# Automatic Dating of Textual Documents

**Rui Miguel Fernandes Figueira**

Thesis to obtain the Master of Science Degree in

**Telecommunications and Computer Engineering**

Supervisors: Prof. Doutor Bruno Emanuel da Graça Martins

Prof. Doutor Daniel Coelho Gomes

**Examination Committee**

**October 2017**

# Acknowledgements

First of all, I would like to thank to Professor Bruno Martins for the patience and guidance during the last year, through which he contributed significantly, and I am sure that without his knowledge and motivation this work would not have been possible.

I would also like to thank to my family for always being present and supportive during the last year. A special thanks to my parents to give me the opportunity to study in one of the best academies in Portugal, Instituto Superior Técnico.

Finally, I would like to thank to all my colleagues, who have gone through this adventure with me and I am sure that without them, it would be much harder for me to support all the effort that this work needed.

Rui Miguel Fernandes Figueira

# Resumo

Neste trabalho é abordada a atribuição automática de classes temporais a documentos textuais, ou seja, a tarefa de determinar a que período temporal determinado documento se refere, ou em que data o mesmo foi escrito. O método apresentado, baseado numa rede neuronal profunda, tem apenas em conta palavras ou pistas presentes no texto. A rede neuronal proposta visa a exploração da natureza hierárquica dos inputs considerados (i.e., os documentos são modelados como sequências de frases, que por sua vez são sequências de palavras), combinando embeddings pré-treinados, recurrent units e mecanismos de attention para gerar representações intermédias para os conteúdos textuais. De forma a validar o modelo apresentado, este trabalho apresenta também uma série de resultados experimentais com quatro datasets com diferentes características e que nos ajudam a cruzar os nossos resultados com os resultados previamente publicados para os referidos conjuntos de dados.

# Abstract

This work addresses the automated dating of textual documents, i.e. the task of determining when a document is about or when it was written, based only on its text. We rely solely on temporal cues implicit in the text, and advance over previous work in the area by proposing a method based on a deep neural network. The proposed neural architecture explores the hierarchical nature of the input data (i.e., documents are modeled as sequences of sentences, which in turn correspond to sequences of words), combining pre-trained word embeddings, recurrent units and neural attention, for generating intermediate representations of the textual contents. To validate the presented model, this work also presents a series of experimental results in four datasets, with different characteristics and which help us to cross our results to the ones previously presented.

# Palavras-Chave
Keywords

## Palavras-Chave

Datação Automática, Mecanismo de Atenção, Rede Neuronal Profunda, Word Embeddings

## Keywords

Automatic Dating, Attention Mechanism, Deep Neural Network, Word Embeddings

# Contents

# List of Figures

# List of Tables

# 1 Introduction

Temporal text mining has been, and continues to be, an active research area, both within the natural language processing (e.g., studies addressing fine-grained temporal ordering of events (Derczynski, 2017), or the recognition and contextual disambiguation of temporal expressions in textual documents (Strötgen and Gertz, 2016)) and information retrieval communities (e.g., studies addressing the temporal ranking and presentation of search results (Kanhabua et al., 2015; Campos et al., 2014)). This document presents a M.Sc. thesis addressing a less explored temporal text mining problem, namely the automated dating of textual documents based solely on clues implicit in the textual contents (i.e., the task of attributing a temporal label to a document, based only on its text). This document presents a new model for automatic dating of textual documents validated with four different datasets, with different sizes and considering documents of different lengths, and considering also different temporal periods. These include (a) data from the SemEval 2015 shared task on diachronic text evaluation (Popescu and Strapparava, 2015), (b) the RetroC Polish corpus designed for evaluating temporal classifiers (Gralinski and Wierzchon, 2015), (c) documents collected for Wikipedia (e.g., biographies) (Jatowt et al., 2013) and (d) short stories collected from project Gutenberg.

## 1.1 Motivation

In the field on temporal text mining, many previous studies have been made. Most of them leveraged generative approaches based on probabilistic language models (Kumar et al., 2011) or discriminative approaches based on sparse representations and linear classifiers (Niculae et al., 2014), we formulate the problem as a multi-class text classification task, discretizing the timeline into contiguous atomic time spans (i.e., the classes correspond to these discrete chronons), and attempting to infer the chronon that best corresponds to a representation of the text.

Since textual documents have a hierarchical structure (i.e., sentences are composed by words, and documents and then composed by sentences), is natural to approach the dating task considering that some sentences can be more relevant than others, and even inside each sentence, some words can me more time-relevant than others.

However, the attention concept (Yang et al., 2016) was never leveraged in a temporal text mining context. This mechanism, which tries to enhance the more relevant words of each sentence and/or document, can possibly improve the overall performance of an automatic dating method because, also in the dating process, not all parts of a document are equally important to the dating task.

With this new approach, exploring methods based on state-of-the-art deep neural network architectures together with the attention mechanism, we believe that our model can perform as well as the state-of-the-art results presented in the four previously cited datasets.

## 1.2 Proposed Methodology

The proposed model combines different mechanisms for generating intermediate representations from the textual inputs, including bi-directional Gated Recurrent Units (GRUs) for modeling sequential data (Cho et al., 2014), averages of word embeddings similarly to the proposal by Joulin et al. (2016), and neural attention mechanisms for highlighting relevant parts of the inputs (Bahdanau et al., 2014). Taking inspiration on the previous work by Yang et al. (2016), the proposed network architecture explores the hierarchical nature of the input data, combining two levels of GRUs and neural attention (i.e., sentence level and document level). The representations produced with this hierarchical attention approach are concatenated with a simple average of the embeddings for all the words in the input, and then passed to feed-forward nodes that output the most likely chronon.

Three output nodes are considered on the model, in an attempt to further improve results. These correspond to (i) a softmax node that outputs a year, associated to a categorical cross-entropy loss function, (ii) a softmax node that outputs the corresponding decade, also associated to a categorical cross-entropy loss, and (iii) a linear node that outputs the year, although in this case considering a loss function corresponding to the mean squared error between the predicted and ground-truth years. The entire model can be trained end-to-end from a set of labeled documents, leveraging the back-propagation algorithm in conjunction with the Adam optimization method (Kingma and Ba, 2015).

## 1.3 Results and Contributions

The obtained results with the four different datasets previously cited confirm that the proposed approach has consistent results across a diverse set of prediction tasks (e.g., for document collections spanning hundreds and thousands of years, or much shorter temporal periods), confirming the usefulness of the implicit temporal clues available in general textual contents. We test not only the full model proposal, but also the two branches of the model individually (i.e., we perform tests considering only the average of the embeddings of the inputs, and in the other side, considering only the hierarchical attention approach). Our full model corresponds to accuracy values of 35.93%, 14.72%, 40.66% and 17.17%, and mean absolute error values of 36.58, 16.29, 5.27 and 16.47 years, respectively on the SemEval, Wikipedia, Gutenberg and RetroC texts. Together, these results show that a modern neural architecture for text classification can at least approach and in some cases outperform previous methods and that, even in the absence of temporal extraction resources, it is possible to achieve good results across a diverse set of texts.

The main contribution of this M.Sc. thesis is this novel model which introduces the attention mechanism in the automatic dating tasks. This approach was never taken into account by the temporal text mining community to this type of tasks.

We also performed tests with 4 different datasets with gives us a good basis to support and validate our model. With such different datasets (i.e., with different values in characteristics like sentence length, document length, number of words, etc.) we prove that our model is resilient enough to achieve good results is datasets with different characteristics.

## 1.4 Structure of the Document

The rest of the M.Sc. thesis document is organized as follows: Chapter 2 surveys previous related work. Chapter 3 details the proposed approach, presenting the architecture of the deep neural network that was considered for addressing document dating as a supervised classification task. Chapter 4 presents the experimental evaluation of the proposed method, detailing the datasets, the evaluation methodology, and the obtained results. Finally, Chapter 5 summarizes our main conclusions and presents possible directions for future work.

# 2 Concepts and Related Work

The following chapter details the fundamental concepts and previous studies made in the temporal resolution area. Section 2.1 details fundamental concepts related to the work presented in this document. Section 2.2 presents an overview of the most relevant studies presented in the document dating area.

## 2.1 Fundamental Concepts

This section presents relevant concepts for my M.Sc. thesis. Subsection 2.1.1 discusses how textual documents can be represented for computational analysis. Subsection 2.1.2 presents a short explanation on the naïve Bayes classifier, one of the simplest supervised classification approaches. Subsection 2.1.3 focuses on classic discriminative classifiers, particularly on simple perceptrons and multilayered neural networks. Subsection 2.1.4 addresses convolutional neural network classifiers. Finally, Subsection 2.1.5 focuses on recurrent neural network classifiers.

### 2.1.1 Representing Textual Documents for Classification

In text classification problems, documents are commonly represented through smaller components, like words or $n$-grams (i.e., sequences of $n$ consecutive words or characters). One of the possible ways to formalize this information is using the vector space model. In brief, the vector space model is an algebraic model to represent textual documents, where a weight is associated to each component (i.e., word or $n$-gram) of a document (Salton et al., 1975). Each document is thus represented as a feature vector $d_j = (w_{1,j}, w_{2,j}, \ldots, w_{t,j})$, where $t$ is the number of components (i.e., individual words or $n$-grams in the entire document collection), also referred to as features, and where $w_{i,j}$ is the weight of the feature $i$ in the context of the document $j$, representing its importance.

In fact, within the vector space model, the weight assigned to each feature can be calculated in several different ways. The simplest is a binary scheme, which sets $w_{t,j} = 1$ if the word or $n$-gram $i$ is present in the document $j$, or $w_{t,j} = 0$ otherwise. Another possibility, if we need a more accurate value for the weights of the features, is using the TF-IDF heuristic (Manning et al., 2008).

In brief, TF-IDF combines two different heuristics. The term frequency (TF) corresponds to the number of times that a term $t$ occurs in a document $d$, whereas the inverse document frequency (IDF) measures the importance of a word or $n$-gram in the context of a set of documents, saying if some term is rare or common in the collection. The IDF can be computed as follows:

$$\text{IDF}(t) = \log\left(\frac{td}{d_t}\right) \tag{2.1}$$

In the previous equation, $td$ is the total number of documents of the collection under study, and $d_t$ is the number of documents where the term $t$ appears. Finally, to compute the TF-IDF value, we multiply the TF and IDF scores, as shown in Equation 2.2.

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t) \tag{2.2}$$

To classify a given textual document according to a pre-defined set of classes, based on a training set of documents labelled according to the considered classes,, one of the simplest options that we have is using the $k$-nearest neighbours ($k$NN) algorithm. Given a training set, when classifying a given textual document $d_{test}$, the $k$NN algorithm returns the most common value among the $k$ training examples that are more similar to $d_{test}$.

To find the most similar training documents, we can leverage the cosine similarity between TF-IDF vectors. Since the test and training examples are represented as vectors, we can compute the cosine of the angle $\alpha$ between the test vector $d_{test}$, and each example $d_{train}$ in the training set, as shown below:

$$\cos\,(\alpha) = \frac{d_{test} \cdot d_{train}}{\|d_{test}\| \times \|d_{train}\|} \tag{2.3}$$

The standard vector space model produces representations for documents corresponding to high-dimensional sparse vectors, in which each individual term (i.e., each word or $n$-gram) is independent (i.e., orthogonal) from other terms. Alternative methods have also been developed,

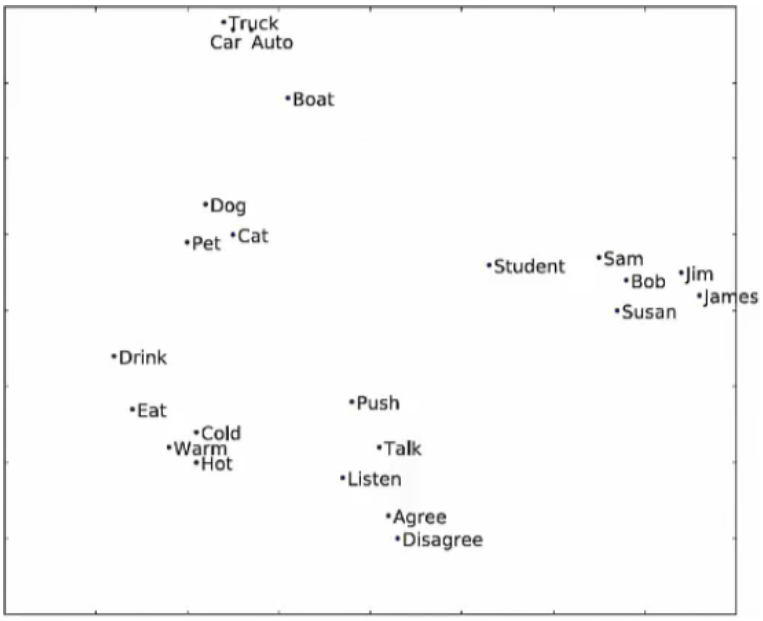


Figure 2.1: Two-dimensional projection of word embeddings

for instance based on producing a low dimensional representation of the document vector (e.g., through a singular value decomposition), or based on word representations (i.e., embeddings) that can capture semantic similarity between words.

The word embedding technique, first proposed by Bengio et al. (2003), is nowadays widely used in the NLP community. In this approach, the words in a document are also represented as high-dimensional vectors (e.g., vectors of 300 latent dimensions), and the words with similar vectors tend to be semantically similar. Figure 2.1 shows a simple vector space example where we can see, for instance, that words like *cat* and *dog* have similar vectors and they are both related to pets. The creation of word embeddings is based on the principle that words occurring in the same context tend to have a similar meaning.

