gap discharge medications
acetaminophen mg tablet sig
tablets po q4h every hours as needed for pain t ha
bisacodyl mg tablet delayed release e.c. sig
two tablet delayed release e.c.
po daily daily as needed for constipation
docusate sodium mg capsule sig
one capsule po bid times a day

disp

capsule s
refills hydromorphone mg tablet sig tablets po q4h every hours as needed for pain
disp
tablet s refills
venlafaxine mg tablet sig
two tablet po bid times a day
fluticasone mcg actuation spray suspension sig
two spray nasal daily daily as needed for nasal congestion
dexamethasone mg tablet sig one tablet po
q6h every hours for days
take mg q6hrs
daterangetoken
take mg q12
daterangetoken
take mg q24
december
then stop
disp
tablet s refills
levetiracetam mg tablet sig tablets po bid times a day

sic

have a friend family member check your incision daily for signs of infection
take your pain medicine as prescribed
exercise should be limited to walking no lifting straining or excessive bending
you may wash your hair only after sutures and/or staples have been removed
you may shower before this time using a shower cap to cover your head
increase your intake of fluids and fiber as narcotic pain
medicine can cause constipation
we generally recommend taking an over the counter stool softener such as docusate colace while taking narcotic pain medication
unless directed by your doctor do not take any anti inflammatory medicines such as motrin aspirin advil and ibuprofen etc
clearance to drive and return to work will be addressed at your post operative office visit
```

Listing 5.2: The sample after preprocessing and tokenization

5.2 Word embeddings

The first layer of our network maps words to their continuous embedding space. Each word $w \in R_v$ is mapped to $x \in R_e$ using the embedding weight $W_e \in R^{v \times e}$, where v is the vocab size and e is the embedding dimensionality.

Both Baumel *et al.* (2017) and Mullenbach *et al.* (2018) used word vectors pre-trained on the MIMIC dataset using the *continuous bag-of-words* (CBOW) word2vec model (Mikolov *et al.* , 2013). We swap them with word embeddings pre-trained on an external English corpus using the GloVe model (Pennington *et al.* , 2014). The intuition is that we initialize the embedding layer using vectors trained on common English and then we fine-tune them on medical English during training. Vocabulary words not covered by the pre-trained vectors are initialized randomly. For $e = 100$ we use word embeddings trained on Wikipedia, for $e = 300$ we use word embeddings trained on CommonCrawl, which is a larger corpus and thus has a larger coverage (vectors of dimension 100 pre-trained CommonCrawl are not publicly available).

Mullenbach *et al.* (2018) normalizes the word vectors upon initialization of the embedding layer. We don't perform any normalization as in our experiments we found that it causes a decrease in performance with our chosen vectors.

5.3 Hierarchical model

We propose a hierarchical model conceptually similar to the one used by Baumel *et al.* (2017). The layers architecture remain similar, with the hierarchical segmentation in sentences and words and the double attention layer. However, because the documents contained in MIMIC are composed of a considerable number of sentences (300-400), bidirectional GRUs are unable to handle effectively the resulting long-term temporal dependencies. Because of this, we swap the encoder of the sentence-level layer with a convolutional encoder, since CNNs have been proven to be very effective for this task by Mullenbach *et al.* (2018). We retain the recurrent encoder in the word-level layer since sentences are very short (a few dozens of tokens at most), thus recurrent networks are most effective at detecting temporal patterns inside them.

A document D has L sentences s_i, with each sentence containing T_i words. $w_{ti}, t \in [1, T_i], i \in [1, L]$ represents the tth word in the ith sentence.

1. **Embedding layer** Each word w_{ti} is mapped to a vector through an embedding matrix W_e
$$x_{it} = W_e w_{it} \tag{5.1}$$

2. **Word encoder** The document is segmented in sentences. Each sentence is the input sequence for a bidirectional Gated Recurrent Unit, which summarizes information from both directions at word level, incorporating the contextual

information in the representation. As output we take the sequence of hidden states of the GRU at each time step, where the tth hidden state represents the tth input word. We obtain hi, an embedded representation of the ith sentence of the input document.

$$h_{it}^{fw} = GRU^{fw}(x_{it}) \tag{5.2}$$
$$h_{it}^{bw} = GRU^{bw}(x_{it}) \tag{5.3}$$
$$h_{it}^{*} = [h_{it}^{fw}, h_{it}^{bw}] \tag{5.4}$$
$$h_{it} = tanh(W_w h_{it}^{*}) \tag{5.5}$$

3. **Word attention** The importance of each word h_i is measured through its dot-product similarity with a vector u_w, normalized through a softmax function, obtaining the importance weight αit.

   u_w can be interpreted as a high level representation of a highly informative word for the coding task. It's initialized randomly and jointly learned during the training.

