Departament de Sistemes Informàtics i Computació
Universitat Politècnica de València

Named Entity Recognition in multilingual handwritten texts

MASTER'S THESIS

Master's Degree in Artificial Intelligence, Pattern Recognition and Digital Imaging

Author: David Villanova Aparisi
Director: Carlos David Martínez Hinarejos
External director: Verónica Romero Gómez

Year 2020-2021

Resum

En el nostre treball presentem un únic model basat en aprenentatge profund per a la transcripció automàtica i el reconeixement d'entitats nomenades a partir de textos manuscrits històrics. Aquest model aprofita les capacitats de generalització dels sistemes de reconeixement, combinant xarxes neuronals artificials i n-grames de caràcters. Es discuteix l'avaluació d'aquest sistema i, com a conseqüència, es proposa una nova mètrica d'avaluació basada en el concepte de distància d'edició. Amb la finalitat de millorar els resultats respecte a aquesta mètrica, s'avalua l'aplicació d'estratègies de correcció d'errors i la descodificació a partir de les *n* millors hipòtesis per a l'obtenció de transcripcions sintàcticament correctes.

Paraules clau: Reconeixement de text manuscrit *offline*, reconeixement de entitats nomenades, sistema de reconeixement combinat, exploració de les *n* millors hipòtesis

Resumen

En nuestro trabajo presentamos un único modelo basado en aprendizaje profundo para la transcripción automática y el reconocimiento de entidades nombradas a partir de textos manuscritos históricos. Este modelo aprovecha las capacidades de generalización de los sistemas de reconocimiento, combinando redes neuronales artificiales y n-gramas de caracteres. Se discute la evaluación de dicho sistema y, como consecuencia, se propone una nueva métrica de evaluación basada en el concepto de distancia de edición. Con el fin de mejorar los resultados con respecto a dicha métrica, se evalúa la aplicación de estrategias de corrección de errores y la decodificación a partir de las *n* mejores hipótesis para la obtención de transcripciones sintácticamente correctas.

Palabras clave: Reconocimiento de texto manuscrito *offline*, reconocimiento de entidades nombradas, sistema de reconocimiento combinado, exploración de las *n* mejores hipótesis

Abstract

In our work we present a single Deep Learning based model for the automatic transcription and Named Entity Recognition in historical handwritten texts. Such model leverages the generalization capabilities of recognition systems, combining Artificial Neural Networks and n-gram character models. The evaluation of that system is discussed and, as a consequence, a novel evaluation metric, built upon the edit distance concept, is proposed. As a means to improve the results in regards to such metric, the application of error correction strategies is assessed, as well as the exploration of the *n*-best hypothesis to obtain syntactically correct transcriptions.

Key words: Offline handwritten text recognition, named entity recognition, combined recognition system, *n*-best hypothesis exploration

Contents

List of Figures

List of Tables

CHAPTER 1
Introduction

1.1 Motivation

There is a multitude of reasons for which to work with Handwritten Text Recognition (HTR) [24]. Even with the appearance of new technologies, we still employ written text as a means to communicate and interact. Being able to obtain automatic transcriptions from such signal may be useful for a number of day-to-day tasks, and may be done via online [34] or offline approaches [41].

Moreover, a large volume of historical archives are not digitized, which makes them vulnerable to degradation as time goes on. In order to preserve such information, an effort must be made to obtain digital versions of such documents. To expand on this issue, it must be mentioned that the documents which were digitized are mostly available as collections of scanned pages. Working with such format is a time consuming task, as the writing of many of the titles is obfuscated; many times the services of a paleographer are required to decipher the contents of the images, resulting in a significant monetary expense.

The main objective of historical HTR [43] is to build robust systems which are capable of obtaining accurate transcriptions from the scanned pages provided. Such task is an open area of study, as the present technology is capable of giving good results but still has to deal with the challenges that each corpus provides.

However, there are many collections of records in which obtaining a perfect transcription is not necessarily the objective. In these cases, the goal is rather to perform some kind of information retrieval from the images, targeting specific fields within the records. A way to solve this problem is to rely on Named Entity Recognition (NER) [32], which is a process that allows to identify parts of text based on their semantic meaning, such as proper names or dates. The current NER technology employs Natural Language Processing (NLP) models which are trained from clean text, making it susceptible to input errors.

This work aims to not only obtain valid transcriptions, but also to perform NER in such process. The result of this combined approach is the transcription of the text together with the tags that help identify and categorize the Named Entities. The addition of such tags not only provides some semantic information, but is also a necessary step towards the construction of a collection of linked documents.

To give an intuition on how such database would operate, each document would need to be tagged with the Named Entities that appear in it. The users would then be able to query through the entirety of the collection by providing Named Entities, obtaining a list of the documents related with such Entities. We must reiterate that the implementation of such concept is not within the scope of this work, but rather a greater ambition to pursue.

Our work approaches the task of historical HTR and NER over a multilingual corpus. Our choice for such a corpus is motivated by the expected application of this system. In a real case, the provided solution should be able to deal with the writing style of different authors and different languages. The selected corpus can be seen as a valid sandbox on which to build such system.

1.2 Objectives

The main objective of the proposed work is the study of a Machine Learning approach that is capable of performing both HTR and NER over a multilingual corpus written by several authors. To this end, the employed architecture will be a combined model that employs character n-grams as language models and an Artificial Neural Network to model the morphological knowledge.