The method uses a small window with $i$ words (i.e., the context of the term $t$) to predict the probability of a term $t$. In some cases the context is considered to be only the $i$ words before the term $t$, but one can considered also words occurring after the target word. The basic architecture of word embedding models is as follows:

- **Embedding Layer**: Layer that generates the word embeddings by multiplying an index vector with an embedding matrix.
- **Intermediate or Inner Layer(s)**: One or more layers that produce an intermediate representation of the input.
- **Softmax Layer**: Final layer that produces a probability distribution for the words in

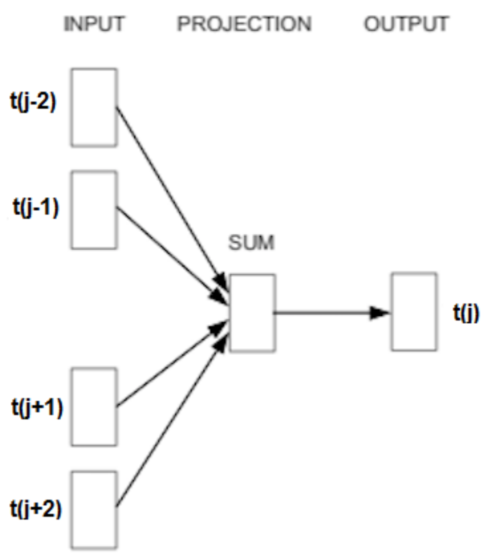


Figure 2.2: CBoW model architecture

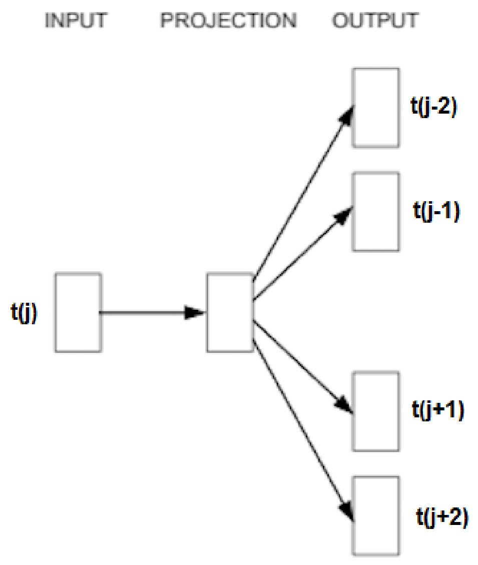


Figure 2.3: Skip-gram model architecture

the vocabulary. There are many approaches to build this final layer, like the hierarchical softmax (i.e., used in the fasttext solution presented in the related work section) or the differentiated Softmax (i.e., using a sparse matrix instead of a dense one).

Since Bengio et al. (2003) first presented the concept of word embeddings, many other models have been proposed, and one of the most popular approaches is the word2vec method by Mikolov et al. (2013). In their proposal, the authors introduce two new log-linear models:

- **Continuous Bag-of-Words (CBoW)**: In this bag-of-words approach (i.e., the order of the words in the history does not influence the prediction) uses the context of the term $t$ (i.e., the surrounding words within a window) to predict the term $t$. The architecture of the CBoW model, with a window size of 4 words, is shown in Figure 2.2.

- **Skip-gram**: The skip-gram model does precisely the opposite of the CBoW. With a term $t$ as input, this method tries to predict words within a window before and after the term $t$. The architecture of the skip-gram model, with a window size of 4 words, is shown in Figure 2.3.

This way to represent documents, among many advantages, allows us to easily compare documents using the distance between them (i.e., the distances between words within the documents). Kusner et al. (2015) proposed a method to measure similarity between documents leveraging word occurrence within documents together with word embeddings. The authors define the similarity between two documents as the inverse of their distance (i.e., two similar documents have a smaller document distance, while two non similar documents have a bigger document distance), and the document distance can be computed from the minimum weighted cumulative cost to move all words from one document to another.

To calculate the cost of moving one word to another, the authors propose a measure called word travel cost. This measure is simply the norm of the difference between the embeddings of the two words.

### 2.1.2 Naïve Bayes Classifiers

The naïve Bayes classifier is one of the simplest probabilistic classifiers. It is based on the Bayes theorem, which says that:

$$\mathrm{P}(c|d) = \frac{\mathrm{P}(d|c) \times \mathrm{P}(c)}{\mathrm{P}(d)} \tag{2.4}$$

The naïve Bayes classifier, also assumes that, given a document $d$, each one of the terms $\{t1, \ldots, t_k\}$ that compose the document are independent from each other, and thus we can simplify the previous formula as follows:

$$\mathrm{P}(c|d) \simeq \mathrm{P}(c) \prod_{k=1}^{n_d} \mathrm{P}(t_k|c) \tag{2.5}$$

In Equation 2.5, $\mathrm{P}(t_k|c)$ is the conditional probability of term $t_k$ occurring in a document of class $c$, while $\mathrm{P}(c)$ is the prior probability of a document occurring in class $c$ (i.e., the probability of a document occurring in the class c, without taking into account the terms present in the document).

In text classification, the goal is trying to find, as accurately as possible, the best class for each document (i.e., in the naïve Bayes method, the maximum a posteriori class, which is simply the class $c$ with the highest value of $\mathrm{P}(c|d)$).

The parameters $\mathrm{P}(c)$ and $\mathrm{P}(t_k|c)$ need to be estimated based on a training set. For that, we use the maximum likelihood estimate, which is given by the relative frequency and corresponds to the most likely value of each parameter in the training data. For instance:

$$\mathrm{P}(c) = \frac{d_c}{td} \tag{2.6}$$

In the previous equation, $\mathrm{P}(c)$ is the relative frequency of class $c$ in the total number of classes, $d_c$ in the number of documents belonging to class $c$, and $td$ is the total number of documents of the collection under study. Similarly $\mathrm{P}(t|c)$ is the relative frequency of term $t$ in documents belonging to class $c$, being calculated as follows:

$$\mathrm{P}(t|c) = \frac{n_{c,t}}{\sum_{t'} n_{c,t'}} \tag{2.7}$$

In the previous equation, $n_{c,t}$ is the number of occurrences of therm $t$ in documents of the class $c$, while $\sum_{t'} n_{c,t'}$ represents, the total number of terms in the class $c$.

In some cases, we have words that do not occur in all classes, and this leads to parameters where $\mathrm{P}(t|c) = 0$. To overcome this issue, we need to introduce parameter smoothing, for instance through the Laplace smoothing method, which gives a small and equal probability to all the unknown words (Juan and Ney, 2002). In this case, the parameter $\mathrm{P}(t|c)$ would be calculated as follows.

$$\mathrm{P}(t|c) = \frac{n_{c,t} + 1}{\sum_{t'} n_{c,t'} + |V|} \tag{2.8}$$

In the previous equation $|V|$ is the number of words of the vocabulary (i.e., the length of the vocabulary vector).

### 2.1.3 Discriminative Classifiers

Discriminative classifiers learn a direct map of inputs to the class labels (Ng and Jordan, 2002). The single-layer perceptron is one of these classifiers. Initially proposed by Rosenblatt (1958), this model offers a simple way to classify linearly separable patterns. The classification model corresponds to:

$$\hat{\mathrm{c}} = w_0 + \sum_{k=1}^{n} t_k w_k \tag{2.9}$$

$$\varphi = \begin{cases} 1 & \text{if } \hat{c} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

In Equation 2.9, $\hat{c}$ is the predicted class, based on inputs $\{t_1, \ldots, t_k\}$ and weights $\{w_0, \ldots, w_k\}$. Initially, the weights of each parameter are initialized at random values, and they are then updated during the training process. In each iteration of the algorithm, if the produced output $\hat{c}$ is correct (i.e., equal to the desired output $c$) we keep the weights, otherwise they are updated in the direction of correctly classifying the training instances. This process is repeated until convergence, or until a predetermined training threshold is obtained. In each time that the prediction does not match the correct output, in order to update the weights, an update $\Delta$ is calculated as shown in Equation 2.11 and them added to the previous weight.

$$\Delta_{w_k} = \eta(\hat{c} - c)t_k \tag{2.11}$$

In the previous equation $\eta$ is a predetermined learning rate. After the training is complete, the perceptron is able to associate each input to the correct output class.

For classifying patterns that are not linearly separable, we can rely on a multilayer perceptron (MLP), which is a neural network with 3 or more layers (i.e., one input layer, one or more hidden layers, and one output layer), as shown in Figure 2.4. This method is similar to the single-layer perceptron, and the only major difference is in the parameter update method. Another of the differences is that, in the cap of an MLP, the most common ativation function is the sigmoid, as shown in the Equation 2.12.

$$\varphi(v) = \frac{1}{1 + \exp(-v)} \tag{2.12}$$

When training a MLP, the update is made based on a method called backpropagation. The method is divided into three main stages:

- **Propagation**: Input values are propagated through the neural network. In each neuron, we have a $v$ value which is computed from the input of each neuron through a weighted aggregation as shown in Equation 2.9. The output of each neruon is given by Equation 2.12. This procedure is followed until we obtain the output of the last layer of the MLP.
- **Backpropagation**: Using the output value $\hat{c}$ and the desired value of the output $c$, we

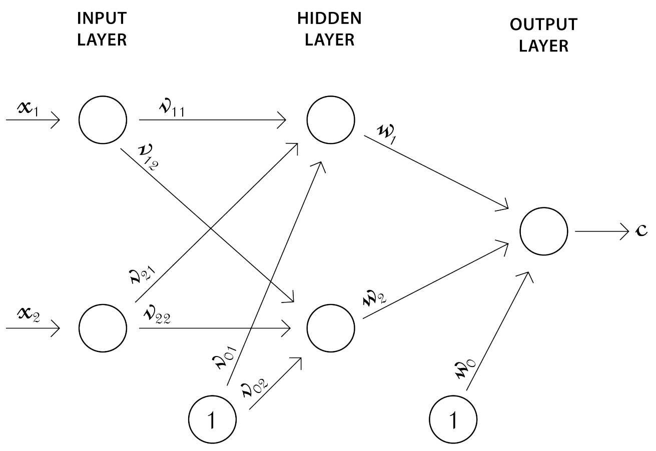


Figure 2.4: Example of a multilayer perceptron.

propagate information backwards through the neural network in order to calculate the deltas in each neuron. The update $\Delta$ of the output neurons is given by:

$$\Delta_{Oi} = \varphi'(v_i) \times (c_i - \hat{c}_i) \tag{2.13}$$

In turn, in the inner layers, we need to take into account the multiple connections, so the update $\Delta$ is given by:

$$\Delta_{I_i} = \varphi'(v_i) \times \left( \sum_{i=1}^{n} \Delta_{O_i} \times w_i \right) \tag{2.14}$$

- **Updating Weights**: With the updates $\Delta$ calculated in the previous step, we are now able to update the weights of the connections between nodes, using the expression below, and assuming $n$ as the iteration number:

$$\mathrm{w_i}(n+1) = w_i(n) + \eta \times \Delta_i(n) \times c \tag{2.15}$$

In the equation above, $\eta$ is a learning rate and it should be set in the beginning of the training.

The aforementioned steps should be repeated in order to minimize the error between the predicted and desired value.

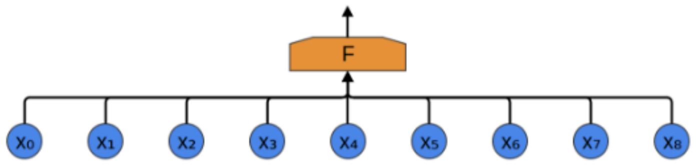


Figure 2.5: A simple convolutional neural network with only two layers

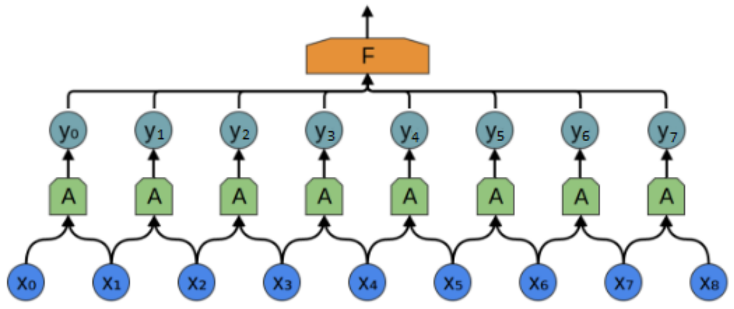


Figure 2.6: A CCN with multiple layers and a window of two inputs

### 2.1.4 Convolutional Neural Network Classifiers

Convolutional neural networks (CNNs) can be understood as a type of neural network that uses many times identical copies of the same neuron. With this type of network, and unlike the MLP, we can have an unbounded number of inputs. This particular network architecture has been extensively used in NLP applications to model sequences of words (Zhang and Wallace, 2015).

CNN's are composed by a variable number of layers, but the first one is always, in the context of NLP, the vector representation of the sequence (e.g., the sentence) under analysis, where each input value $x$ is a word or $n$-gram.

The simplest and more commonly used approach, with only two layers, involves connecting all the input values to a fully-connected layer, as shown in Figure 2.5. In a more sophisticated approach, we can create an intermediate layer. In this layer, each neuron $A$ will look only to a small segment of the inputs. This segment or window can have a variable size, representing the observed segments as features, that will feed the fully connected layer, as shown in Figure 2.6.

Considering only 1 neuron A like is shown in Figure 2.6 , the output $c_0$ is given by the next equation.

$$c_0 = A(x_0, x_1) \tag{2.16}$$

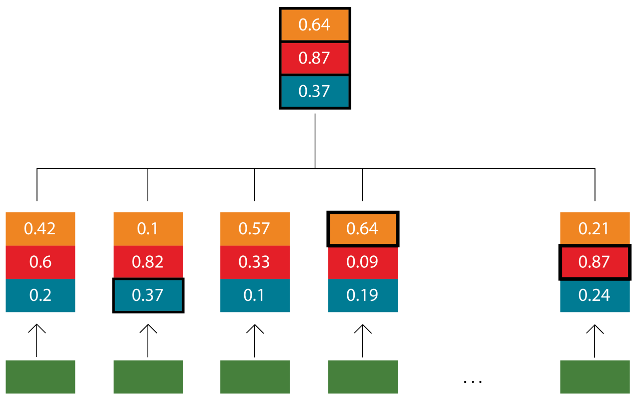


Figure 2.7: Max-over-time pooling step

Convolutional layers can be iterweaved or followed by a pooling layer. Pooling layers are used to reduce the spatial size of the representation, thus effectively reducing the amount of parameters in the network, and the computational effort associated to the operation.