$$\alpha_i = \frac{exp(H_i^{\top} u_w)}{\sum_t^T exp(h_{ti}^{\top} u_w)} \tag{5.6}$$

   Each sentence embedding s_i is computed as a weighted sum of the word representations scaled by their attention weights.

$$s_i = \sum_t^T \alpha_{it} h_{it} \tag{5.7}$$

4. **Sentence encoder** A convolutional layer combines the adjacent sentences in the matrix S, formed by concatenating the sentence vectors s_i: the filters are all of the same size and the stride is set to 1, so that with appropriate input padding the output is a matrix with a length equal to the length of the input. For every step n we compute

$$H' = tanh(W_s * S + b_s) \tag{5.8}$$

5. **Code-dependent sentence attention** The attention at sentence level is computed in a similar way as the word level attention. In this case however we compute an attention vector for each label ℓ by the product between H' and u_l, where u_l is the context vector for label l. The resulting vector is passed

through the softmax function, obtaining a the normalized attention vector for label l.

$$\alpha_\ell = \frac{exp(H'^\top u_\ell)}{\sum_i^L exp(h_i'^\top u_\ell)} \tag{5.9}$$

$$v_\ell = \sum_i^L \alpha_{\ell i} h_i' \tag{5.10}$$

6. **Document classification**

   The document vector v_ℓ is a high level representation of the document with respect to label ℓ and can be used as features vector for the classification task by feeding it to a fully connected layer and a final sigmoid function σ to compute the likelihood for each label ℓ.

$$y_\ell = \sigma(W_\ell^\top v_\ell + b_\ell) \tag{5.11}$$

7. **Training**

   The binary cross-entropy loss is computed on the output of the last layer. The model is then trained to minimize this loss function.

$$L_{BCE}(y, \hat{y}) = -\sum_{\ell=1}^{\mathcal{L}} y_\ell \log(\hat{y}_\ell) + (1 - y_\ell) \log(1 - \hat{y}_\ell) \tag{5.12}$$

5.4 Labels embeddings

Similarly to Shi *et al.* (2017) we embed the codes description through an encoder and we use the obtained representations as context vector to compute the label-dependent attention. However instead of multilayer recurrent network we use a single bidirectional Gated Recurrent Unit, using its last hidden state as embedded representation of each code.

The embedded code description of each code obtained through the recurrent encoder is a representation of that code. That representation is then used as context vector for the attentional matching for that code.

The embedding matrix W_e is shared with the document encoder, while the parameters of the encoder are independent, in order to fit the different language styles of the two input sets.

Formally for every code ℓ, given its description d_ℓ, the corresponding context vector u_ℓ is computed from its description d_l as follows

$$X_\ell = d_\ell W_e \tag{5.13}$$
$$h_\ell^{fw} = GRU^{fw}(X_\ell) \tag{5.14}$$
$$h_\ell^{bw} = GRU^{bw}(X_\ell) \tag{5.15}$$
$$h_\ell = [h_\ell^{fw}, h_\ell^{bw}] \tag{5.16}$$
$$u_\ell = tanh(W_a h_\ell) \tag{5.17}$$

where X_ℓ is the sentence matrix of word vectors for description d_ℓ and h_ℓ^* denotes the hidden state of a GRU cell at the last time step.

We apply label embeddings to both the CAML model by Mullenbach *et al.* (2018) and our hierarchical model. For the attentional matching we compute the similarity with a code representation, which is actually a sentence embedding (the code description): however in CAML we are computing we are comparing the code embedding with a word of the document, while in the hierarchical model we are computing the similarity with a sentence in the document.

Chapter 6

Experimental Setup

6.1 MIMIC-III dataset

We rely on the publicly available MIMIC-III dataset (Johnson *et al.* , 2016), which contains de-identified EHRs of 58,976 patient visits at the Beth Israel Deaconess Medical Center from 2001 to 2012. MIMIC-III includes raw notes for each hospital stay in different categories—discharge summary report, discharge summary addendum, radiology note, nursing notes, etc. Following Baumel *et al.* (2017) and Mullenbach *et al.* (2018) we retain only discharge summaries and their addenda, which contain the most informative diagnostical information. Sometimes there is more than one discharge summary for each patient admission as addenda are incorrectly tagged as discharge summaries reports. The timestamp of the documents gives us the correct reading order for the patient admission. Also, some of the hospital stays do not have discharge summaries: following previous studies, we only consider those that do (Perotte *et al.* , 2013; Baumel *et al.* , 2017; Mullenbach *et al.* , 2018).

The dataset includes 8,929 unique ICD codes (2,011 procedures, 6,918 diagnoses) for the patients who have discharge summaries. However 5 codes are invalid, hence we ignore them and we evaluate on 8,924 codes. Codes are structured hierarchically, where top level categories and more generic and lower level codes indicate specific diseases. It is worth noting that the codes representation through a hierarchy is only for convenience purposes: the only valid codes for diagnoses and billing are the leaf nodes, which represent the full codes. MIMIC-III doesn't contain any information on the codes hierarchy, only full codes are assigned to the nodes.