To verify the behavior of the approach, a comparison with the state of the art for this task will have to be done. The goal will be then to evaluate the quality of the hypotheses of our system in terms of different well-established metrics such as: Character Error Rate (CER), Word Error Rate (WER), Precision, Recall and F1 score.

Once the performance of our model is up to par, our next target will be the design and implementation of a metric which evaluates the system with regards to its final application. Before that, an exploration of different evaluation strategies will have to be done as a means to obtain some insight into this matter.

Once such metric is well defined and developed, our final step will be to improve the score that our model obtains with such evaluation. For this purpose, the usage of error correction strategies will be assessed. Moreover, alternative decoding methods that rely on the exploration of the n-best hypothesis will also be tested.

To summarize, our goals in this work can be listed as:

1. Implement a combined model capable of performing HTR and NER over a multilingual corpus.
2. Compare the performance of such model with the state of the art for the given task.
3. Design and develop a metric that takes into account the final application of the system during its evaluation.
4. Improve the score of our model with such evaluation, exploring different alternatives without modifying the structure nor the parameters of the system.

1.3 Document structure

This document is structured in six different chapters. In Chapter 2, we will present an analysis on the recent contributions to both HTR and NER, as well as the technology being used to solve the combined task. In Chapter 3, the theoretical foundation on which our work is based will be introduced. Following that, in Chapter 4 we will delve into the experimental method employed and the results obtained with our approach. In Chapter 5, some insight will be given on the nature of such results as they are discussed. Chapter

6 will conclude the presentation of our work, giving a brief summary on the main contributions of our project. In this last chapter, we will also present different routes to expand upon what has been developed.

CHAPTER 2
State of the art

In this chapter we give an overview of the current approaches being used on historical HTR tasks, as well as the difficulties that are commonly faced in this matter. After that, we present a brief description of the early approaches that were used in NER and how the technology has evolved towards the current state-of-the-art. Finally, we introduce the current methods employed in tasks which require both HTR and NER, as well as the results previously obtained in our task.

2.1 Handwritten Text Recognition in historical documents

The goal of HTR is to extract the most likely transcription, $\hat{w}$, from a source image of text, x. The technology to solve this problem has evolved overtime similarly to that which has been employed in Automatic Speech Recognition (ASR). The equation that defines the HTR problem is, given the probability distribution p, as follows:

$$\hat{w} = \underset{w}{\operatorname{argmax}}\, p(w \mid x) \tag{2.1}$$

There are many examples in which automatic transcription techniques have been applied to historical writings. In [9], the performance of different models was assessed on the task of transcribing ancient scientific handwritten documents in Arabic. Such task was split in a sequence of three sub-tasks: page segmentation, text line extraction and optical character recognition. This pipeline of processes is commonly found when dealing with conventional HTR tasks [42, 20].

One of the competitors in that task was based on Google Cloud Vision [50], which employs Convolutional Neural Networks (CNN) [26, 1] to detect and extract the lines. After that, some heuristics determine the direction of the writing and the writing style. Such characteristics are then presented to the text recognition model, also based on CNN, so that it can make better predictions.

However, recent research has shown that it is possible to obtain complete transcriptions without resorting to document analysis and line extraction. In [46], new state-of-art performance is obtained on paragraph level recognition on the IAM dataset [29]. On the other hand, the performance on single-line recognition still falls short when compared to the aforementioned approach.

Overall, we have seen how supervised Machine Learning is capable of obtaining accurate transcription on corpora of varying complexity. Nevertheless, the commercial application of these systems is subject to a number of difficulties to overcome. First of all, the quality of the scanned images tends to be poor as the text is deteriorated with the passing of time. Moreover, there is a lack of labeled samples as the manual transcription of these texts tends to be costly, since many times the services of a paleographer are needed to decipher the writing. Such complications are detrimental to the performance of supervised learning approaches.

In [7], the mentioned problems are faced by making use of transfer learning [35] on a Generative Adversarial Network [16]. With this kind of approach, the starting point is a pre-trained model on which a small number of learning iterations are made over the available training samples of the target task. The correct initialization of the parameters of the model via transfer learning alleviates the effect of having a reduced data set.

However, the usage of transfer learning is not the only way to tackle this issue. In [23], a Convolutional Recurrent Neural Network is used for reading Tibetan manuscripts. This network is trained from synthetic data, generated by performing data augmentation [12] over the input samples by adding artificial noise. The model is then adapted to the real samples by following a regularization process.

When evaluating this system, the original labeled samples are used. As the artificially generated data consists of complete lines and the samples are pages with their transcription, a line segmentation process is necessary. The model in charge of performing this process learns its parameters using unsupervised learning methods.

Line segmentators usually learn their parameters from samples, either using supervised or unsupervised methods. In [22], the authors present a line extraction method that does not learn its parameters, but uses a distance graph as a projection of the document and works on this graph to achieve the segmentation. This is an example on how to deal with the lack of samples, this time regarding the image segmentation task.

2.2 Named Entity Recognition

The goal of NER is to identify parts of text with specific semantic meaning, such as proper names or dates. The input to this process is some sequence of text, w, and the output is the same text with the most likely tags that identify the Named Entities, $\hat{t}$. The equation that defines the NER problem is, given the probability distribution p, as follows:

$$\hat{t} = \underset{t}{\operatorname{argmax}}\, p(t \mid w) \tag{2.2}$$

The evolution in NER technology is defined by the advances that have been made in the world of Machine Learning. Early proposals for solving this problem were based on the application of hand-crafted rule sets, which could arise from large collections of names [44] or from patterns at the lexical or syntactic level. In any case, these proposals required exhaustive data sets, because of their characteristic intolerance to words outside the vocabulary or texts from a domain other than the one for which the systems were designed.