In the NLP context, it is usual to have sentences with variable length. For modeling this, one of the most widely used approaches is the max-over-time pooling approach (Collobert et al., 2011). The idea of this pooling scheme is to capture the most important feature for each feature map. Figure 2.7 illustrates this procedure. As we can see, we can have any unknown number of words (or windows of words), represented in green, but the final result will always be a fixed size feature map. With this procedure, we overcome the problem of having variable size inputs.

### 2.1.5 Recurrent Neural Network Classifiers

In regular neural networks, the inputs are assumed to be independent from each other but it is not always truth. Recurrent Neural Networks (RNNs) where designed to handle input sequences where the output of each node is dependant on previous computations on the other elements of the sequence. Figure 2.8 presents an example of an RNN where each node receives an input $x_i$ and produces an output $y_i$, passing that information to the next node of the network.

One problem in standart RNNs is that they begin to struggle with long input sequences. Some different approaches addressed this issue. Long Short-Term Memories (LSTMs) proposed by Hochreiter and Schmidhuber (1997) and Gated Recurrent Units (GRUs) proposed by Chung et al. (2014) are two approaches where the influence of the previous information is controlled

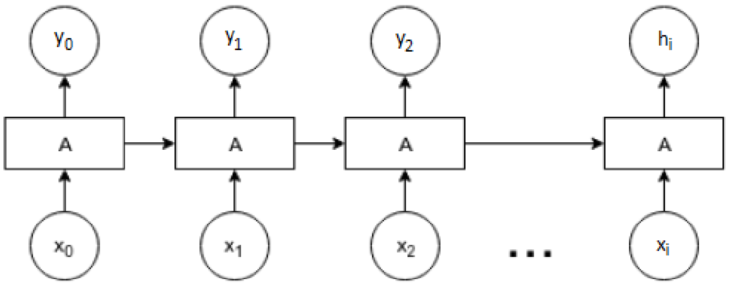


Figure 2.8: Simple Recurrent Nerual Network example

trough gating mechanisms. Since the gates determine how much of the previous state vectors information should be kept, this types of networks can better handle long input sequences by learning how the internal memory should behave.

## 2.2 Related Work

This section describes some of the most relevant previous studies related to automated document dating. The section also describes recent studies describing state-of-the-art methods for text classification, which can be used for document dating.

### 2.2.1 Language Modeling Methods

Kumar et al. (2011) presented a simple proposal for the temporal resolution of text, based on the concept of maximum likelihood estimation presented in Subsection 2.1. The proposed method starts by dividing the overall temporal range that was considered into discrete units with a pre-determined granularity. In their experiments, the authors determined that each *chronon* (i.e., the atomic interval $c$ upon which a discrete timeline is constructed) corresponds to a single year, but the granularity can be adjusted to other values.

Whereas previous proposals like those of Jong et al. (2005) or Kanhabua and Nørvåg (2008) measured the similarity between a document and each *chronon* through a normalised log-likelihood ratio measure (NLLR), in this study the authors have chosen to compute the unnormalized likelihood of some *chronon* $c$ given the document $d$ using the Kullback–Leibler divergence $\text{KLD}(\Theta^d||\Theta^c)^{-1}$, where $\Theta^d$ is the probability distribution of the test document, and $\Theta^c$ is the probability distribution of the pseudo-document $d^c$ corresponding to a *chronon*.

$$\mathrm{P}(c|d) \simeq \frac{\mathrm{KLD}(\Theta^d||\Theta^c)^{-1}}{\sum_c \mathrm{KLD}(\Theta^d||\Theta^c)^{-1}}, \text{ with } \mathrm{KLD}(\Theta^d||\Theta^c) = \sum_i \Theta^d(i) \log\left(\frac{\Theta^d(i)}{\Theta^c(i)}\right) \tag{2.17}$$

A pseudo-document $d^c$ is generated for each *chronon* $c$ by compiling all training documents whose labeled span overlaps $c$. Then, using the pseudo-document of each *chronon*, the authors measure the affinity between a test document and a specific span of time, as shown in Equation 2.17.

Regularization is made with Dirichlet smoothing (Zhai and Lafferty, 2004). However, instead of giving a predetermined prior (i.e., initial value) for all documents and *chronons*, the authors determined a specific prior value. With $|V_{d^c} \cup V_d|$ being the size of the vocabulary of the pair document-*chronon*, the Dirichlet smoothing is performed as shown bellow:

$$\hat{\theta}_t^d = \eta \frac{n_{d,t}}{|d|} + (1-\eta)\frac{1}{|V_{d^c} \cup V_d|} \text{ ,with } \eta = \frac{|d|}{|d| + \mu} \tag{2.18}$$

In the previous equation, $n_{d,t}$ is a counting function which gives the number of occurrences of the term $t$ in the document $d$, and $|d|$ is the number of words in the document $d$. Instead of using directly the hyper-parameter $\mu$, the authors choose to introduce another hyper-parameter $\lambda$, where $\mu = \frac{\lambda}{|V_{d^c} \cup V_{d_i}|}$.

The reason for using the aforementioned approach is that, in this case, we only consider words that are present in the document $d$, or in the pseudo-document of the *chronon* $d^c$, ignoring all other words. This can be especially useful in small documents, which have few words compared to the full collection.

To choose the best *chronon* $c$ for each document, the authors simply select the most-likely *chronon* under $\mathrm{P}(c|d)$.

For evaluation, the authors used 678 Gutenberg short stories[1] associated to the publication year, divided in two groups and with the learning set composed of 333 short stories, where the test set was composed of 345 short stories. To tune the parameters the author used an external corpus of biographies found on Wikipedia, considering only individuals whose complete lives

[1]`http://www.gutenberg.org/wiki/Short_Stories_(Bookshelf)`

occurred between 3800 B.C. and 2010 A.D.. The midpoint between the birth and death dates is set as the ground truth for each biography.

When predicting a single year for a document, a natural error measure between the predicted year $\hat{y}$ and the actual year $y^*$ is the difference $|\hat{y} - y^*|$. The comparison baseline in the Gutenberg corpus was the midpoint of the publication date (i.e., 1903).

Analysing the results over the Wikipedia biographies collection when tuning the *chronon* size between $\{1, 2, \ldots, 100\}$, the authors concluded that the minimum mean error is obtained with a *chronon* span of 40 years. After that, and using the same *chronon* span on the Gutenberg collection, the authors compared their approach with the baseline method, having managed to reduce the mean prediction error from 36 to 28 years.

### 2.2.2 Graph-Based Methods

In some cases, documents do not have explicit temporal expressions and this can pose difficulties to automated document dating. Jatowt et al. (2013) proposed a method to overcome that problem. The basic idea of their approach is to generate a co-occurrence graph reflecting the relations between words and dates, and then using the generated graph to attribute the most suitable date to each test document.

To create the graph, the authors formed 5 news article collections with documents collected from the Google News Archive[2]. Each of this collections represents one different country (i.e., Germany, UK, Japan, France and Israel). These 5 collections, with an average of 100k news articles each, were used to create 5 different co-occurrence graphs. Based on each collection, the authors constructed a graph $G(V, E)$ where $V$ is the set of vertices, with each vertice representing a unique word, while $E$ is the set of relationships between words.

The creation of the co-occurrence graph is divided into two stages: the calculation of word-time associations and the estimation of the temporal weights.

The authors define two different relations. Each direct association (i.e., between two words co-occurring in same sentences) has a weight that is calculated as show in Equation 2.19:

$$\mathrm{A}_{\mathrm{dir}}(t_i, t_j) = \frac{\mathrm{count}(t_i, t_j)}{\mathrm{count}(t_i) + \mathrm{count}(t_j) - \mathrm{count}(t_i, t_j)} \tag{2.19}$$

[2]`http://news.google.com/archivesearch`

In the previous equation, $\mathrm{count}(t_i, t_j)$ is the number of sentences where the terms $t_i$ and $t_j$ co-occur, while $\mathrm{count}(t_i)$ and $\mathrm{count}(t_j)$ are the number of sentences containing $t_i$ and $t_j$, respectively.

At the same time, and because some words do not co-occur with the dates of the documents, the authors defined another type of relationship. The context-based association is based on the intuition that, if a word $t$ is related with many words that are related with the time $c$, the word $t$ tends to be also related with the time $c$. The weights for these context associations are calculated as follows:

$$\mathrm{A}_{\mathrm{con}}(t_i, c) = \frac{1}{|V|} \sum_{j=1}^{|V|} \mathrm{A}_{\mathrm{dir}}(t_j, t_i)^2 \times \mathrm{A}_{\mathrm{dir}}(t_j, c) \tag{2.20}$$

In the previous equation $|V|$ is the number of vertices in the graph.

To estimate the temporal weights used to estimate the document focus time, the authors put forward and hypothesis which states that "A word has high discriminative capability for determining document focus time if it has strong association with only few time points and weak association with other time points.". Based on this, the authors propose two approaches:

- **Temporal entropy** weights, which are defined from the entropy over the association scores of the word with all the time points, calculated as follows.

$$w_t^{temE} = \max_j(\mathrm{E_j}) - \mathrm{E_t}, \text{ with } \mathrm{E_t} = -\sum_i \mathrm{P}_t(c_i) \ln \mathrm{P}_t(c_i) \tag{2.21}$$

- **Temporal kurtosis** weights, which apply the kurtosis measure on the word-time association scored, and are calculated as follows.

$$w_t^{temK} = \frac{\sum_i (\mathrm{A}(t, c_i) - \mu)^4}{\mathrm{count}(c)\sigma^4} \tag{2.22}$$

In the previous equation, $\mathrm{count}(c)$ is the total number of time points under analysis, while $\mu$ and $\sigma$ denote the mean and standard deviation of the associations of $t$ with the time points, respectively.

To calculate the document focus time based on the graph and weights, the authors proposed 4 approaches to associate the document to the most suitable time: The first one, as presented in Equation 2.23, is based on averaging time associations of terms contained in the document, using all the unique words.

$$\mathrm{S_U}(d,c)\frac{1}{|d|}\sum_{t\in d} w_t^{tem}\mathrm{A}(t,c) \tag{2.23}$$

The second approach is simply an extension of the first one using the notion of term frequency and calculated as follows.

$$\mathrm{S_{TF}}(d,c)\frac{1}{n_{d,t}}\sum_{t\in d} w_t^{tem} n_{d,t}\mathrm{A}(t,c) \tag{2.24}$$

In the previous equation, $n_{d,t}$ is the number of times the term $t$ occurs in the document $d$.

The third approach is also an extension of the previous ones, introducing the notion of word's importance. The word's importance $w_t^{imp}$ is the normalization of the TextRank (Mihalcea and Tarau, 2004) score of each word in a document, calculated as follows.

$$\mathrm{S_{TR}}(d,c) = \frac{1}{n_{d,t}}\sum_{t\in d} w_t^{imp} w_t^{tem} n_{d,t}\mathrm{A}(t,c) \tag{2.25}$$

The fourth approach, called by the authors as the extended version, is slightly different. The authors first extract all the dates in the document, and generate a document-time association only with the extracted dates. $\mathrm{S_{DATE}}(d,c)$ denotes the score of a time $c$ calculated in this way. This method is finally combined with any of the previous three as follows.

$$\mathrm{S_{EXT}}(d,c) = \mathrm{S_{DATE}}(d,c) + \mathrm{S}(d,c) \tag{2.26}$$

The document focus time is the time point with highest association score, as shown in Equation 2.27, with $\mathrm{S}(d, c)$ being any of the four proposals seen before corresponding to Equations 2.23, 2.24, 2.25 or 2.26.

$$\mathrm{c}(d) = \arg\max_{t} \mathrm{S}(d, c) \tag{2.27}$$

In their experiments the authors managed to achieve an average error of evaluation of 16.1 years, without using the extended version that incorporates the temporal expressions. When using the extended version, the authors were able to achieve a much lower average error, respectively 2.83 years.

In another recent study, Mishra and Berberich (2016) proposed a method that generates an event graph with excerpts of news articles as nodes, and with various inter-excerpt relations as edges. This graph was then used for automatically dating the excerpts. From a given set of excerpts describing events, the authors estimate an excerpt-time model capturing the temporal scope of each excerpt.

The authors have initially randomly selected 100 Wikipedia events. For each event, they obtained the top-10 documents using a standard text retrieval model. This resulted in an average of 150 excerpts in each document. From the complete collection of excerpts $R$ the authors defined a sub-set $R_{seed}$ composed by excerpts containing temporal expressions.

To generate the event graph, the authors start by estimating an empirical excerpt-time model for each excerpt in $R_{seed}$ from their implicit time expressions. For the remaining excerpts in $R$ the time models are initialized to a uniform distribution. A time model for an excerpt can be understood as a probability distribution that captures the temporal period of the excerpt. The graph is generated considering all $R$ entries as nodes, and their relationships as weighted edges. These relationships try to reflect similarity between the dates of the nodes. Two nodes with similar dates will have a closer relationship (i.e., higher weight) than tw nodes with distant dates (i.e., lower weight). Finally, the empirical time models from the excerpts with temporal expressions are propagated to the ones without any temporal references, which are strongly related to them.

An example of a simple graph can be seen in the Figure 2.9. In this example we have three

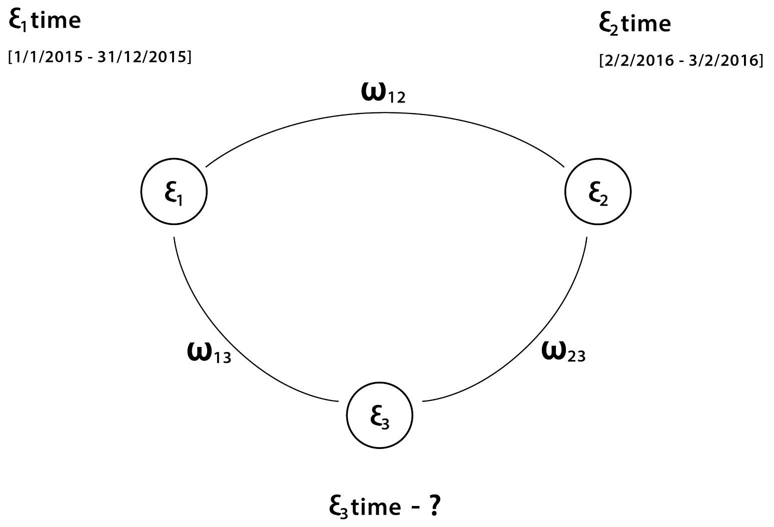


Figure 2.9: Example of a graph generated through this model

different excerpts all related to the same event. In $\varepsilon_1$ and $\varepsilon_2$ there exist implicit time expressions, referring to the dates present in the figure. On the other hand, in $\varepsilon_3$ there is no implicit time expression. In this case, the $\varepsilon_3$ time is estimated through the weights and nodes connected to it.