The codes in the dataset are distributed such that a minority of the codes makes up the majority of the distribution of label occurrencies. Because of the high number of labels their unbalanced distribution the predictive power of the models will be severely diminished for the rare codes. The distribution is showed in Figure6.1.

We follow the train, test, and development splits publicly shared by Mullenbach

et al. (2018). These splits are patient independent. The statistical properties of all the sets are shown in table 6.2. As shown in Figure 6.3a, 6.3b and 6.3c, the frequency distribution of the codes is similar between the splits.

The discharge summaries can be very long, and this can cause issues as any architecture which relies on temporal dependencies will suffer without long-term memory. The mean number of words per note is 1505, with a standard deviation of 775. The quantile function of the number of words per note is represented in **??**. Discharge summaries exhibit a fixed structure: the note is usually divided into sections such as "history of present illness", "past medical history", "hospital course", "social history", "family history", "discharge diagnosis", "discharge medications". Some sections of the documents are more relevant than others for the classification task, the most important of all being the diagnosis part. However every section can hold relevant information: for example the patient history section is important as some codes are related to past conditions or procedures (e.g. "personal history of malignant neoplasm of prostate" or "personal history of venous thrombosis and embolism"). Finally the notes can be hard for the models to classify as they are written in medical English, which has a number of differences with respect to common English used in other corpora, and misspellings are frequent.

level	# nodes	total count	min count	max count	coverage	mean	std dev
0	1	891094	891094	891094	1.00	891094.00	0.00
1	4	891094	22544	553750	1.00	222773.50	207280.98
2	73	891094	0	140257	1.00	12206.77	23305.51
3	526	891094	0	53211	1.00	1694.10	5105.10
4	3585	887782	0	23412	1.00	247.64	1272.01
5	10038	768131	0	20703	0.99	76.52	538.27
6	7313	249010	0	12891	0.98	34.05	301.07
7	867	10304	0	812	0.96	11.88	48.88

Table 6.1: Statistics about codes distribution in MIMIC-III in relation to the the ICD-9 hierarchy

6.2 Implementation details

We implemented the code for our experiments using PyTorch. As optimizer we use Adam (Kingma & Ba, 2014) with an initial learning rate of 0.0001. We train the models with early stopping, using precision@8 on the development set as stopping criterion with a patience of 10 epochs. We apply a dropout before each layer to reduce overfitting. We also perform layer normalization (Ba *et al.* , 2016) at the

set	# admission ids	# patients	# unique codes	# total codes	# admissions per patient	# codes per admission
full	52726	41127	8929	848688	1.28	16.10
train	47723	36998	8692	758212	1.29	15.89
dev	1631	1374	3012	28897	1.19	17.72
test	3372	2755	4085	61579	1.22	18.26

Table 6.2: The statistics of the different splits of the dataset

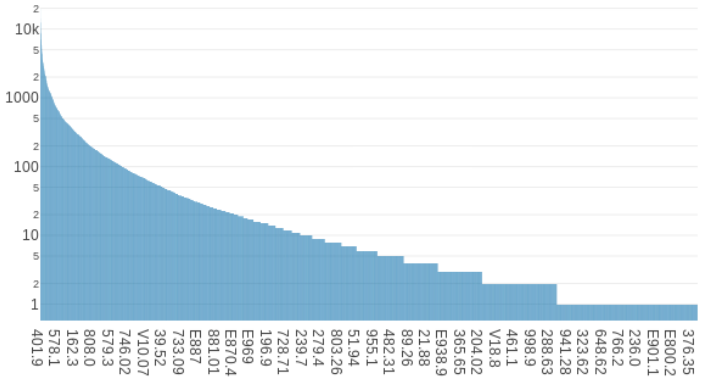


Figure 6.1: Frequency of distribution of ICD-9 codes in MIMIC-III dataset

output of each layer: normalization considerably improved convergence speed and was crucial for the performance of the hierarchical model.

To optimize the hyperparameters of our models we perform a grid search on the word embeddings size d_e, the features size for the GRU hidden state d_w, the number of channels c and the kernel size k of the convolutional layers and on the dropout probability p_w, p_s.

All models were trained with a batch size of 8.