Another classical approach is the application of methods in the framework of supervised learning focused on feature extraction. In this way, the NER problem can be transformed into a multi-class classification problem or a sequence annotation problem. In either case, the starting point is a set of annotated samples from which feature extraction must be performed for the correct representation of the data. This process is of vital importance for the construction of a good supervised NER system, which is why we can find a multitude of alternatives for the representation of words in the form of a vector of Boolean or numeric values.

The use of word-level features such as morphology or morphosyntactic tagging [57] or document or corpus features [30] are examples that have been applied. From these feature vectors, a multitude of supervised Machine Learning algorithms have been applied, including Hidden Markov Models [39], Decision Trees [38] and Support Vector Machines [10].

Motivated by the scarcity of labeled samples, alternatives based on unsupervised methods emerged. One of the most common technologies in this case is clustering, which sees its application in NER systems that extract proper nouns from terms grouped by contextual similarity [33]. The fundamental idea behind these methods is the use of lexical resources, lexical patterns and statistics extracted from large corpora for the inference of proper noun mentions. An example of the use of this technology can be found in [56], where an unsupervised approach is applied to the extraction of named entities in biomedical text.

The state of the art in NER, as in multiple Machine Learning tasks, is dominated by Deep Learning models [45]. Advances in computational power with the use of Graphical Processing Units, together with the discovery of techniques that allow working with large numbers of layers, have led to the creation of neural models capable of learning millions of parameters to solve classification tasks of almost any nature; all this in the last 5 years of research [18, 52, 51].

The solving of NER by Deep Learning techniques is based on representing the input string as a sequence of embeddings obtained from a neural model. Although there are many alternatives in this matter, the recent use of machine translation architectures such as the Transformer [40] or the more recent BERT [11] stand out. This representation in the form of a numerical vector, capable of accounting for semantic relations, is used as an input for a network that generates a sequence of labels as output. These models can make use of recurrent networks or pointer generation [27] to take advantage of contextual information, although there are other more classical alternatives such as the use of a multilayer perceptron [14] or conditional random fields [48].

Another component that is commonly used used alongside those already presented is what are known as Language Models. These models allow the use of contextual information during classification, forming a prediction about the next hypothesis from previous hypotheses. Research on these models is linked to the world of Computational Linguistics, although they have seen their application in a multitude of Machine Learning tasks. Currently, one of the most widespread are the n-grams, whose operation is explained in Chapter 3 of [21]. Neural approaches to Language Modelling have also been explored [4] and have recently gained relevance [19, 25].

2.3 HTR and NER in historical texts

As we have seen, the goal of HTR is to obtain the transcription from scanned documents while that of NER is to tag the Named Entities in digital text files. When trying to perform HTR and NER over scanned documents, the first approach that may come to mind would be to first obtain the transcription and to then apply NER techniques over such output to obtain the tagged digital text. The alternative to the sequential method is the development of an end-to-end model that obtains the tagged output from the scanned documents in one step.

Recent results indicate that the application of end-to-end models can improve the performance on such task [8] by avoiding error propagation, which arises as a consequence of the sequential approach. The cause for this type of error is the fact that the model which performs NER expects a clean input, similar to that with which it has been trained. However, the output of the HTR process is not exempt from mistakes, hindering the performance of the NER model. Such effect has also been seen in other tasks of similar complexity. In [15], both speech recognition and NER are performed via an end-to-end model, obtaining better results than those produced with a sequential implementation.

This trend also seems to manifest in the results previously obtained in the corpus with which we are going to work. In [6], the "combined model" provides a stronger performance than that of the sequential approach during the evaluation phase. Moreover, the characteristics of the corpus are also thoroughly described there. This article is the starting point for our work. Our first model will try to emulate the end-to-end approach presented there. From then onwards, we will try to improve the obtained results.

CHAPTER 3

Theoretical foundation

In this chapter we will describe the theoretical foundation on which our work is based. We will start by presenting the fundamentals of the employed architecture. Next, we will go over the evaluation strategies that have been tried. Lastly, the error correction strategies with which we have experimented will be explained.

3.1 HTR and NER via end-to-end modelling

3.1.1. Problem formulation

Before introducing the chosen approach to deal with the problem, it may be useful to formalize its definition. First of all, it must be clarified that we are dealing with HTR at line level: the input to our system is the image area corresponding to a line in the text. The HTR problem may then be seen as the search for the most likely word sequence, $\hat{w} = \hat{w}_1\hat{w}_2...\hat{w}_t$, given the representation of the input line image: a feature vector sequence $x = x_1x_2...x_m$. This leads us to a search in the probability distribution $p(w \mid x)$:

$$\hat{w} = \underset{w}{\operatorname{argmax}}\, p(w \mid x) \tag{3.1}$$

If we consider the sequences of NE tags $t = t_1t_2...t_t$ related with the word sequence w as a hidden variable in equation 3.1, we obtain the following equation:

$$\hat{w} = \underset{w}{\operatorname{argmax}} \sum_t p(w, t \mid x) \tag{3.2}$$