The excerpt models adopted in this study were:

- **Excerpt-Text Model** $E_{text}$ refers to a language model estimated from the $\varepsilon_{test}$, where $\varepsilon_{test}$ is the excerpt text part. The generative probability of a term $t$ from the excerpt-text model $E_{text}$ is given by:

$$\mathrm{P}(t|E_{text}) = (1-\lambda)\cdot\mathrm{P}(t|\varepsilon_{text}) + \lambda\cdot[\beta\cdot\mathrm{P}(t|d_{event}) + (1-\beta)\cdot\mathrm{P}(t|d_{context})] \qquad (2.28)$$

In Equation 2.28, $d_{event}$ is the textual description of the event collected from the Wikipedia Current Event[3] portal, and $d_{context}$ is the source document. Since the method proposed by the authors uses only a subset of terms, the probability must be re-normalized as follows:

$$\hat{\mathrm{P}}(t|E_{text}) = \frac{\mathrm{P}(t|E_{text})}{\sum_{t'}\mathrm{P}(t'|E_{text})} \qquad (2.29)$$

[3] `http://en.wikipedia.org/wiki/Portal:Current_events`

- **Excerpt-Time Model** $E_{time}$ is the probability distribution that captures the time interval for the event described in the excerpt. The probability of any time unit $\tau$ from the time model $E_{time}$ is obtained trough the iteration over all the temporal expressions, as shown in Equation 2.30.

$$\mathrm{P}(\tau|E_{text}) = \sum_{[tb,te]\in\varepsilon_{time}} \frac{\mathbb{1}(\tau \in [tb_l, tb_u, te_l, te_u])}{|[tb_l, tb_u, te_l, te_u]|} \tag{2.30}$$

  In the previous equation, $tb_l$ and $tb_u$ are respectively the lower and upper bounds of the beginning time of the event, and $te_l$ and $te_u$ are respectively the lower and upper bounds of the ending time of the event. The indicator function $\mathbb{1}$ returns 1 if there exists an overlap between the time unit and the the interval, and 0 otherwise. The equation is also re-normalized, as in Equation 2.29.

The weight between each two nodes of the graph represents the relationship strength between them, and is measured as follows.

$$w_{i,j} = \exp\left(\frac{\delta_{i,j}^2}{\sigma^2}\right) \tag{2.31}$$

In Equation 2.31, $\delta_{i,j}$ is the similarity between the excerpts $\varepsilon_i$ and $\varepsilon_j$, and $\sigma$ is a scaling parameter, set as the average similarity between the excerpts.

The authors proposed a similarity measure given by the sum of the text similarity $\delta_{text\ i,j}$, positional similarity $\delta_{pos\ i,j}$, conceptual similarity $\delta_{cep\ i,j}$ and contextual similarity $\delta_{ctx\ i,j}$. The position similarity is measured as proposed by Tao and Zhai (2007) and shown in Equation 2.32 , while all of the other measures are calculated trough Jensen-Shannon divergence, calculated as shown in Equation 2.33. Notice that the Jensen-Shannon divergence essentially corresponds to a symmetric and normalized version of the previously introduced Kullback–Leibler divergence.

$$\delta_{pos\ i,j} = \begin{cases} \log(a + \exp(-\text{Dist}(\varepsilon_i, \varepsilon_j))) & \text{if } \varepsilon_i, \varepsilon_j \in d \\ 0 & \text{otherwise} \end{cases} \tag{2.32}$$
$$\text{with Dist}(\varepsilon_i, \varepsilon_j) = \max(\text{pos}(\varepsilon_i), \text{pos}(\varepsilon_j) - \min(\max(\text{pos}(\varepsilon_i, \varepsilon_j))))$$

$$\begin{aligned} \text{JSD}(E_{i\ text} \| E_{j\ text}) = \frac{1}{2}\text{KLD}(E_{i\ text} \| \text{M}) + \frac{1}{2}\text{KLD}(E_{j\ text} \| \text{M}), \\ \text{with M} = \frac{1}{2}(E_{i\ text} + E_{j\ text}), \end{aligned} \tag{2.33}$$

Finally, the distribution propagation method used by the authors is based on the label propagation algorithm proposed by Zhu et al. (2003). They argue that if two excerpts have a strong inter-excerpt relationships between them, then they may refer to the same event, and hence have a similar temporal scope.

The authors start by collecting the set of excerpts $R$, and extract the earliest and latest time from them. Then, they estimate the $E_{time}$ for each excerpt in the collection $R_{seed}$, and add this information to the labeled class $c_l$ (i.e., lower class). For the remaining excerpts, the authors assume that the probability of any time unit $\tau$ is uniform, and add this information to the labeled class $c_u$ (i.e., upper class). Then, the graph is constructed with the events as nodes, and the weights for the edges are calculated. Finally, the algorithm iterates two steps until convergence: first, all excerpts propagate their time models, and second the excerpt-time modes in $c_l$ are reinitialized. After convergence, the excerpt-time models from the unlabeled excerpts can be retrieved.

To evaluate the proposed method, the authors performed a leave-one-out cross validation experiment, with $R_{test}$ corresponding to the test excerpt, and with $R_{seed}$ corresponding to all the remaining ones. To compare the quality of results the authors have used 6 different measures:

- **Model Quality** to measure how close an excerpt-time model $E_{time}$ is to the $\varepsilon_{time}$.

$$\text{MQ} = \frac{1}{|R_{test}|} \sum_{\varepsilon \in R_{test}} -\text{KLD}(\varepsilon_{time} \| E_{time}) \tag{2.34}$$

- **Generative Power** measures the likelihood of generating the original time intervals *time* in the time part $\varepsilon_{time}$ from the estimated time model $E_{time}$ after propagation.

$$\text{GP} = \sum_{\varepsilon \in R_{test}} \left( \frac{1}{|\varepsilon_{test}|} \sum_{time \in \varepsilon_{test}} \text{P}(time|E_{time}) \right) \tag{2.35}$$

- **Precision** The authors define for each excerpt a ranked list *Rank* of temporal expressions, based on their generative probabilities. They also define the binary function Rel($time$, $time_{estimated}$) which gives 1 if there exists an overlap, and 0 otherwise. Using this notion of relevance, precision is measured as follows.

$$\text{P} = \frac{1}{|Rank|} \sum_{time \in \varepsilon_{time}} \sum_{time_{estimated} \in Rank} \text{Rel}(time, time_{estimated}) \tag{2.36}$$

Using the same notion of relevance, the authors also measured recall, mean average precision, and normalized discounted cumulative gain.

The methods used by the authors can be divided in three main groups: based on the local time (LT), based on the nearest neighbor (NN), and distribution-propagation (DP) based. The LT method takes into account only the information in the source document. The authors considered approaches using the publication date (pd) only, and using also the surrounding information (S). On the other hand, in the NN and DP methods, the authors considered many combinations of inter-excerpt relations, such as position (P), text (T), conceptual (N) and contextual (X). They used also a method which randomly sets the weights on the edges of the graph. To compare their results against the previous state-of-the-art, the authors used the method proposed by Jatowt et al. (2013).

The method from Jatowt et al. (2013) presented the worst performance in every granularity. Otherwise, DP-TPNX proved to be the most effective method for all the measures in all granularities. Is also worth mentioning that all the NN methods performed better than the LT methods in therms of model quality.

### 2.2.3 Word Embedding Methods for Text Classification

The popularity of neural networks on NLP applications is increasing. Still their training and test speed is still relatively slow, when using large datasets. One of the most successful proposals regarding computational efficiency was made by Joulin et al. (2016). This group of Facebook researchers proposed a method called *fasttext* which was able to achieve the same accuracy of some deep learning classifiers, while substantially reducing the training and test times. Note that this method was not proposed for document dating. The focus of this study was sentiment analysis, but the same text classification method can also be used for document dating (i.e., instead of attributing a sentiment label, attributing a date label).

The base architecture was the cbow model proposed by Mikolov et al. (2013) for generating word embeddings, but with a label instead of the middle word. The authors have also used the hierarchical softmax (i.e., a softmax approach based on binary trees), which leads to a significant decrease in the word prediction task time. In the tree, each leaf represents a possible word, and the probability on each node is equal to the probability of the path from the root to the node which, in the case of the node $n_{l+1}$, is given by:

$$\mathrm{P}(n_{l+1}) = \prod_{i=1}^{l} P(n_i) \tag{2.37}$$

In the previous equation, $n_i$ are the parent nodes and $l$ is the depth in the tree. To predict the label of each document, the authors start by representing the document as a collection of $n$-grams. The features are then embedded (i.e., $n$-gram are represented as dense vectors) and averaged to form the hidden variable. Finally, they choose the label with the highest probability, minimizing the negative log-likelihood over the classes as follows:

$$-\frac{1}{td}\sum_{k=1}^{td} c_k \log(\varphi(BAx_k)) \tag{2.38}$$

In Equation 2.38 $c_k$ is the label, $x_k$ is the bag of features of the document $k$ and $A$ and $B$ are the weight matrices. The function $\varphi$ is the softmax function.

The authors performed experiments in sentiment analysis and tag prediction. On sentiment analysis, their accuracy results were very similar to previous state-of-the-art results, and with the usage of bi-grams their results were even better. Regarding to the training times, when comparing to neural network based methods, the authors were able to gain up to at least a 15000x speedup.

### 2.2.4 SemEval'15 Task 7: Diachronic Text Evaluation

SemEval is an annual workshop of evaluations of computational semantic analysis systems, with a variable number of tasks. Popescu and Strapparava (2015), summarized the proposals presented in the Task 7 of the 2015 edition of SemEval, which focused on diachronic text evaluation. This task consisted in the automatic dating of small documents, with the accuracy measured with 3 types of granularity: Fine (2 years interval for Subtask 1, and 6 years for Subtask 2), Medium (6 years interval for Subtask 1, and 12 years for Subtask 2) and Coarse (12 years interval for Subtask 1, and 20 years for Subtask 2).

There were 4 proposals with submitted results in these two Subtasks, but not all the teams performed all Subtasks. AMBRA and IXA presented results for both Subtasks, while USAAR only submitted results for the first Subtask, and UCD only approached Subtask 2. Table 2.1 presents a summary of the results obtained by the proposals submitted in subtask 2.

AMBRA (Zampieri et al., 2015) is an approach based on a learning-to-rank framework using pairwise comparisons (Liu, 2009), that was previously proposed by Niculae et al. (2014). Their classifier is trained to learn a linear function which preserves the temporal ordering of the documents. They compare each new training example with the ones already in the dataset deciding, one by one, which is older and which is newer, constructing a dataset ordered like a timeline. To make a prediction, the score of the new example is calculated using the trained linear function, and then the predicted interval is the one minimizing the average distance to the *k*-nearest neighbours. In their experiments, the authors use k=10.

Salaberri et al. (2015) presented IXA, a proposal that takes into account four approaches in order to determine the time period of time in which the piece of news was written. The first approach consists of searching for time mentions in the piece of text. This approach is characterized by a very high precision but a very low recall, since only 10% of the training samples have a mentioned time period but in those cases 85% correspond to the correct date. The second one consists of searching for named entities in the text, and then linking them with the time period described in Wikipedia. The authors regard that not all texts are assigned a

Table 2.1: Summary of the results presented on the SemEval Subtask 2

| | AMBRA | | | IXA | | | UCD | | |
|---|---|---|---|---|---|---|---|---|---|
| | Precision | Acuracy | MAE | Precision | Acuracy | MAE | Precision | Acuracy | MAE |
| Fine | 14.3 | 60.5 | —- | 3.77 | 26.18 | —- | 46.3 | 75.92 | 14 |
| Medium | 14.3 | 76.7 | —- | 6.77 | 42.8 | —- | 47.3 | 84.66 | 19 |
| Coarse | 29.2 | 86.8 | —- | 9.87 | 62.25 | —- | 54.3 | 91.04 | 19 |

time period by this approach, because some entities can not be found on Wikipedia. The third approach uses Google NGrams[4]. The percentage of occurrences of all nouns is calculated, and then the year that corresponds to the highest percentage is associated to each noun. Afterwards, the average value of all the years is calculated. If the year associated to a noun differs in 40 or more years, the authors consider this noun to be period-specific and consequently, the time period that includes this year is assigned to the text. If there is more than one time-specific noun, the year assigned is the average of all time-specific nouns. The fourth approach consists of using relevant linguistic features to language change, in combination with machine learning. Finally, to ultimately determine the period of time in which a text was written, the authors take into account the precision given by each approach, which means that the period of time yielded by the approach with the maximum precision is the chosen one. Their results in the Subtask 2 are shown in the Table 2.1

UCD (Szymanski and Lynch, 2015), the proposal only tested in the Subtask 2, opted to train an SVM classifier for each granularity (i.e., 6, 12 or 20 years) using a set of stylistic features from the texts like character n-grams, part-of-speech tag n-gram, word n-gram and syntactic phrase-structure rule occurrences. These features were then combined with the Google Books Syntactic N-Grams database (Goldberg and Orwant, 2013). With this combination of features the authors were able to detect some of the most meaningful features like the ['d] abbreviation of *-ed* used mostly in the end of the 18th century. The authors choose to use 4000 features in their experiments in order to maximize accuracy and minimizing running time. They tested many combinations of features but the most accurate method was using all the 4 features combined with the Google Books Syntactic N-Grams database. That was the final proposal presented by the authors to the competition, with the official results (i.e., with the training and test datasets of SemEval'15) presented in the Table 2.1

USAAR, the proposal that only submitted results in Subtask 1, used a crawler to solve the problem. Based on text snippets, they were able to retrieve dated webpages that contained those snippets, and used those dates to attribute timestamps to the texts presented in this task.