code	description
401.9	unspecified essential hypertension
38.93	venous catheterization not elsewhere classified
428.0	congestive heart failure unspecified
427.31	atrial fibrillation
414.01	coronary atherosclerosis of native coronary artery
96.04	insertion of endotracheal tube
96.6	enteral infusion of concentrated nutritional substances
584.9	acute kidney failure unspecified
250.00	diabetes mellitus without mention of complication type ii or unspecified type not stated as uncontrolled
96.71	continuous invasive mechanical ventilation for less than 96 consecutive hours
272.4	other and unspecified hyperlipidemia
518.81	acute respiratory failure
99.04	transfusion of packed cells
39.61	extracorporeal circulation auxiliary to open heart surgery
599.0	urinary tract infection site not specified
530.81	esophageal reflux
96.72	continuous invasive mechanical ventilation for 96 consecutive hours or more
272.0	pure hypercholesterolemia
285.9	anemia unspecified
88.56	coronary arteriography using two catheters
244.9	unspecified acquired hypothyroidism
486	pneumonia organism unspecified
38.91	arterial catheterization
285.1	acute posthemorrhagic anemia
36.15	single internal mammary coronary artery bypass
276.2	acidosis
496	chronic airway obstruction not elsewhere classified
99.15	parenteral infusion of concentrated nutritional substances
995.92	severe sepsis
V58.61	long term current use of anticoagulants

Table 6.3: Top 30 codes in MIMIC-III

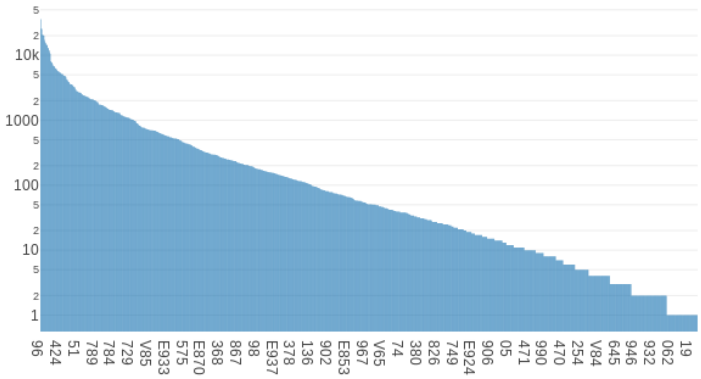
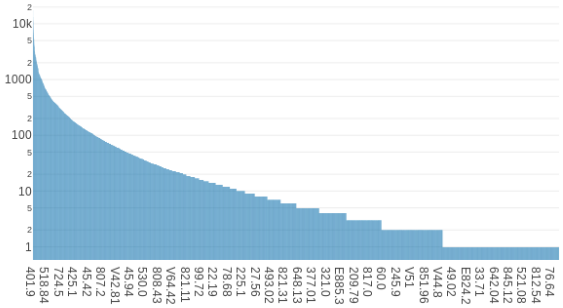


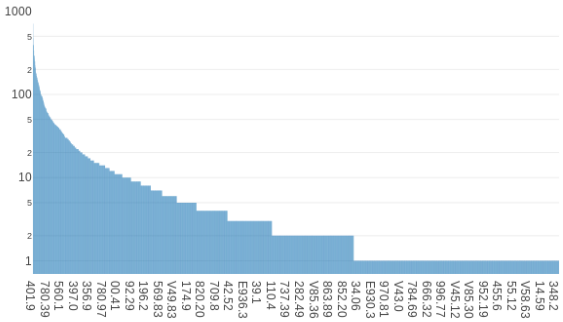
Figure 6.2: Frequency of distribution of ICD-9 header codes in MIMIC-III dataset

	candidates	Mullenbach *et al.* (2018)	Our
d_e	100, 300	300	300
c	50, 100, 125, 200	125	100
d_w	100, 200	-	200
k	3, 5, 9, 10, 15	9	3
p_w	0.2, 0.5	0.2	0.5
p_s	0.5, 0.5	-	0.5

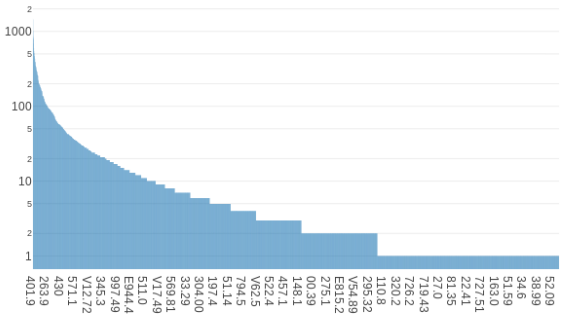
Table 6.4: Hyperparameter candidates for each model and optimal values selected by grid search



(a) Training



(b) Development



(c) Validation

Figure 6.3: Frequency of distribution of ICD-9 codes for the three splits of the dataset

Chapter 7

Evaluation metrics

To directly compare the results of our experiments with the prior work for our evaluation we focus on the micro-averaged and macro-averaged F1 and area under the ROC curve (AUC).