By using Bayes' rule to decompose the probability in terms of optical and syntactical knowledge and approximating the sum to the maximal element, we get the following:

$$\hat{w} = \underset{w}{\operatorname{argmax}} \sum_t p(x \mid w, t) \cdot p(w, t) \tag{3.3}$$

$$\approx \underset{w}{\operatorname{argmax}} \max_t p(x \mid w, t) \cdot p(w, t) \tag{3.4}$$

From the equation 3.4 we can explicitly search for the most likely tagging sequence $\hat{t}$ during the decoding process, resulting in:

$$(\hat{w}, \hat{t}) \approx \underset{w,t}{\operatorname{argmax}}\, p(x \mid w, t) \cdot p(w, t) \tag{3.5}$$

If the information contained in t was added to the transcription, w, we would obtain the tagged transcription h. The resulting equation 3.6 resembles that of the original HTR problem, but taking into account that the hypothesis, h, to be generated is not only the transcription, but also the best tagging:

$$\hat{h} \approx \operatorname*{argmax}_{h} p(x \mid h) \cdot p(h) \tag{3.6}$$

3.1.2. Hybrid architecture

As we have mentioned in the previous formulation, we require a system that is capable of estimating both the optical probability, $p(x \mid h)$, and the syntactical probability, $p(h)$. The chosen architecture employs a Convolutional Recurrent Neural Network (CRNN) [54] to estimate the optical probability and a character n-gram to estimate the syntactical one. Both models are combined following the approach presented in [5].

To give some insight on the behavior of the proposed model, we start by analyzing the operation of the CRNN. The input to the optical model is a sequence of frames, $x = x_1x_2...x_m$, that is extracted from the text line to be transcribed. Each one of these frames, x_i, is processed sequentially by the convolutional part of the neural model, obtaining a tensor, $f(x_i)$, of fixed dimensions. The operation of this convolutional block can be seen, thus, as a data-driven feature extraction process.

This representation of the input frame, $f(x_i)$, is one of the two inputs to the succeeding recurrent block, the other one being the output state from the previous frame, x_{i-1}. The recurrent block performs some linear operations with non-linear activation functions to finally estimate the optical probability given the current frame. During the computation of the output for the following frame, x_{i+1}, the output state from the previous frame, x_i, will be used as an input, allowing the usage of past context in the computation of the optical probability. To better visualize this concept, we show a graphical representation in figure 3.1.

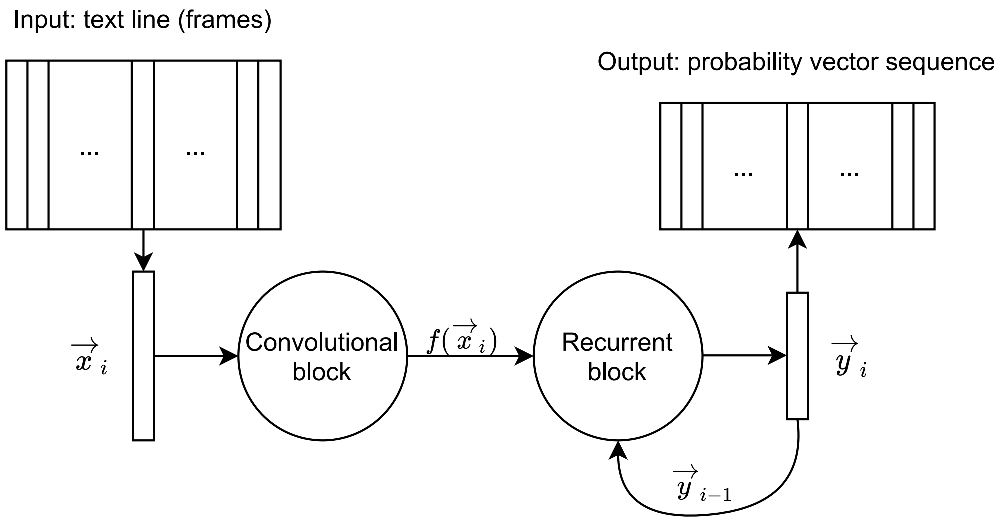


Figure 3.1: Simplified representation of the CRNN architecture.

To further describe how the hybrid architecture works, we follow our explanation with the analysis of the character n-gram language model. As we have mentioned, the role that this component plays is the estimation of the syntactical probability $p(h)$. This syntactical probability can be decomposed by applying the chain rule, obtaining the following expression:

$$p(h) = p(h_1) \cdot p(h_2 \mid h_1)...p(h_t \mid h_{t-1}, ..., h_1) \tag{3.7}$$

$$= \prod_{k=1}^{t} p(h_k \mid h_{1:k-1}) \tag{3.8}$$

However, the context that our model can handle is not infinite. To cope with this issue, such probability estimate is approximated by considering the $n-1$ previous characters and tags. The resulting equation is:

$$p(h) \approx \prod_{k=1}^{t} p(h_k \mid h_{k-n+1:k-1}) \tag{3.9}$$

The previous equation is implemented via a Stochastic Finite State Automata (SFSA) whose states are identified by the context being taken into account. The probability to transit to different states from a given one corresponds with the probability $p(h_k \mid h_{k-n+1:k-1})$. Figure 3.2 shows an example on how a 6-gram would operate on a HTR problem.