In the first Subtask, USAAR's proposal was clearly the best method, with at least 95.3% accuracy. However, since this approach is based in web crawling it can not be generalizable. AMBRA and IXA both perform better than the baseline, specially in the medium and coarse granularities. In the second Subtask, IXA improved only a bit, and specially in the fine granularity they only had 4% more than the baseline. On the other side, UCD got the best results,

[4]`http://books.google.com/ngrams`

being able to get 91% accuracy on the coarse granularity.

### 2.2.5 Classification Methods with Attention Approaches

In their most recent work, Yang et al. (2016) proposed a new method called Hierarchical Attention Network, designed to capture two characteristics of the documents. The first one is that documents got an hierarchical structure, i.e., words that composes sentences and sentences that composes the full document; and the second one is that different words and sentences got different informative relevance, i.e., some words or sentences are more relevant to classify a specific document.

The authors proposed and architecture divided in 4 parts (i.e., word encoder, word attention, sentence encoder and sentence attention) with an activation layer at the end, as shown in the Figure 2.10.

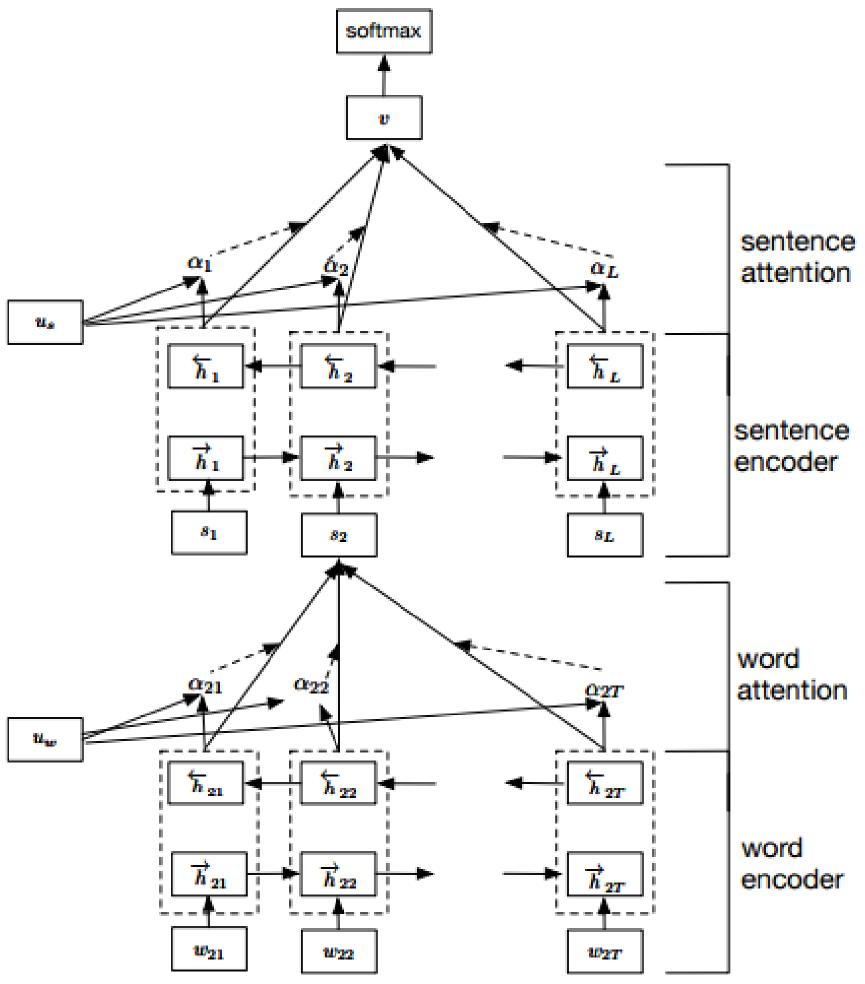


Figure 2.10: Hierarchical Attention Model by Yang et al. (2016)

The authors first embed the words to vectors using and embedding matrix. Then they apply a word encoder, which consists in a bidirectional GRU. This approach summarizes the information around each word, in both directions. The second layer is an attention layer working at word level, to extract the most important words in the context of the sentence. This importance is measured as the similarity between the word vector, and the context vector (that can be understood as the vector which answers the question ”what is the informative word?”). Afterwards, the same procedure is repeated regarding to sentences instead of words. In the end is applied a softmax activation.

With this proposal, the authors approached the task of, with basis on the textual field of a review, predict the score of atributed by the user in that same review. The authors considered 6 different datasets, namely the Yelp'13, Yelp'14, Yelp'15, IMDB, Yahoo Answers and Amazon datasets, being able to outscore all the previous results with an accuracy improvement between 3% and 6%.

### 2.2.6 Other Related Publications

Gralinski and Wierzchon (2015) proposed a new corpus for training and evaluating automatic dating systems, the RetroC dataset. Their dataset is a corpus with 50000 samples (40000 training samples, and 10000 test samples) with a 200 year span (between 1814 and 2013).

The corpus is mostly based on publications available in Polish digital libraries and only a minimal post-processing was applied. Dates presented in the text were not removed. The publication dates were collected from the metadata from the digital libraries.

This dataset was used in an online competition (Graliński et al., 2017) on the Gonito platform[5]. The aim of the competition was to determin the time period associated with each text sample.

From the many presented proposals, the authors highlighted 4 different approaches to address the automatic dating problem. The first one looks only to the year references in the text, and got 3 premises: If no year reference is found, return the median year (i.e., 1913); If only one year is found, return that year; if more than one year is referenced, return the latest year. This method yields a RMSE of 46.4 years. The second approach takes attention to the changes in the Polish orthography over the years. The authors divided the datasets into three periods (1814-1918, 1918-1936 and 1936-2013) adjusted the scores accordingly and take the median year

[5]`http://gonito.net/challenge/retroc`

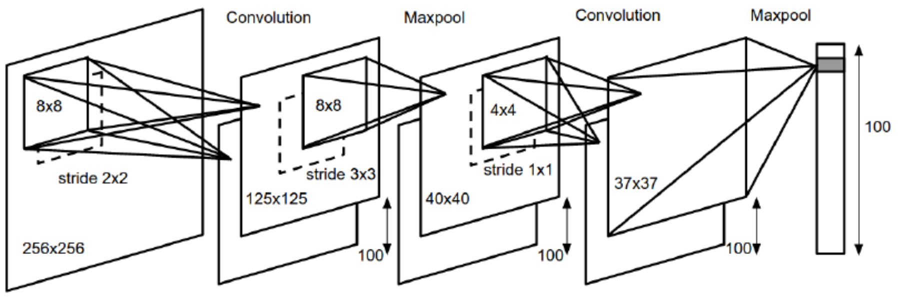


Figure 2.11: Image Model architecture by Li et al. (2015)

of the winning period. This method achieved a 86.6% precision for recognition of one of the three periods and a RMSE of 50.9 years. The third method considered was a regression approach with the output taking continuous values. Using the Vowpal Wabbit open-source learning system (Langford et al., 2009) and lower-case tokens and/or character pentagrams as features, they managed to achieve a RMSE of 33.5 years in the test dataset.

Even combining the 3 approaches, the results didn't had a significant improvement (together they had a RMSE of 33.1 years)

The best performed approach was achieved combining the third approach (the regression model using the Vowpal Wabbit learning system) with a basic neural network with 6 neurons. The final score of this method was a RMSE of 24.8 years.

Another approach, based in a pure neural network, was presented and managed to achieve a similar result (RMSE of 24.8 years) but accordingly to the authors, the proposal was mot fully transparent.

In a slightly different approach of the ones presented before, Li et al. (2015) presented a method which combined two methodologies. The author's goal is to estimate the publication date of scanned documents, analyzing not only the textual information present in the text (collected using an Optical Character Recognition (OCR) tool), but also the visual features of the document such as calligraphy or images. The authors developed three types of models: an Image model with only image input, a Text model with only a text input, and a combined model, with both approaches.

The Image model receives as input a $256 \times 256$ patch, and is composed by two convolutional layers interspersed by maxpool layers as shown in Figure 2.11.

The text model receives as input the result of the OCR converted into bag-of-words and then each document is represented as a 50001-dimensional vector (composed by the counts of the 50000 most common words, and the last feature being the number of out-of-vocabulary words). This input is connected to an embedding layer with output dimensionality 100, which is afterwards connected to an hidden layer with a rectified linear activation function. Each one of this individual approaches ends with a prediction layer.

The combined model is basically a concatenation of the previous two models.

With this model they considered two different tasks. A classification task, where they split the documents in 4 different classes (each one corresponding to one century); and a regression task where they directly estimate the publication year of the document.

In the classification task the authors managed to achieve at least 80% accuracy in every century using the combined method. Note that in this task, in some of the classes, the combined model was not the most effective one.

In the regression task the combined model achieved best results than the simplest ones in all of the classes and they achieved a MAE of 23 years or lower in every class.

## 2.3 Overview

In the present chapter, I presented the fundamental concepts to understand the work that has been made in the context of my M.Sc. thesis. Furthermore, I also reviewed some of the more relevant studies made in the area, with special attention on the ones related to the datasets used in this work (e.g., the submitted proposal for the SemEval 2015 Task 7 (Popescu and Strapparava, 2015), the Jatowt et al. (2013) graph-based approach or the language modeling method from Kumar et al. (2011)).

# 3 Experimental Method

This chapter presents the considered automatic dating approach. Section 3.1 outlines the proposed model, with a detailed explanation of each aspect of the proposed methodology. The training parameters considered are also detailed in Section 3.2. Section 3.3 presents a summary of the chapter.

## 3.1 Proposed Model

Our model takes inspiration in some of the most recent works made in the area of natural language processing, such as Joulin et al. (2016), Yang et al. (2016) and Duarte et al. (2017).

We propose a neural network architecture that aims to determine a date label associated to each textual document, and that combines two approaches previously reviewed and used in the field of sentiment analysis, namely the FastText method proposed by Joulin et al. (2016) and .

Our task is formulated as multi-class classification problem, discretizing the timeline into contiguous atomic time spans (i.e., the classes correspond to these discrete chronons, each with a duration of one year) and attempting to determin which label is more suitable to each textual sample.

Figure 3.1 represents the architecture of our proposal, which is detailed next. For an in-depth introduction to deep neural networks for natural language processing, complementing the descriptions from the following sections, the reader can refer to the tutorial by Goldberg (2016).

Since all textual documents have an hierarchical structure (i.e., words form the different sentences, and the sentences compose the global document), our model starts by creating a representation for the several sentences based in their words, and then aggregates those into a global representation for the entire document.

This two-level hierarchical approach is illustrated in Figure 3.1, with the word-level part of the model (i.e., the part that generates a representation from a given sentence, based on the

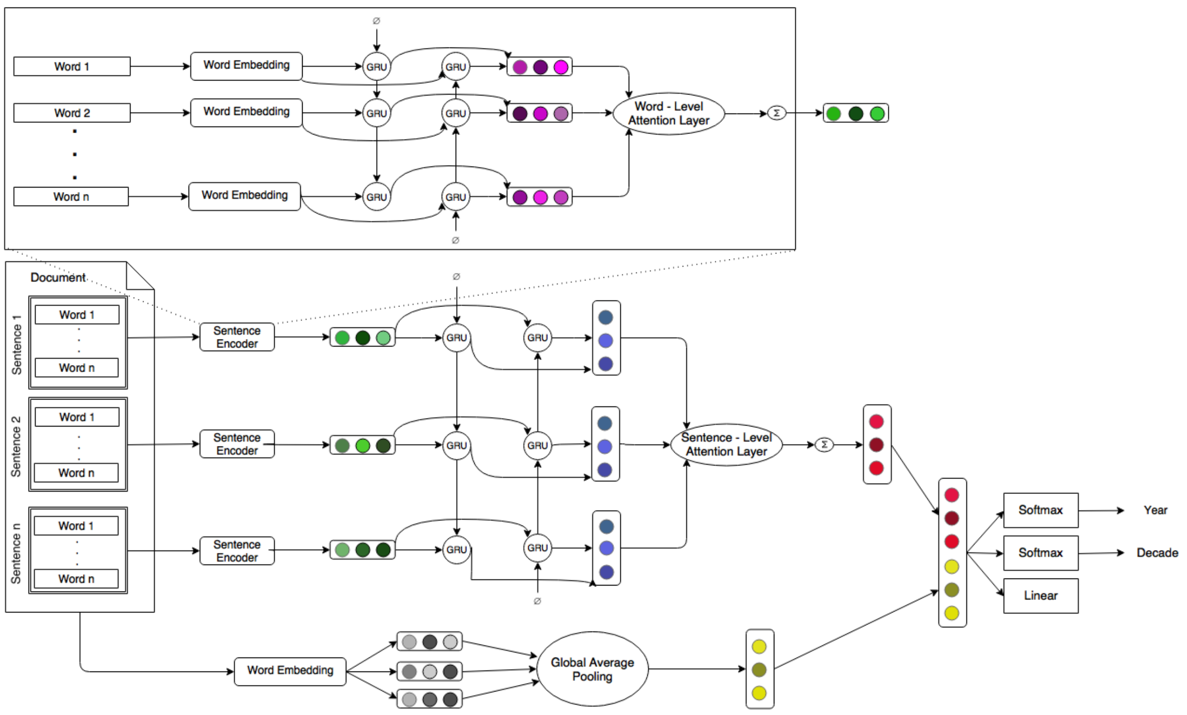


Figure 3.1: Neural Network Architecture Schema

composing words) shown in the box at the top. A recurrent neural network node known as a Gated Recurrent Unit (GRU) is used at both levels to build the representations (shown in purple in the word-level, and in blue in the sentence-level), and we specifically considered bi-directional GRUs (Cho et al., 2014) combined with neural attention mechanisms (Yang et al., 2016).