F1-score is a harmonic mean of precision and recall. It is widely used to evaluate the performance of a binary classifier on imbalanced data. The AUC score is computed as the area under the ROC curve, obtained by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. Intuitively, the ROC AUC score measures the probability that the model assigns higher score for a positive instance than negative one. The lower bound is 0.5, which is the score obtained by a classifier that classifies the samples as positives with probability 0.5 (a random classifier). Since ROC AUC measures the true negatives, which are extremely frequent for this problem, it tends to yield very high scores (above 0.9), so we must take into account this factor during the evaluation of the models.

Micro-averaged values are calculated by treating each text, code pair as a separate prediction, while macro-averaged values are calculated by averaging metrics computed per-label. As a result the macro-averaged metrics put the same emphasis on every label prediction, while micro-averaged metrics balance the difference in number of occurrences between the labels.

Considering that ICD code assignment is generally sparse, with most codes labeled as false and only a few as true, we give more weight to the micro-averaged scores for our evaluation, as they are more informative quantitative metrics for the task at hand.

Chapter 8

Results

Table 8.1 provides the evaluation of our models and the chosen baseline CAML, by Mullenbach *et al.* (2018). We show the differences in performance over the chosen evaluation metrics in terms of chosen model, preprocessing and tokenization and word vectors initialization.

First we observe a general improvement in performance when initializing the embedding matrix with word vectors pre-trained on a general external corpus using Glove with respect to word vectors pre-trained on MIMIC using word2Vec. It's worth noting that the improvement is remarkable even with a restricted vocabulary coverage of (50% for the Wikipedia corpus), meaning that the network is able to effectively train the remaining randomly initialized vectors.

With respect to our preprocessing and tokenization, we measure its impact on CAML using word embeddings pretrained on Wikipedia. We mainly observe that there is a increase in micro-precision, at the cost of a decrease in micro-recall that overall yields a comparable micro-F1 score.

We then swap CAML with our hierarchical model, while retaining our embedding layer and preprocessing. We observe that the overall performance of this model is similar to the CAML performance. However, as opposed to CAML, we didn't fine-tune the hyperparameters for this model, so there is still a margin for performance improvement.

Moreover, as we will show in the qualitative evaluation of the results, the sentence level attention mechanism yields more meaningful and interpretable results compared to the word-level attention mechanism of the original model.

A considerable improvement in performance is brought by embedding the code for the attentional matching. This mechanism is beneficial for both CAML and our model: in both cases we observe an improvement of more than 1% in micro-F1 (0.5409 vs 0.5522 for GRU+CNN, 0.5461 vs 0.5581 for CAML) and an improvement of more than 5% in macro-AUC (0.8793 vs 0.9392, 0.8949 vs for CAML 0.9474).

We also evaluate our models using relaxed metrics: we discount errors at the deeper level of the hierarchy tree by considering as prediction target only the ICD

codes headers. For example, for ground truth code 401.9 ("unspecified essential hypertension"), if instead the code 401.1 ("benign essential hypertension") is predicted, this is not considered an error, but failing to predict any code that's a child of code 401 ("essential hypertension") is considered an error.

With this setup the target space is considerably smaller, with only 1168 labels. As expected the relaxed metrics are significantly higher. The macro-averaged metrics also become more relevant, given the lower labels cardinality and the more balanced distribution (shown in Figure 6.2).

Model	Preproc	Word vectors (coverage)	Precision		Recall		F1		AUC	
			macro	micro	macro	micro	macro	micro	macro	micro
CAML	CAML	MIMIC word2vec (1.0)	0.0837	0.6322	0.0677	0.4428	0.0748	0.5208	0.8865	0.9842
CAML	CAML	Wikipedia GloVe (0.5)	0.0880	0.6027	0.0832	0.4918	0.0855	0.5416	0.8950	0.9855
CAML	Our	Wikipedia GloVe (0.5)	0.0980	0.6268	0.0837	0.4838	0.0903	0.5461	0.8949	0.9846
biGRU+CNN	Our	Common Crawl GloVe (0.7)	0.0854	0.6239	0.0808	0.4775	0.0830	0.5409	0.8793	0.9827
CAML + labels embedding	Our	Common Crawl GloVe (0.7)	0.0937	0.6163	0.0930	0.5100	0.0934	0.5581	0.9474	0.9910
biGRU+CNN + labels embedding	Our	Common Crawl GloVe (0.7)	0.0891	0.6494	0.0852	0.4804	0.0871	0.5522	0.9392	0.9898

Table 8.1: Results on MIMIC-III, full codes (8924 labels)

Model	Preproc	Word vectors (coverage)	Precision		Recall		F1		AUC	
			macro	micro	macro	micro	macro	micro	macro	micro
CAML	CAML	MIMIC word2vec (1.0)	0.2635	0.7457	0.1597	0.5085	0.1988	0.6047	0.6073	0.7530
CAML	CAML	Wikipedia GloVe (0.5)	0.2696	0.7256	0.1950	0.5642	0.2263	0.6348	0.6311	0.7807
CAML	Our	Wikipedia GloVe (0.5)	0.2816	0.7344	0.2027	0.5615	0.2357	0.6364	0.6365	0.7793
biGRU+CNN	Our	Common Crawl GloVe (0.7)	0.2615	0.7325	0.2002	0.5554	0.2268	0.6317	0.6348	0.7763
CAML + labels embedding	Our	Common Crawl GloVe (0.7)	0.2700	0.7182	0.2316	0.5921	0.2535	0.6491	0.6560	0.7945

Table 8.2: Results on MIMIC-III, header codes (1168 labels)

Chapter 9

Qualitative evaluation

In the medical domain interpretability of the results is of fundamental importance. For the automated ICD coding task the system should be able to explain why it predicted each code.