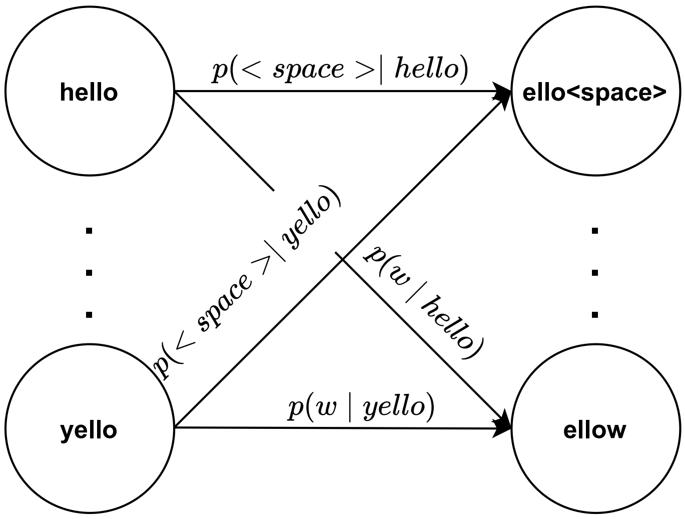


Figure 3.2: Simplified representation of a character 6-gram and its transition probabilities.

3.1.3. Training process

The training process of the presented architecture is structured in three parts which are represented in figure 3.3. First of all, the CRNN is trained over the training samples using the CTC loss function [17] and BackPropagation Through Time (BPTT) [53]. Then, the transition probabilities for the character n-gram language model are estimated over the transcriptions of the same training samples via Maximum Likelihood Estimation (MLE), a process which is described in Chapter 3 of [21].

As both models are trained individually, additional parameters should be trained in order to adjust the weight that we are giving to each classifier. To be more precise, we will use two parameters: the Optical Scale Factor (OSF) and the Word Insertion Penalty (WIP). The OSF serves the purpose of weighting the relevance of the optical model during the decoding. The WIP aims to ensure that the system is not biased towards shorter hypotheses. This last parameter is required because the usage of a character model tends to penalize longer hypotheses, as the probability of a given sequence of characters is defined by the product of the probability of each of its elements, which are numbers between zero and one. If we add those parameters to equation 3.6 and apply logarithmic notation, the expression that we obtain is:

$$\hat{h} \approx \underset{h}{\operatorname{argmax}}\, \alpha \cdot \log p(x \mid h) + \log p(h) - n\,Q \tag{3.10}$$

Where n is the length of the output sequences, Q is the WIP and α is the OSF. Both the OSF and WIP are estimated over the validation set, obtaining the combination that delivers the best performance for such set. With the computation of those parameters, the training process is concluded.

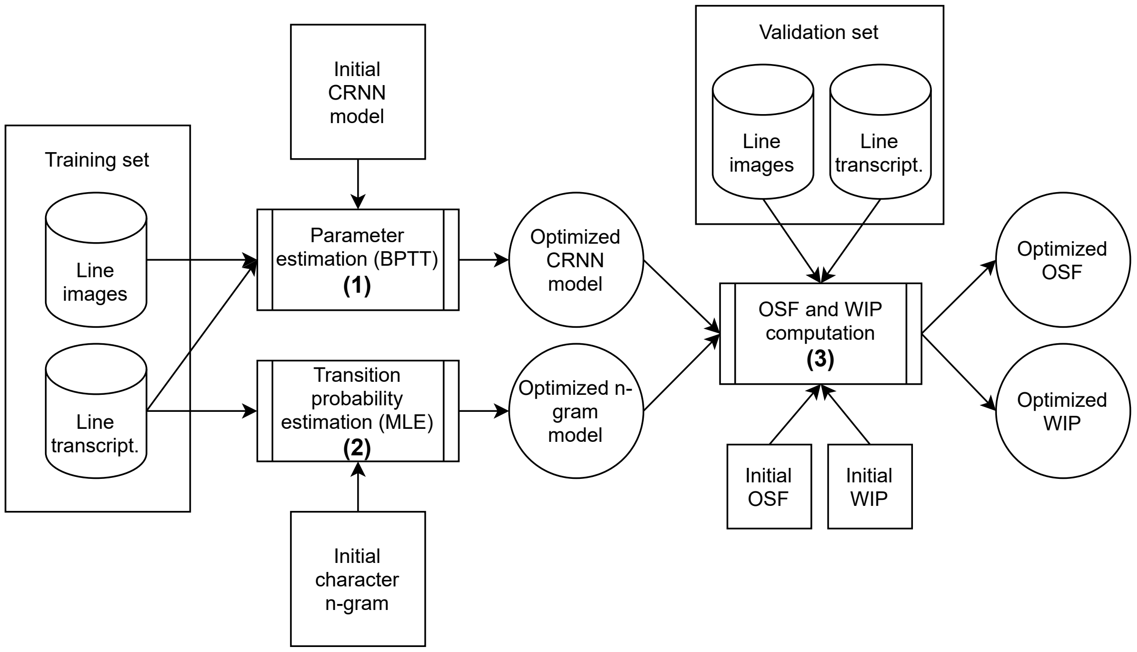


Figure 3.3: Representation of the training process and its three parts.

3.1.4. Viterbi decoding

When calculating the most probable transcription from the input signal, it is required to perform a left-to-right analysis of that signal taking into account the different possible outputs. However, when dealing with an exponentially growing number of hypotheses, obtaining the output becomes an algorithmic problem that requires an efficient solution. The solution to such problem is well known: the Viterbi algorithm [13].