In the first level of our model, the GRUs leverage the word embeddings as input, while in the second level they use as input the sentence representations (shown in green) generated in the previous level. In the case of tests with English data, we specifically leveraged 300-dimensional word embeddings, pre-trained on the Common Crawl and considering cased words, made available in the context of the GloVe project (Pennington et al., 2014). The experiments with texts in Polish language were similarly made with the use of pre-trained word embeddings made available by Bojanowski et al. (2016). The embeddings layer is are initialized with basis on these pre-trained values, and then adjusted during model training.

GRUs model sequential data by having a recurrent hidden state whose activation at each time step is dependent on that of the previous time step. A GRU computes the next hidden state $h_t$ given a previous hidden state $h_{t-1}$ and the current input $x_t$ using two gates (i.e., a reset gate $r_t$ and an update gate $z_t$), that control how the information is updated, as shown in Equation 3.1. The update gate (Equation 3.2) determines how much past information is kept and how much new information is added, while the reset gate (Equation 3.4) is responsible for how much the past state contributes to the candidate state. In Equations 3.1 to 3.4, $\tilde{h}_t$ stands for the current new state, $W$ is the parameter matrix for the actual state, $U$ is the parameter matrix for the previous state, and $b$ a bias vector.

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{3.1}$$

$$z_t = \sigma\big(W_z x_t + U_z h_{t-1} + b_z\big) \tag{3.2}$$

$$\tilde{h}_t = \tanh\big(W_h x_t + r_t \odot (U_h h_{t-1} + b_h)\big) \tag{3.3}$$

$$r_t = \sigma\big(W_r x_t + U_r h_{t-1} + b_r\big) \tag{3.4}$$

Bi-directional GRUs perceive the context of each input in a sequence by outlining the information from both directions. Concatenating the output of processing a sequence forward $\overrightarrow{h}_{it}$ and backwards $\overleftarrow{h}_{it}$ grants a summary of the information around each position, $h_{it} = [\overrightarrow{h}_{it}, \overleftarrow{h}_{it}]$.

Since words and sentences can be differently informative in specific contexts (i.e., different relevance in the context of the task), the model also includes two levels of attention mechanisms (i.e., one at the word level and one at the sentence level), that allow the model to pay more attention to individual words/sentences when constructing representations, regarding its relevance to the task. For instance, in the case of the word-level part of the network, the outputs $h_{it}$ of the bi-directional GRU encoder are fed to a feed-forward node (Equation 3.5), resulting in vectors $u_{it}$ representing words in the input. A normalized importance $\alpha_{it}$ (i.e., the attention weights, represented in purple in Figure 3.1) is calculated as shown in Equation 3.6, using a context vector $u_w$ that is randomly initialized. The importance weights in $\alpha_{it}$ are then summed over the whole sequence, as shown in Equation 3.7.

$$u_{it} = \tanh\left(W_w h_{it} + b_w\right) \tag{3.5}$$

$$\alpha_{it} = \frac{\exp\big(u_{it}^T u_w\big)}{\sum_t \exp\big(u_{it}^T u_w\big)} \tag{3.6}$$

$$s_i = \sum_t \alpha_{it} h_{it} \tag{3.7}$$

The vector $s_i$ from Equation 3.7 is finally taken as the representation of the input (represented in green in the Figure 3.1). The part of the network that processes the sequence of sentences similarly makes use of bi-directional GRUs with an attention mechanism, taking as input the representations produced for each sentence, as shown in Figure 3.1.

The representation that is produced as the output of the sentence-level attention mechanism (represented in red in Figure 3.1), which encompasses the entire output, is also concatenated with an alternative representation built through a simpler mechanism which, taking inspiration on the good results reported by Joulin et al. (2016), computes the average of the embeddings for all words in the input sentences (represented in yellow in Figure 3.1). The word embeddings layer is shared by the hierarchical attention and the averaging mechanisms, and thus while one part of the model uses multiple parameters to compute representations for the inputs, the other part of the model acts as a shortcut that can more directly propagate errors back into the embeddings, so that they can be updated.

The output layer of the model considered 3 different nodes, each one with a different output value. The first one consists in a softmax node associated to a categorical cross-entropy loss function, producing an output label corresponding to the most suitable year to each test sample. This can be considered as the main output node since all the metrics presented are calculated considering the output of this node. The main intention of the other two output nodes was to improve the global performance of the model. The second node is also a softmax and also associated with a categorical cross-entropy loss function, but in this case producing an output corresponding to the predicted decade of each test sample. The final output layer is a linear node which considers a loss function corresponding to the mean squared error between the predicted year and the ground-truth year.

## 3.2 Training Parameters

The entire model is trained end-to-end from a set of documents assigned to gold-standard years, leveraging the back-propagation algorithm (Rumelhart et al., 1988) in conjunction with the Adam optimization method (Kingma and Ba, 2015). The adjustments of the parameters of the model was made using the Semeval dataset.

In the combined loss function, the three output nodes have weights of 1.0, 1.0 and 0.25, respectively. To set this weight values, many different combinations of values were tested, and the final weights correspond to the best combination for the Semeval dataset.

The word embedding layer considered a dimensionality of 300. Were performed tests with embedding layers of dimensionality 50, 100, 200 and 300 (corresponding to the available GloVe embeddings), and the test considering a dimensionality of 300 was the best performer. The output of the GRUs had a dimensionality of 150 and so, the combination of forward and backward GRU produces representations for words/sentences with 300 dimensions. Model training was made in batches of 32 instances, using the default parameters for the Adam optimization algorithm, and considered a stopping criteria based on the combined training loss, finishing when the difference between epochs was less than 0.0000001 with a patience value of 2.

The implementation of the model relied mostly on the keras[1] deep learning library, although the scikit-learn[2] machine learning package was also used for specific operations (e.g., for computing the evaluation metrics and for creating the cross-validation data splits that were considered in the experimental evaluation).

## 3.3 Summary

This chapter presents a detailed description of the proposed model for the automatic dating task. The different layers of the proposed architecture, illustrated in Figure 3.1, were detailed in Section 3.1. Afterwards, in Section 3.2, we overviewed the training parameters considered for the presented approach.

[1]`http://keras.io`
[2]`http://scikit-learn.org`

# 4 Experimental Results

The experimental results can be summarized by various statistics, such as the predictive accuracy (i.e., the percentage of documents assigned to the correct year), the Root Mean Square Error (RMSE) between the estimated and the ground-truth dates, or the Mean Absolute Error (MAE). The formulas corresponding to these last two metrics are as follows.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}} \tag{4.1}$$

$$MAE = \frac{\sum_{i=1}^{n}|\hat{y}_i - y_i|}{n} \tag{4.2}$$

In Equations 4.1 and 4.2, $\hat{y}_i$ corresponds to a predicted date, $y_i$ corresponds to a true date, and $n$ is the number of predictions. Using multiple error metrics can have advantages, given that individual measures condense a large number of data into a single value, thus only providing one projection of the model errors that emphasizes a certain aspect of model performance. For instance, Willmott and Matsuura (2005) proved that the RMSE is not equivalent to the MAE, and that one cannot easily derive the MAE value from the RMSE (and vice versa). While the MAE gives the same weight to all errors, the RMSE penalizes variance, as it gives errors with larger absolute values more weight than errors with smaller absolute values. When both metrics are calculated, the RMSE is by definition never smaller than the MAE. Chai and Draxler (2014) argued that the MAE is suitable to describe uniformly distributed errors, but because model errors are likely to have a normal distribution rather than a uniform distribution, the RMSE is often a better metric to present than the MAE. Multiple metrics can provide a better picture of error distribution and thus, in our study, we present results in terms of the MAE and RMSE metrics.

This chapter is organized as follows. In Section 4.1 is presented a description for each one of the datasets used in the validation of the model. Section 4.2 overviews the previous results obtained with the datasets described in the previous Section. Section 4.3 presents the results obtained with the proposed model. Section 4.4 analyses the advantages of the attention mechanism. Finally, Section 4.5 presents an overview of the chapter.

## 4.1 Description of the Datasets

To test our model we used four different datasets. Some of them where already used in some studies (e.g., SemEval 2015 dataset) while others were collected by us (e.g., Gutenberg dataset). They all have different distributions and characteristics in order to fully test the capability of our methodology.

Table 4.1 presents a summary of the characteristics of the datasets used in our experiments. A more detailed description of the collections is presented in the following subsections.

### 4.1.1 SemEval'15 Dataset

The first dataset taken into account in our experiments is the one referent to the Subtask 2 of the SemEval 2015 task 7. It is composed by 4168 training examples and 1041 test samples, with their distributions shown in Figure 4.1. Each of the samples is associated to 3 types of intervals: Fine (6 years), Medium, (12 years) and Coarse (20 years).

The dataset is composed by parts of text from various news, present in journals available in electronic format, specially NPA[1], SPR[2] and BDY[3]. Unlike in Subtask 1, where there are historical events or named entities clearly mentioned, in Subtask 2 such information is missing, but there is enough information to assign a time interval to the excerpt at least for a human being.

Table 4.1: Statistical characterization for the datasets used in our experiments.

| | SemEval | RetroC | Gutenberg | Wikipedia |
|---|---|---|---|---|
| Number of documents (train/test) | 4168/1041 | 40000/9910 | 1000 | 195 |
| Number of classes (years) | 311 | 200 | 54 | 108 |
| Vocabulary size | 26041 | 1738577 | 88650 | 55744 |
| Number of words p/doc (avg.) | 66.331 | 513.729 | 4711.837 | 346.441 |
| Number of sentences p/doc (avg.) | 2.478 | 39.974 | 336.101 | 9.138 |

[1]`http://newspaper.archive.com`

[2]`http://archive.spectator.co.uk/`

[3]`http://www.bodley.ox.ac.uk/ilej/`

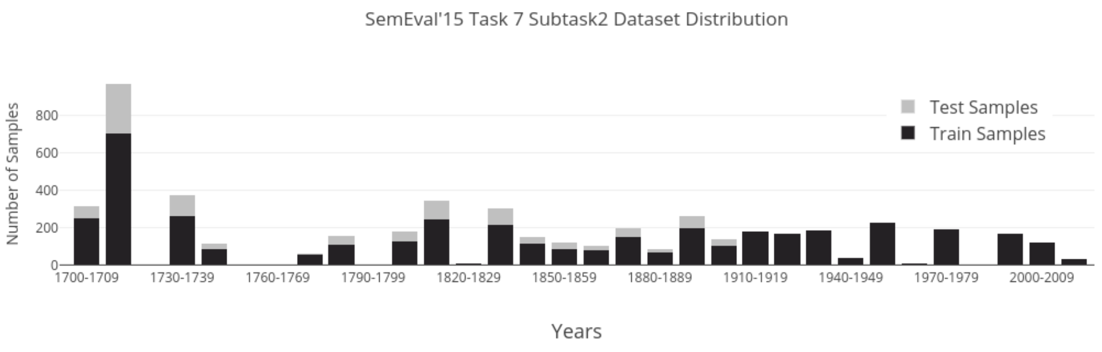


Figure 4.1: Distributions of the SemEval'15 dataset

We processed the dataset, representing each interval by his middle point. This means that, if our method's prediction is equal to the correct class (i.e., interval middle point), our prediction is correct; otherwise is wrong. Due to that fact, the granularity is irrelevant. In any case, we are comparing our results to the ones obtained in the coarse granularity (i.e., higher results).

In the SemEval'15, there were 3 presented proposals to this Subtask 2, as described in the section 2.1.

### 4.1.2 Gutenberg Dataset

The Gutenberg Project is a digital free library with a very large collection of eBooks, which relies on a community effort to digitize and share out-of-copyright works.

This dataset, collected by us from the Australian Gutenberg project website[4], is composed by short stories with publication dates between 1892 and 1945. It is composed by 1000 prose texts in plain text format, and distributed as shown in Figure 4.2.

This corpus is completely different from the previous one, with much larger sentences, and with a way more larger number of sentences per document as shown in the Table 4.1

This corpus is not split into training and test sets, so in this case we opt to use cross-validation with 10 folds to collect the results.

[4]`http://gutenberg.net.au/`

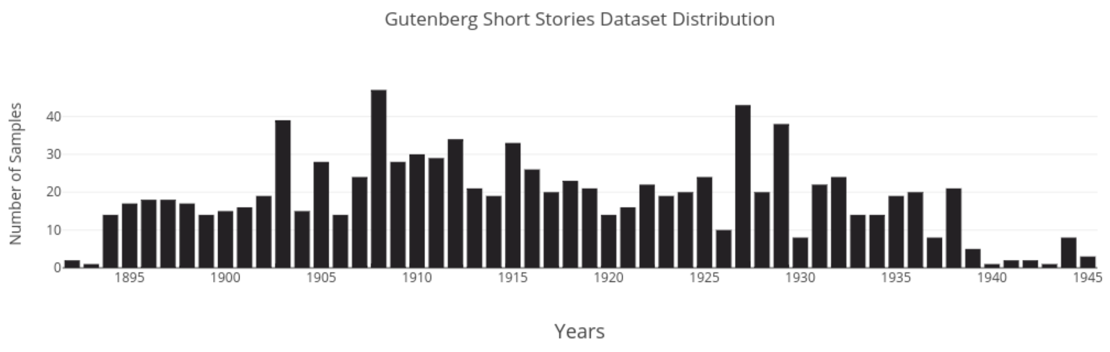


Figure 4.2: Distributions of the Gutenberg dataset

### 4.1.3 Wikipedia Dataset

Wikipedia pages are also valuable for evaluation purposes. This free online encyclopedia is based on collaborative writing by its users, and where each event described got its focus time (i.e., time period associated to the event described) well defined. This dataset was based in the one that Adam Jatowt used in their previous work (Jatowt et al., 2013). It is composed by 195 historical events from 4 different countries with the following distribution: 48 events from UK and France, 49 events from Germany and 50 events Israel. The initial dataset from Adam Jatowt was composed by 250 samples (50 from each country), but since the it's work, some of the wikipedia pages were removed and/or merged. We were also unable to collect the 50 events from Japan. All the events selected occurred between 1900 and 2013.