A way to provide insight into the decisional process of the model is to examine which parts of the text were considered as most relevant to each code. Since the labeled-dependent attention mechanism used by current models assigns an importance value to the tokens in the input document, examining attention cores assigned to different parts of the documents can show the linguistic patterns used by the model to make the predictions for each code.

We perform a human evaluation of the quality of the explanations provided by the attention mechanism by examining a set of randomly sampled documents labeled with the 10 most frequent codes in the dataset.

We examine the results produced by CAML and by our hierarchical model. In both cases we embed the codes descriptions for the attentional matching. By choosing these two models we can examine the differences in quality and interpretability for the attention allocated at word level (CAML) and at sentence level (GRU+CNN model).

Figure 9.1, 9.2, 9.3 show the CAML behaviour. In 9.1 the model highlights significant tokens, while in 9.3 we see that even if the document was classified correctly the tokens selected weren't meaningful. Figure 9.2 shows an intermediate scenario, with a mix of relevant and non relevant tokens.

Figure 9.4, 9.5, 9.6 show the behaviour of the hierarchical model. Similarly we show an example of good, mixed and poor attention allocation. However it is observable that when attention is allocated at sentence level, the highlighted portions of the text are more meaningful and more interpretable. Moreover, we observed empirically that sentence with non relevant tokens are far less likely to be highlighted by the hierarchical model than by CAML.

Finally, we generally observe that while the highlighted tokens are generally health-related, they are often not directly related to the code examined. This is

caused by the fact that the models learn indirectly comorbidity information, so to classify a certain pathology they learn the correlation to diseases that are likely to co-occur.

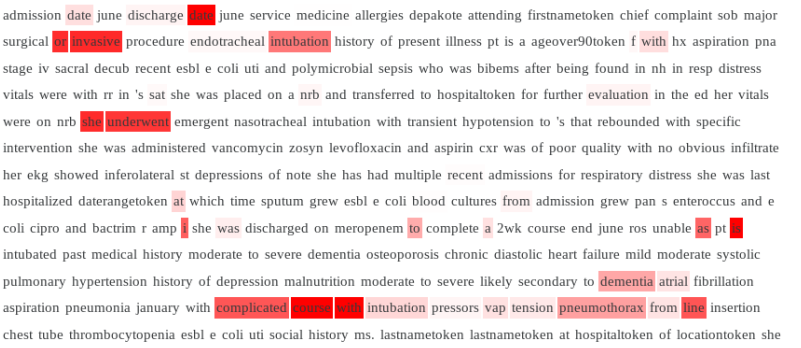


Figure 9.1

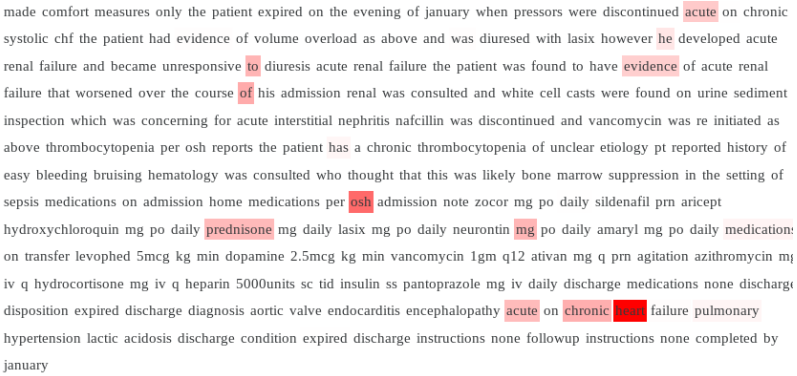


Figure 9.2

was also asked to check her fingersticks times a day and discuss reads with her pcp nametoken will see her pcp days after discharge acute renal failure this pt 's arf resolved with iv fluid rescuscitation anion gap osmolar gap suspect that this was in large part due to renal failure and dehydration ketones were negative hyponatremia corrected sodium level on presentation was actually this resolved with normalization of hyperglycemia sarcoidosis patient was started on prednisone mg daily about weeks ago for treatment during this admission the pt was continued on mg po prednisone with plans to follow up with dr lastnametoken her pulmonologist to discuss other possible treatment options fluticasone spray stopped while on prednisone esp since pt has thrush on bactrim mwf prilosec ca vit d hospitaltoken for proph hypertension bp meds initially held in setting of arf

Figure 9.3

his creatinine was on admission and rose to following administration of clofarabine per sct protocol

he developed metabolic acidosis and worsening confusion over the course of several days leading up to his transplant

on sct day he was febrile and started on vancomycin and cefepime

on sct day he was transferred to icu for worsening mental status increased respiratory distress and renal failure

he remained febrile and was seen by the id consult team and transitioned to meropenem levofloxacin voriconazole

he was intubated electively for multiple procedures including mri vp shunt tap by neurosurgery and bronchoscopy and extubated the following day

he was started on ambisome in place of voriconazole for positive galactomannan on bal presumed aspergillosis pneumonia

he was seen by the neprhology team and started on hd for presumed clofarabine toxicity

mental status improved and he was called out to the bmt service

however two days later he was at dialysis when he developed an episode of altered mental status associated with tachycardia hypertension and hypoxia to on 2l nc

his respiratory status continued to decline and he was reintubated and transferred to the icu

due to electrolyte derangements cvvh was started as well