The difficulty that arises from the usage of such algorithm is the memory consumption, as we need to store partial results to recover the best output sequence. The solution to this situation is a controlled sacrifice of the optimality by employing Beam Search [28]. The fundamental idea behind this decoding strategy is to periodically disregard those partial hypotheses which are unlikely to be the optimal one by comparing them to the current best partial hypothesis.

3.2 Evaluation metrics

3.2.1. Character and Word Error Rates

The Character Error Rate (CER) and Word Error Rate (WER) are both based on the edit distance concept and serve the purpose of measuring the similarity between two text sequences. More specifically, the CER is computed by calculating the Levenshtein distance [55] between the hypothesis and the ground truth sequence at character level and dividing it by the length of the ground truth sequence, thus measuring the character-level error. By considering i_c, s_c and d_c as the number of insertions, substitutions and deletions, respectively, and $n_c(t)$ the length of the target sequence, the CER between the hypothesis, h, and the target sequence, t, can be expressed as:

$$CER(h,t) = \frac{i_c + s_c + d_c}{n_c(t)} = \frac{D_c(h,t)}{n_c(t)} \tag{3.11}$$

The calculation of the WER is similar to that of the CER, the only difference being that the minimal unit of information is the word in this case. Therefore, the Levenshtein distance is calculated by considering the edit operations at word level and dividing it by the number of words in the ground truth sequence. This metric is notoriously more pessimistic than the CER, as a word with a single character error is considered a mistake upon comparison with the target. If we consider i_w, s_w and d_w as the number of insertions, substitutions and deletions, respectively, and $n_w(t)$ the number of words in the target sequence, the WER between the hypothesis, h, and the target sequence, t, can be expressed as:

$$WER(h,t) = \frac{i_w + s_w + d_w}{n_w(t)} = \frac{D_w(h,t)}{n_w(t)} \tag{3.12}$$

The usefulness of these metrics on the evaluation of the combined HTR and NER problem is debatable. It is clear that the edit distance can be used to compare two sequences of text. In our case, however, the focus does not rely on the quality of the transcription as a whole. Rather, the target is to identify the Named Entities correctly. The tags being employed are mere characters for the computation of the CER and the WER. As such, both metrics cannot evaluate the syntactical correctness of the output, thus limiting their usefulness on the evaluation of the proposed solution. Then again, we will still use the aforementioned metrics to compare our work to that of [6].

3.2.2. Precision, Recall and F1-Score

As we have seen, we cannot fully evaluate the system in terms of CER and WER. The next step in the evaluation process is, thus, to consider a more specific metric that can take into account the syntactical correctness of the output. The approach that is followed starts with the extraction of the Named Entities from each of the hypothesized tagged transcriptions and the ground truth ones. The result is a collection of files with just the detected Named Entities for each one of the lines, disregarding the rest of the transcription. With such information we can compute the Precision, Recall and F1-Score for the whole corpus. Such metrics require the computation of the number of True Positives (TP), False Positives (FP) and False Negatives (FN).

To calculate the number of TP and FP, we can explore the hypothesized Named Entities from each line and compare them to their ground truth counterpart. If a hypothesized Named Entity is present on the ground truth file, then it is considered a TP. Otherwise, it is a FP. A different way to calculate the TP would be to inspect the Named Entities in the ground truth lines and to check if they are present in the corresponding hypothesis. A Named Entity which is present in a reference file and not in the output of the system is considered a FN. To better visualize this computation, we propose the representation in figure 3.4.

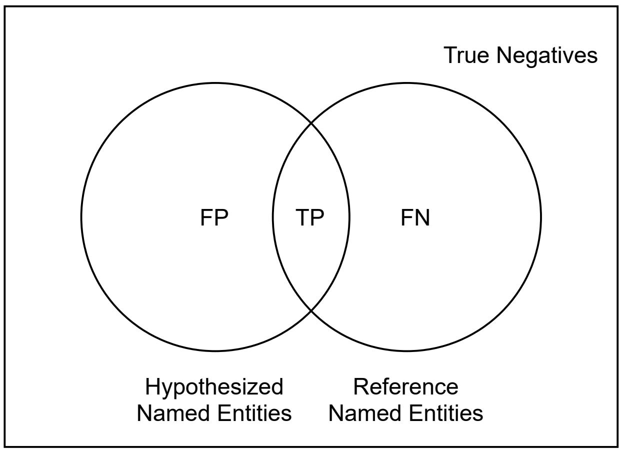


Figure 3.4: Representation of the computation of the TP, FP and FN for a single line.

Knowing the number of TP, FP and FN we can, then, calculate the Precision, Recall and F1-Score with the following equations:

$$P = \frac{TP}{TP + FP} \qquad R = \frac{TP}{TP + FN} \qquad F_1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{3.13}$$

As we can see, these metrics evaluate the performance of the system based on the detected Named Entities. However, their usage has some drawbacks worth mentioning. Firstly, order constraints are not being considered. If a Named Entity appears originally at the beggining of the sentence and, for some unknown reason, is hypothesized to appear at the end of the sentence, it would be considered as a TP.

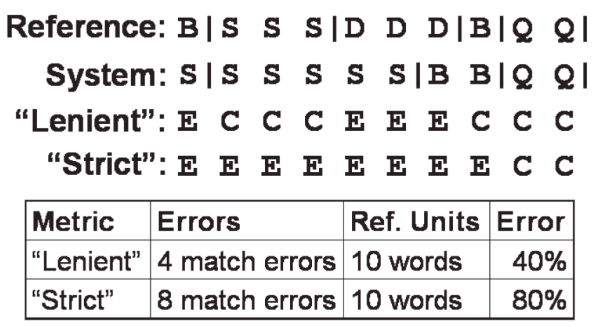
Secondly, the number of appearances of a Named Entity cannot be easily taken into account with this computation. To give an example, consider that a proper name appears originally one time in a line and that it is hypothesized that it appears a total of three times. In such a case, we may decide to account for three TP or just one, depending on the approach we follow to calculate them.