In order to simplify the approach, the ground truth is the middle point between the start date and end date of each event (i.e., with the same procedure used in the SemEval dataset).

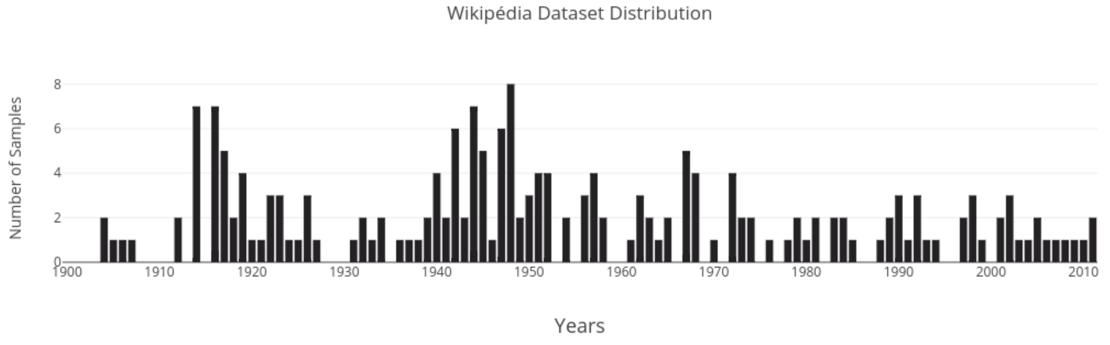


Figure 4.3: Distributions of the Wikipedia dataset

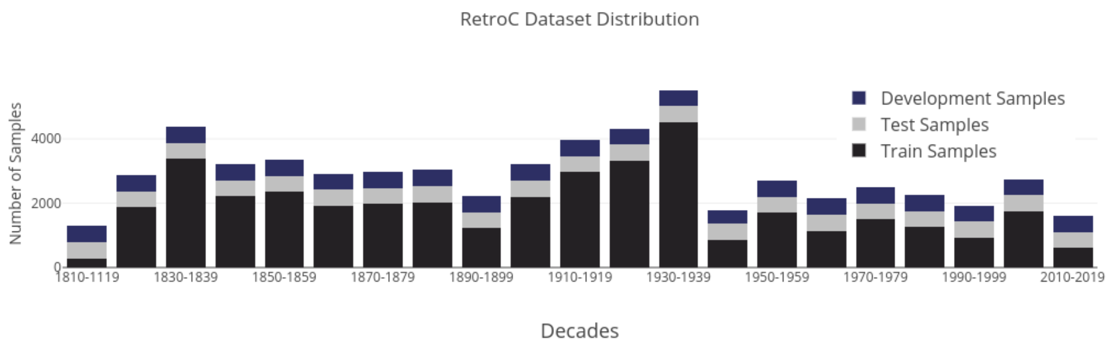


Figure 4.4: Distributions of the Retroc dataset

The distribution of the full dataset can be seen in the Figure 4.3.

As well as in the previous dataset, this one is also not divided in training/test splits, so we use again cross-validation, but in this time, due to the small amount of samples, we use only 5 folds.

### 4.1.4 Retroc Dataset

*RetroC* (Gralinski and Wierzchon, 2015) is a Polish-language diachronic corpus, mostly based on publications available in Polish digital Libraries, and with publication times between 1814 and 2013.

The publication dates associated to each sample were collected from the metadata of the digital libraries without any manual verification, so there is no guarantee that all of the dates given in the dataset are correct.

As we can see in Figure 4.4 the test set is balanced with respect to publication year, with 50 samples for each class. On the other hand, the training set don't have any particular pattern.

## 4.2 Previous Results

Some of these datasets were previously used in other studies, and despite punctual differences in some cases (i.e., slightly different datasets, different number of classes considered or different years associated to the text), it is important to cross our results to the previous studies made on these types of datasets. Table 4.2 shows a summary of the best results for the 4 considered datasets.

Starting with the dataset of SemEval 2015. Three different proposals were submitted on the Subtask 2 (that used the dataset presented before, and used in our experiments). These proposals were already detailed in Chapter 2. The UCD (Szymanski and Lynch, 2015) approach was the better one on the task, achieving a precision of 54.2% on the coarse granularity, and a MAE of 19 years. Two other proposals were submitted on this task, with AMBRA (Zampieri et al., 2015) getting 29.2% precision and a MAE of 31.74 years, while the IXA (Salaberri et al., 2015) approach was the least successful one with a precision of 9.8

There are few works done using texts collected from the Gutenberg Project website, but Kumar et al. (2011) used a similar dataset in their experiment (despite the use of other features), as explained in detail in the related work section. With their model, the authors managed to achieve a MAE of 34 years. The dataset used in their work is not freely available, so the samples used in our experiments are not the same. In any case the aim of the study and the characteristics of the samples are very similar.

The wikipedia dataset used in our tests is a part of a dataset previously used by Jatowt et al. (2013). In their work (as explained in detail in the related work section), the authors do not train the model with the wikipedia pages, so their training procedure is different that the one considered in my proposal. They considered two main modules. In the basic one they managed to achieve a MAE of 18.3 years, while in the extended model which incorporates an auxiliar calculation using the dates in the text, they had a MAE of 2.83 years

RetroC dataset was used in an online challenge[5] (Graliński et al., 2017) and, as detailed in the related work section, many proposals were submitted. The best one was the combination of the Vowpal Wabbit learning system (Langford et al., 2009) with a complementary neural network, managing to achieve a RMSE of 24.8 years in the test set. With the development set, their model produced a RMSE of 17.2 years. Another interesting approach, using a pure neural network, had a very similar result of 24.9 year of RMSE in the test set, but accordingly

Table 4.2: Summary of Previous Results

| | Best Result | | | | Interesting Alternative | | | |
|---|---|---|---|---|---|---|---|---|
| | Reference | MAE | RMSE | Accuracy | Reference | MAE | RMSE | Accuracy |
| SemEval | Szymanski and Lynch (2015) | 19 | — | 0.542 | Zampieri et al. (2015) | 31.74 | — | 0.292 |
| RetroC | VW + NN by Graliński et al. (2017) | — | 17.2 | — | VW only by Graliński et al. (2017) | — | 22.0 | — |
| Gutenberg | Kumar et al. (2011) | 34 | — | — | — | — | — | — |
| Wikipedia | Extended Proposal by Jatowt et al. (2013) | 2.83 | — | — | Basic Proposal by Jatowt et al. (2013) | 18.3 | — | — |

[5] `http://gonito.net/challenge/retroc`

to Graliński et al. the experiment was not fully transparent so is not detailed in their article.

## 4.3 Model Results

In our experiments we use the percentile 75 values for the number of sentences per text and words per sentence values. With this values we guarantee a small lose of information in the 25% bigger samples, while avoiding the waste of information in the smaller ones. Table 4.3 sum up the percentile 75 values for the 4 datasets used in our experiments.

We performed 3 different tests, in each dataset, with our model. The first two columns of the table present the two simpler approaches, considering each one of them one of the two branches of our model. The first one considers only the inputs received (i.e., the words that composes the sentences of the texts) and makes an average of the embeddings associated to the inputs. This can be considered the simplest branch of our model. The second one presents the performance of the hierarchical attention mechanism. In this branch the input is also the embedding layer, but then are applied the Bidirectional GRUs and attention layers at the word and sentence levels. The Full Model approach is the one that concatenates the two previously explained branches.

Note that in the Wikipedia results, the accuracy result presented measures the number of correct predictions considering their real time interval, while the value in parenthesis only considers the middle point accuracy.

In the SemEval dataset we managed to achieve an accuracy of 35.93%. This result would rank our solution in the second place only outscored by the UCD (Szymanski and Lynch, 2015) proposal which had 54.2%. Regarding the MAE, we also have a worst performance with 36.58 years while UCD lowered that number to 19 years. This improvement in the results of the UCD proposal can possibly be explained by the usage of the Google Books N-Grams database. Comparing with the other proposals presented in the competition, our model outscored the IXA (Salaberri et al., 2015) approach (the worst presented proposal) by a large margin, since they

Table 4.3: Percentile 75 for the different datasets

| | SemEval | RetroC | Gutenberg | Wikipedia |
|---|---|---|---|---|
| Number of Sentences | 3.0 | 47.0 | 449.0 | 10.0 |
| Words per Sentences | 37.0 | 18.0 | 19.0 | 47.0 |

Table 4.4: Summary of the Model Results

| | Avg Embeddings | | | Hierarchical Attention | | | Full Model | | |
|---|---|---|---|---|---|---|---|---|---|
| | Accuracy | MAE | RMSE | Accuracy | MAE | RMSE | Accuracy | MAE | RMSE |
| SemEval | 35.06% | 35.49 | 64.35 | 7.20% | 73.40 | 98.15 | 35.93% | 36.58 | 63.90 |
| RetroC | 18.03% | 15.61 | 30.16 | 0.05% | 52.50 | 61.54 | 17.17% | 16.47 | 31.86 |
| Gutenberg | 44.64% | 4.35 | 8.53 | 5.72% | 11.16 | 14.21 | 40.66% | 5.27 | 9.57 |
| Wikipedia | 16.61% (15.27%) | 10.06 | 28.7 | 12.22% (9.33%) | 18.09 | 26.83 | 14.72% (14.28%) | 16.29 | 24.52 |

only managed to achieve an accuracy of 9.8%. The authors did not presented any MAE or RMSE values, so we could not compare those parameters. Finally, regarding the AMBRA (Zampieri et al., 2015) proposal, we managed to have a better accuracy (35.93% from our model, 29.2% from AMBRA), but we had a worst MAE value (36.58 years in our proposal, 31.74 years with AMBRA's method).

The results on the RetroC dataset were not impressive. We were not able to approach the best performed methodology (i.e., the vowpall wabbit regression combined with the neural network). In any case, with a RMSE of 31.86 years in the full model, our approach performed way better than their all of the base models presented, namely the years reference method (with a RMSE of 45.9 years), the polish orthography method (with a RMSE of 46.6 years) and the external sources method (with a RMSE of 57.4 years).

In the Gutenberg dataset, ours results were way better than the ones presented by Kumar et al. (2011). We were able to achieve a MAE of 5.27 years in the full model, which is way lower than the 34 years presented in Kumar's paper. In any case, we should regard that we used a different dataset, since the dataset used in their work was not publicly available.

Our results in the Wikipedia dataset were somehow positive. Taking into account the results presented by Jatowt et al. (2013) with a very close dataset, we managed to outscore their basic method (i.e., approach without the document-time associations based on dates extracted from documents) with an average error of 16.29 against the 18.3 achieved by Jatowt et al. (2013). Unfortunately we were not able to outscore the extended version, which already leveraged the dates extracted from the texts, and produced an average error of 2.83 years. This can be easily explained since we, as Jatowt et al. in their first approach, do not give special attention to dates present in the text.

Despite having interesting results in some of the used datasets, in most of the cases, we were not able to surpass the state-of-the-art results.

Analysing the 3 different tests made with each dataset and presented in Table 4.4 we can see that, in some cases, the hierarchical attention mechanism was able to improve the results obtained by the average of the word embeddings. In the SemEval dataset, the full model (i.e., combining the attention model with the average of the embeddings) had a better accuracy and better RMSE than the approach considering only the average of the embeddings. In the wikipedia dataset, the attention model had a lower RMSE than the approach considering only the average of the word embeddings. The full model had an even lower value of RMSE. In the other two datasets, the hierarchical attention model had much worse results when compared to the average of the embeddings.

The reasons behind these results may have different sources. For example, the datasets used could not be the most appropriate to test the model (Duarte et al. (2017) and Yang et al. (2016), in their experiments with hierarchical attention mechanisms, used datasets with different characteristics). Another reason can possibly be the fact that we rely solely on clues present in the texts, while other approaches used some other features in the training process. In any case, it would require additional tests to confirm or refute this possible causes.

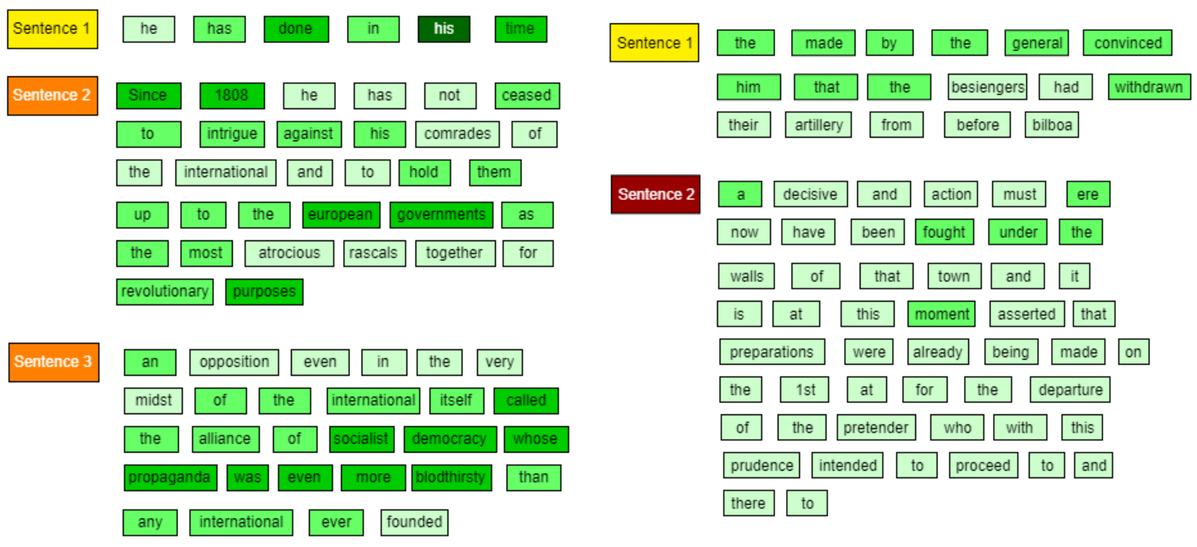
## 4.4 Analysis of the Attention Mechanism

The attention mechanism, as implemented in our model, can also offer model interpretability, allowing us to see which parts are more or less relevant to the classification task. In Figure 4.5 we can see two examples of attention weights assigned to the sentences and words in two different textual documents from the SemEval collection.