his abx were broadened to double cover aspergillus posaconazole ambisome vanc was changed to dapto for vre coverage bactrim was added for pcp

nametoken at approximately am on june housestaff were called to evaluate the patient after his nurse noticed his a line pressure locationtoken dropped abruptly to sbp in the 10s 20s on evaluation he was unresponsive and did not have palpable pulses

at that time back board was placed beneath patient and cpr was initiated

pulseless electrical activity was noted on the monitor at interval minute pulse checks no shockable rhythm

he received epinephrine mg iv x doses at five minutes apart

housestaff notified the patient 's wife who requested that the resuscitation be stopped

Figure 9.4

required po ativan as needed inactive issues hypertension the patient was generally normotensive with sbps ranging off of medication

her home antihypertensives were held on admission at the recommendation of hematology though chlorthalidone lisinopril and atenolol have not been commonly associated with thrombocytopenia there have been case reports of low platelets with chlorthalidone and captopril with a plan to restart one medication at a time once platelets become stable

arrhythmia patient reported a history of arrhythmia on admission which she states is the reason she uses the atenolol

the arrhythmia seems most likely due to palpitations from pvcs based on limited documentation in hospitaltoken

primary care and cardiology notes

she was monitored on telemetry in the icu and other than sinus bradycardia to the 50s with sleep no arrhythmias were noted

she remained asymptomatic osa patient reported using cpap at home but did not know her settings

she was seen by the respiratory therapist who selected settings that resulted in good quality sleep in house per patient report

she required continuous o2 monitoring per hospital protocol although she eventually requestd it be removed

medications on admission atenolol mg po daily chlorthalidone mg po daily

lisinopril mg po daily cholecalciferol vitamin d3 unit po daily when remembers

vitamin b12 po daily when remembers discharge medications atenolol mg tablet sig one tablet po once a day chlorthalidone

mg tablet sig one tablet po once a day lisinopril mg tablet sig one tablet po once a day cholecalciferol vitamin d3 unit tablet sig

Figure 9.5

ros negative for fevers chills nightsweats cough chest pain sob abdominal pain recent diarrhea rash joint pains or medications chagnes

past medical history dm2 on oral meds only not consistently taking htn cad s p ptca hyperlipidemia tah february

Figure 9.6

Chapter 10

Conclusion and Future Work

In this work we introduced a hierarhical neural model for automated coding of clinical notes. The architecture is composed by an embedding layer, a recurrent layer and a convolutional layer, with attentional layers at word and sentence level. The sentence level attentional matching is computed using embedded representation of the codes based on their textual descriptions.

We evaluated our model on the MIMIC-III dataset, surpassing the previous state of the art model according to several quantitative metrics. We also provided a qualitative evaluation of the results by inspecting how the model distributes the attention scores to different sections of documents when assigning a given code.

We performed a more accurate preprocessing of the input documents with respect to previous studies, resulting in similar quantitative metrics but with improved readability of the processed documents. We demonstrated that initializing the embedding matrix with word vectors trained on an external common English corpus and using textual decriptions of the codes for the attentional matching significantly improve the performance of the model. We showed that while our proposed hierarchical model has comparable performance with respect to a single layer convolutional model with a word level attentional matching, the results produced by the hierarchical model are more meaningful and interpretable upon qualitative evaluation.

We also evaluated our models using relaxed metrics, taking as target predictions only the codes headers instead of the full codes, resulting in a much lower labels cardinality.