Lastly, there is also an issue regarding the strictness of the metric. A single character error in the transcription of a Named Entity would lead to consider it as a FP. Even if it is theoretically true that it is a mistake and that such Named Entity was not detected correctly, we may still be able to make use of those mistakes in the expected final application of the system. By applying approximate search [2] in a document collection, we should be able to retrieve those documents with small transcription mistakes.

3.2.3. Lenient and Strict

The drawbacks that arise with the evaluation of the system in terms of Precision and Recall have motivated the research in alternative evaluation strategies. The consideration of the Lenient and Strict [3] metrics, which were originally employed for Dialog Act Segmentation evaluation, is the next step in our search for an adequate metric.

In order to explain how both metrics work, we present figure 3.5. In this example we see how the sequence is tagged in the reference and in the output of the system, resulting in a sequence of dialog acts. The Lenient metric marks as correct the tags in the output that match the reference, independently of how the boundaries for its Dialog Act were set. The Strict metric, on the other hand, only marks as correct the tags in the output that match the reference and that belong to Dialog Acts with the same boundaries as the reference.



```
 Reference: B|S  S  S|D  D  D|B|Q  Q|
    System: S|S  S  S  S  S|B  B|Q  Q|
"Lenient":  E  C  C  C  E  E  E  C  C  C
 "Strict":  E  E  E  E  E  E  E  E  C  C
```

Metric	Errors	Ref. Units	Error
"Lenient"	4 match errors	10 words	40%
"Strict"	8 match errors	10 words	80%

Figure 3.5: Comparison between the Lenient and Strict metrics. Figure taken from [3].

The Strict metric takes into account the alignment of the output by considering the boundaries, which is a desirable property in the evaluation of our system. The implementation of these metrics, however, is not exempt from difficulties given the structure of the system output. First of all, the lengths of the reference and the system output do not match. Secondly, the expected output can have nested Named Entities. This poses an additional problem, as we would need to expand the number of possible categories to accommodate for this.

Overall, the metrics seem to incorporate some of the semantics that we would like to take into account. However, the translation of these metrics to our particular case is not immediate. As such, we have chosen to discard the direct usage of these metrics and to take inspiration from them to implement our own evaluation metric.

3.2.4. Entity CER and Entity WER

We have seen the advantadges that the Lenient and Strict metrics have, as well as the difficulties that arise when trying to employ them in the evaluation of the combined task. We have shown, as well, that the Precision and Recall are not exempt from drawbacks when it comes to the evaluation of the combined task. In order to face the drawbacks that were mentioned, a new evaluation metric is proposed. Such metric is based on the edit distance concept and it will evaluate the performance of the system via an analysis of the extracted Named Entities line by line, as it happened during the computation of both Precision and Recall.

More specifically, for each line of the testing corpus we are going to calculate the distance between the hypothesized Named Entities and the reference ones. This distance will be calculated similarly to how the CER and WER are calculated, except that in this case the minimal unit of information is a Named Entity, not a character nor a word. The edit operations that will be considered are the insertion, deletion and substitution of Named Entities. The associated cost to each of the operations can be expressed as:

$$I(i,j) = 1 \tag{3.14}$$

$$D(i,j) = 1 \tag{3.15}$$

$$S_{CER}(i,j) = \begin{cases} 2 & \text{if } E_i \neq E_j \\ 2 \cdot CER(T_i, T_j) & \text{otherwise} \end{cases} \tag{3.16}$$

$$S_{WER}(i,j) = \begin{cases} 2 & \text{if } E_i \neq E_j \\ 2 \cdot WER(T_i, T_j) & \text{otherwise} \end{cases} \tag{3.17}$$

The cost of both the insertion and deletion is set to one. The cost of the substitution, however, depends on several factors. First of all, if the type of Named Entities being compared do not match, $E_i \neq E_j$, the cost is 2: an absolute error of equal cost to an insertion and a deletion. Otherwise, the cost of the substitution depends on the similarity of the transcriptions. Such similarity can be the saturated CER or the saturated WER, depending on how strictly we want to evaluate the performance of the system. In any case, the cost of the substitution will be a value contained in the range $[0, 2]$.

If we consider a matrix of size $(|t| + 1) \times (|h| + 1)$, the Entity CER (ECER) or Entity WER (EWER) between two Named Entity lists is the cost of the best path that starts in position $i = j = 0$ and ends in $i = |t|, j = |h|$, being $|t|$ the length of the reference Named Entities' list and $|h|$ the length of the hypothesized one. The ECER is calculated according to the following recursive equation:

$$ECER(h,t) = C_{CER}(|t|, |h|) \tag{3.18}$$

$$C_{CER}(i,j) = \begin{cases} i \text{ if } j = 0 \\ j \text{ if } i = 0 \\ \min \begin{cases} D(i,j) + C(i,j-1) \\ I(i,j) + C(i-1,j) \\ S_{CER}(i,j) + C(i-1,j-1) \end{cases} \end{cases} \tag{3.19}$$