In the first document, we can see that Sentence 1 is the one with lower attention value. In the sentence with highest weight (i.e., Sentence 2) we can see that there is a year reference (i.e., 1808). The words *Since* and *1808* are stated as two of the most relevant words in the sentence. These values shows that the model gives more relevance to dates instead of more common words.

In the second example, we can see that Sentence 2 is much more relevant than Sentence 1. In the presented case, Sentence 2 got an attention value of 0.55, while Sentence 1 only has an attention value of 0.12. One of the possible reasons for that is the presence of the term *ere*, which is archaic word for *before*, mostly used in the XIX century. In fact, the ground truth value for this document is 1836, which means that this particular term can have an high impact in the prediction results for this example.

This attention mechanism, despite not leading to an increased performance in all of the presented datasets, gives us the opportunity to interpret the attention weights attributed to


Sentence 1
he has done in his time
Sentence 2
Since 1808 he has not ceased to intrigue against his comrades of the international and to hold them up to the european governments as the most atrocious rascals together for revolutionary purposes
Sentence 3
an opposition even in the very midst of the international itself called the alliance of socialist democracy whose propaganda was even more blodthirsty than any international ever founded
Sentence 1
the made by the general convinced him that the besiengers had withdrawn their artillery from before bilboa
Sentence 2
a decisive and action must ere now have been fought under the walls of that town and it is at this moment asserted that preparations were already being made on the 1st at for the departure of the pretender who with this prudence intended to proceed to and there to

(a) (b)

Figure 4.5: Examples of the Distributions of the Attention Weights for two Documents

different parts of the texts and infer which words and/or sentences have more impact on the prediction.

## 4.5 Overview

In this chapter, I presented the experimental results obtained with the proposed methodology. Different tests were made, using very different datasets (detailed in Section 4.1), in order to attest the robustness of the model. Despite the fact that, in some cases, the results obtained were above the ones presented in previous studies, the introduction of the attention mechanism allows us to interpret the weighs attributed to each component of the texts. Additionally, in some cases, we were able to surpass some of the previous results obtained in the datasets (e.g., SemEval 2015 dataset or the Wikipedia dataset).

# 5 Conclusions and Future Work

Automatic dating is an area with growing interest in the NLP community, and with this raising popularity, the number of studies and approaches presented also increases. This novel automatic dating method combines two approaches (never explored in the temporal resolution area), trying to perceive and leverage the context of the words and sentences in textual samples.

Among the reviewed previous work on the area, compiled in Section 2.2, numerous approaches were considered like language modeling, graph-based methods, maximum likelihood calculations, but none of them applied any hierarchical attention mechanism to perceive the context of each word/sentence. This novel approach can have multiple applications and be used in very differenciated type of datasets (i.e., is not exclusive of dating tasks, and can be used in any classification task with textual inputs).

The consistent results presented in different datasets, with very different characteristics, shows the robustness of this model and it's capability to adapt to different types of samples with small adaptations like the number of sentences per document or the sentence dimensions. In any case, the results were not perfect and, in most of the cases, the hierarchical attention mechanism didn't managed to improve the results achieved by the simplest branch of the model (i.e., the average of the embeddings).

Even in this case, we managed to achieve some interesting results, namely in the Wikipedia dataset (when compared to the basic proposal by Jatowt et al. (2013)). Another very positive result was achieved in the Gutenberg dataset, were our model achieved a much lower MAE when compared to the Kumar et al. (2011) approach. Anyway, in this case, we cannot ensure the similitude of the dataset, so our results may not be directly comparable.

## 5.1 Overview on the Contributions

The main contributions of my M.Sc. thesis are as follows:

- **A new method for automatic dating of textual documents**: The proposed methodology introduces a new model in the automatic dating tasks. As far as we know, the attention mechanisms were never taken into account in automatic dating tasks and it was proved with our new model that, in some cases, it can have a positive impact.

  Even without having outstanding results, our model managed to approach some previous state-of-the-art results which, for a new unexplored approach, is a very solid performance.

  We believe that, with some future adjustments (some of them stated in the future work section), we will be able to pair or even surpass some of the previous best result in the area.

- **Experiments with different datasets**: The datasets used to validate the proposed model were very different from each other. One of them composed by few sentences and so, much less information to be used in the train (SemEval); other composed by short stories, which means that the texts has a narrative associated, with much more sentences, but most of them composed only by fictional information (Gutenberg short stories); other one composed by real life facts, with historical events with a clear time span associated which are, at least for an human being, easy to date (Wikipedia dataset); and other using a non-English language (RetroC dataset).

  The usage of multiple datasets is not, by it self, a positive fact, but with this diversity in our datasets, we prove that our model is resilient enough to achieve consistent results in datasets with different characteristics.

- **Gutenberg Project dataset**: This new dataset was hand collected by me from the website of the Australian Gutenberg Project. It is composed by 1000 samples (as detailed in the Section 4.1.2).

  This is a dataset oriented to NLP studies, specially dating tasks, since each sample got a label (present in the first line of each file), corresponding to the publication year of each ebook.

  With the free distribution of this dataset, other works can cross their results with the ones from our model, and try to improve them using the same corpus.

## 5.2 Future Work

Despite the interesting results there are also many ideas for future work, since different options can be considered for improving the neural network architecture. For instance, to better handle out-of-vocabulary words and issues related to changes in word spelling over time, we could consider alternative mechanisms for exploring context in the generation of the word embeddings, or replacing/enriching the embeddings with mechanisms that generate representations from individual characters or character $n$-grams (Bojanowski et al., 2017).

Our neural architecture leverages GRUs to encode sequences of words, but other types of recurrent nodes have also recently been proposed. For instance, the Minimal Gated Unit approach (Zhou et al., 2016; Heck and Salem, 2017) relies on a simplified model with just a single gate. Having less parameters to train can contribute to improving the model effectiveness. In contrast, Multi-Function Recurrent Units (Mu-FuRUs) adopt an elaborate gating mechanism that allows for additional differentiable functions as composition operations, leading to models that can better capture the nuances involved in encoding word sequences (Weissenborn and Rocktäschel, 2016). Other alternatives include Long Short-Term Memory (LSTM) networks with coupled gates (Greff et al., 2016), Structurally Constrained Recurrent Networks Mikolov et al. (2014), IRNNs (Le et al., 2015), and many other LSTM or GRU variants (Greff et al., 2016; Jozefowicz et al., 2015).

Another idea yet relates to the use of sparse modeling methods as an approach to improve the predictions at the output nodes, by using sparsemax instead of the softmax and sigmoid activation at the model outputs (Martins and Astudillo, 2016). Sparse modeling methods could also be used as an approach to improve the interpretability of the attention mechanisms (Niculae and Blondel, 2017) (i.e., standard attention tends to produce dense outputs, in the sense that all elements in the input always make at least a small contribution to the decision, while sparse alternatives can better encourage parsimony and interpretability).

Regarding the datasets used in the study, despite the variety of datasets already used, there are many other options to explore. For example, Bamman and Crane (2011) published a dataset composed by over 10000 OCR'd Latin language texts, each one with an associated publication date. This dataset and pre-trained Latin word embeddings are freely available online[1].

[1] `http://www.cs.cmu.edu/~dbamman/latin.html`

# Bibliography

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations.*

Bamman, D. and Crane, G. (2011). Measuring historical word sense variation. In *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries.*

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016). Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606.*

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5.

Campos, R., Dias, G., Jorge, A. M., and Jatowt, A. (2014). Survey of temporal information retrieval and related applications. *ACM Computing Surveys*, 47(2).

Chai, T. and Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)? - Arguments against avoiding RMSE in the literature. *Geoscientific Model Development*, 7(3).

Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of the Workshop on Synthax, Semantics and Structure in Statistical Translation.*

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555.*

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research.*

Derczynski, L. R. (2017). *Automatically Ordering Events and Times in Text.* Springer International Publishing.

Duarte, F., Martins, B., Sousa Pinto, C., and Silva, M. (2017). A deep learning method for icd-10 coding of free-text death certificates. In *EPIA 2017:Progress in Artificial Intelligence*.

Goldberg, Y. (2016). A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57.

Goldberg, Y. and Orwant, J. (2013). A dataset of syntactic-ngrams over time from a very large corpus of english books. In *Conference on Lexical and Computational Semantics*.

Graliński, F., Jaworski, R., Borchmann, L., and Wierzchoń, P. (2017). The retroc challenge: How to guess the publication year of a text? In *International Conference on Digital Access to Textual Cultural Heritage*.

Gralinski, F. and Wierzchon, P. (2015). RetroC – A Corpus for Evaluating Temporal Classifiers. In *Proceedings of Language & Technology Conference*.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. (2016). LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, PP(99).

Heck, J. and Salem, F. M. (2017). Simplified minimal gated unit variations for recurrent neural networks. *arXiv preprint arXiv:1701.03452*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*.

Jatowt, A., Au Yeung, C.-M., and Tanaka, K. (2013). Estimating document focus time. In *Proceedings of the ACM International Conference on Information Knowledge Management*.

Jong, F. d., Rode, H., and Hiemstra, D. (2005). Temporal language models for the disclosure of historical text. In *Proceedings of the International Conference of the Association for History and Computing*.

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016). Bag of tricks for efficient text classification. *Computing Research Repository*, abs/1607.01759.

Jozefowicz, R., Zaremba, W., and Sutskever, I. (2015). An empirical exploration of recurrent network architectures. In *Proceedings of the International Conference on Machine Learning*.

Juan, A. and Ney, H. (2002). Reversing and smoothing the multinomial naïve Bayes text classifer. In *Proceedings of the International Workshop on Pattern Recognition in Information Systems*.

Kanhabua, N., Blanco, R., and Nørvåg, K. (2015). *Temporal Information Retrieval (Foundations and Trends in Information Retrieval)*. Now Publishers Inc, Hanover, MA, USA.

Kanhabua, N. and Nørvåg, K. (2008). Improving temporal language models for determining time of non-timestamped documents. In *Proceedings of the European Conference on Research and Advanced Technology for Digital Libraries.*

Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. *Proceedings of the International Conference for Learning Representations.*

Kumar, A., Lease, M., and Baldridge, J. (2011). Supervised language modeling for temporal resolution of texts. In *Proceedings of the ACM International Conference on Information and Knowledge Management.*

Kusner, M. J., Sun, Y., Kolkin, N. I., and Weinberger, K. Q. (2015). From word embeddings to document distances. In *Proceedings of the International Conference on Machine Learning.*

Langford, J., Li, L., and Zhang, T. (2009). Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10(Mar).

Le, Q. V., Jaitly, N., and Hinton, G. E. (2015). A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941.*

Li, Y., Genzel, D., Fujii, Y., and Popat, A. C. (2015). Publication date estimation for printed historical documents using convolutional neural networks. In *International Workshop on Historical Document Imaging and Processing.*

Liu, T.-Y. (2009). Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval.*

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval.* Cambridge University Press, New York, NY, USA.

Martins, A. F. T. and Astudillo, R. F. (2016). From softmax to sparsemax: A sparse model of attention and multi-label classification. In *Proceedings of the International Conference on Machine Learning.*

Mihalcea, R. and Tarau, P. (2004). Textrank: Bringing order into texts. In *Association for Computational Linguistics.*

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *Computing Research Repository.*

Mikolov, T., Joulin, A., Chopra, S., Mathieu, M., and Ranzato, M. (2014). Learning longer memory in recurrent neural networks. *arXiv preprint arXiv:1412.7753.*

Mishra, A. and Berberich, K. (2016). Estimating time models for news article excerpts. In *Proceedings of the ACM International on Conference on Information and Knowledge Management.*

Ng, A. Y. and Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naïve Bayes. In *Proceedings of the Annual Conference on Neural Information Processing Systems.*

Niculae, V. and Blondel, M. (2017). A Regularized Framework for Sparse and Structured Neural Attention. *arXiv preprint arXiv:1705.07704.*

Niculae, V., Zampieri, M., Dinu, L. P., and Ciobanu, A. M. (2014). Temporal text ranking and automatic dating of texts. *Conference of the European Chapter of the Association for Computational Linguistics.*

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing.*

Popescu, O. and Strapparava, C. (2015). Semeval 2015, Task 7: Diachronic text evaluation. In *Proceedings of the International Workshop on Semantic Evaluation.*

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6).

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). Learning representations by back-propagating errors. *Cognitive Modeling*, 5(3).

Salaberri, H., Salaberri, I., Arregi, O., and Zapirain, B. (2015). Ixagroupehudiac: A multiple approach system towards the diachronic evaluation of texts. In *Proceedings of the 9th International Workshop on Semantic Evaluation.*

Salton, G., Wong, A., and Yang, C. S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11).

Strötgen, J. and Gertz, M. (2016). Domain-sensitive temporal tagging.

Szymanski, T. and Lynch, G. (2015). Ucd: Diachronic text classification with character, word, and syntactic n-grams. In *Proceedings of the 9th International Workshop on Semantic Evaluation.*

Tao, T. and Zhai, C. (2007). An exploration of proximity measures in information retrieval. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Weissenborn, D. and Rocktäschel, T. (2016). Mufuru: The multi-function recurrent unit. In *Proceedings of the Association for Computational Linguistics Workshop on Representation Learning for NLP*.

Willmott, C. J. and Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate Research*, 30(1).

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. (2016). Hierarchical attention networks for document classification. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Zampieri, M., Ciobanu, A. M., Niculae, V., and Dinu, L. P. (2015). Ambra: A ranking approach to temporal text classification. In *Proceedings of the 9th International Workshop on Semantic Evaluation*.

Zhai, C. and Lafferty, J. (2004). A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 22(2).

Zhang, Y. and Wallace, B. (2015). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *Computing Research Repository*.

Zhou, G.-B., Wu, J., Zhang, C.-L., and Zhou, Z.-H. (2016). Minimal gated unit for recurrent neural networks. *International Journal of Automation and Computing*, 13(3).

Zhu, X., Ghahramani, Z., and Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the International Conference on Machine Learning*.