The relaxed metrics show the potential use for these models as a support tool for ICD coding, as they can guide manual coders close to the correct subtree of the hierarchy, leaving to the operators the selection of the appropriate more specific codes. Moreover, highlighted documents similar to the ones used for our qualitative evaluation can be useful to reduce a physician's reading burden, as he is able to focus on the health related portion of the text.

This study shows the potential for real life applications in view of the high

performance even on noisy formatted data such as the ones contained in MIMIC-III. With more elaborate data preprocessing techniques and with more properly formatted clinical notes, the automatic coding can be even more accurate. For future work we plan to further improve the preprocessing stage of our pipeline, with particular attention to the sentence segmentation, and to use other datasets for the training and validation of our models.

Bibliography

Ba, Lei Jimmy, Kiros, Ryan, & Hinton, Geoffrey E. 2016. Layer Normalization. *CoRR*, **abs/1607.06450**.

Bahdanau, Dzmitry, Cho, Kyunghyun, & Bengio, Yoshua. 2014. *Neural Machine Translation by Jointly Learning to Align and Translate*. cite arxiv:1409.0473Comment: Accepted at ICLR 2015 as oral presentation.

Baumel, Tal, Nassour-Kassis, Jumana, Elhadad, Michael, & Elhadad, Noémie. 2017. Multi-Label Classification of Patient Notes a Case Study on ICD Code Assignment. *CoRR*, **abs/1709.09587**.

Cho, Kyunghyun, Van Merriënboer, Bart, Bahdanau, Dzmitry, & Bengio, Yoshua. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Choi, Eunsol, Hewlett, Daniel, Lacoste, Alexandre, Polosukhin, Illia, Uszkoreit, Jakob, & Berant, Jonathan. 2016. Hierarchical question answering for long documents. *arXiv preprint arXiv:1611.01839*.

Escudié, Jean-Baptiste, Saade, Alaa, Coucke, Alice, & Lelarge, Marc. 2018. Deep Representation for Patient Visits from Electronic Health Records. *CoRR*, **abs/1803.09533**.

Johnson, Alistair E. W., Pollard, Tom J., Shen, Lu, Lehman, Li-wei H., Feng, Mengling, Ghassemi, Mohammad, Moody, Benjamin, Szolovits, Peter, Celi, Leo A., & Mark, Roger G. 2016. MIMIC-III, a freely accessible critical care database. *Scientific Data*, **3**(May), 160035+.

Karmakar, Amitabha. 2018. Classifying medical notes into standard disease codes using Machine Learning. *CoRR*, **abs/1802.00382**.

Kavuluru, Ramakanth, Rios, Anthony, & Lu, Yuan. 2015. An empirical evaluation of supervised learning approaches in assigning diagnosis codes to electronic medical records. *Artificial Intelligence in Medicine*, **65**(05).

Kim, Yoon. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Kingma, Diederik P., & Ba, Jimmy. 2014. Adam: A Method for Stochastic Optimization. *CoRR*, **abs/1412.6980**.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, Haffner, Patrick, *et al.* . 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, **86**(11), 2278–2324.

Mikolov, Tomas, Chen, Kai, Corrado, Greg, & Dean, Jeffrey. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR*, **abs/1301.3781**.

Mnih, Volodymyr, Heess, Nicolas, Graves, Alex, & kavukcuoglu, koray. 2014. Recurrent Models of Visual Attention. *Pages 2204–2212 of:* Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., & Weinberger, K. Q. (eds), *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc.

Mullenbach, James, Wiegreffe, Sarah, Duke, Jon, Sun, Jimeng, & Eisenstein, Jacob. 2018. Explainable Prediction of Medical Codes from Clinical Text. *CoRR*, **abs/1802.05695**.

Pennington, Jeffrey, Socher, Richard, & Manning, Christopher D. 2014. GloVe: Global Vectors for Word Representation. *Pages 1532–1543 of: Empirical Methods in Natural Language Processing (EMNLP)*.

Perotte, Adler, Weiskopf, Nicole, Elhadad, Noémie, Pivovarov, Rimma, Natarajan, Karthik, & Wood, Frank. 2013. Diagnosis code assignment: models and evaluation metrics. *Journal of the American Medical Informatics Association*, **21**(2), 231–237.

Shi, Haoran, Xie, Pengtao, Hu, Zhiting, Zhang, Ming, & Xing, Eric P. 2017. Towards Automated ICD Coding Using Deep Learning. *CoRR*, **abs/1711.04075**.

Yang, Zichao, Yang, Diyi, Dyer, Chris, He, Xiaodong, Smola, Alex, & Hovy, Eduard. 2016. Hierarchical attention networks for document classification. *Pages 1480–1489 of: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Zhang, Ye, & Wallace, Byron. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.