The EWER is calculated in a similar fashion, the difference being the consideration of the WER for the cost of the substitution. Figure 3.6 shows how the EWER could be calculated for an artificial sample. The recursive equation for the computation of the EWER is as follows:

$$EWER(h,t) = C_{WER}(|t|, |h|) \tag{3.20}$$

$$C_{WER}(i,j) = \begin{cases} i \text{ if } j = 0 \\ j \text{ if } i = 0 \\ \min \begin{cases} D(i,j) + C(i,j-1) \\ I(i,j) + C(i-1,j) \\ S_{WER}(i,j) + C(i-1,j-1) \end{cases} \end{cases} \tag{3.21}$$

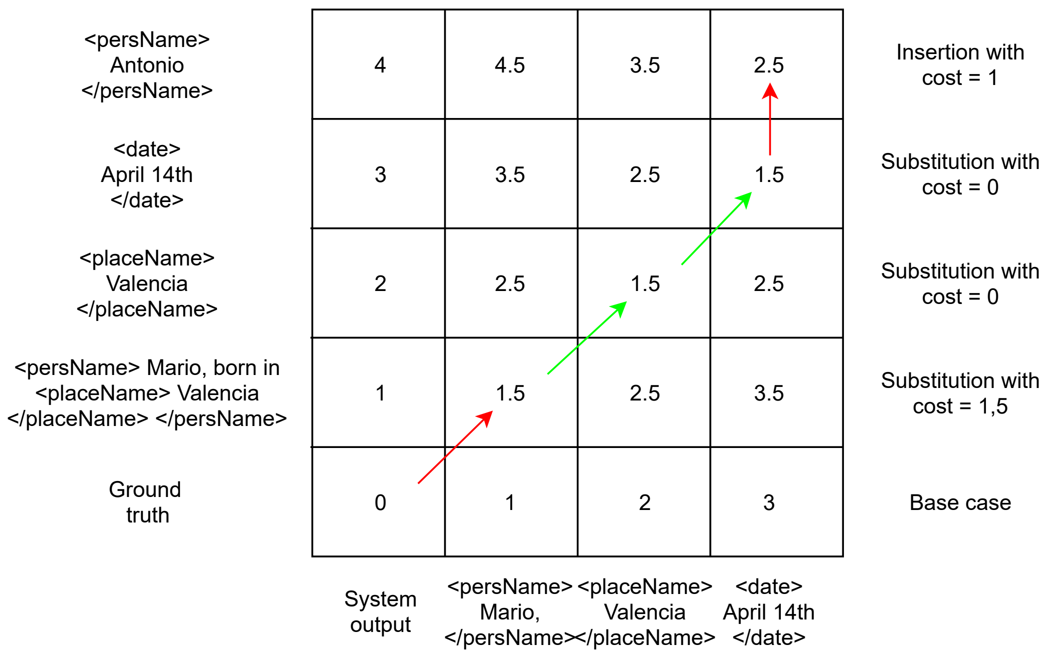

<persName> Antonio </persName>
<date> April 14th </date>
<placeName> Valencia </placeName>
<persName> Mario, born in <placeName> Valencia </placeName> </persName>
Ground truth
4
4.5
3.5
2.5
3
3.5
2.5
1.5
2
2.5
1.5
2.5
1
1.5
2.5
3.5
0
1
2
3
Insertion with cost = 1
Substitution with cost = 0
Substitution with cost = 0
Substitution with cost = 1,5
Base case
System output
<persName> Mario, </persName>
<placeName> Valencia </placeName>
<date> April 14th </date>

Figure 3.6: An example on how the EWER is calculated. Note that the best sequence of edit operations is indicated by arrows.

To better understand how the metric is computed in this particular case, we will analyze each of the edit operations in the optimal sequence of operations. The first operation that is considered is the substitution of the first detected Named Entity for the first Named Entity in the reference. As both of them have the same tags, the cost of the operation is twice the WER, which is calculated by comparing the text contained in each Named Entity. The following operations in the optimal path are substitutions without cost, as both the tagging and the text of the Named Entities match. The final operation is to insert the Named Entity "<persName> Antonio </persName>", which was not detected by the system. This operation carries a cost of 1. As we can see, the best sequence of edit operations leads us to an EWER of 2.5.

In order to evaluate the performance of the model with several samples, we need to extend the definition of the metric. To tackle this issue, we perform a weighted average over each of the lines, l, of the testing corpus, C, by considering the associated hypothesis, h_l, and the reference, t_l:

$$ECER(C) = \frac{\sum_{l \in C} ECER(h_l, t_l)}{\sum_{l \in C} |h_l| + |t_l|} \tag{3.22}$$

$$EWER(C) = \frac{\sum_{l \in C} EWER(h_l, t_l)}{\sum_{l \in C} |h_l| + |t_l|} \tag{3.23}$$

By definition, the proposed metrics deal with some of the drawbacks that the F1-Score presented. Firstly, these metrics consider the order in which the Named Entities appear during the computation of the score for each hypothesized line. Secondly, the number of appearances is also taken into account during the computation of said score, as having more cases of a Named Entity in the output will result in a higher number of deletion operations. Lastly, the strictness of the metric can be adjusted by considering the CER or the WER for the cost of the substitutions. In case of using some kind of approximate search technology for the document database, we may opt to employ ECER for the evaluation of our system.

3.3 Error correction

3.3.1. Error types

Before introducing the error correction strategies with which we have experimented, it may be convenient to perform a slight analysis of the type of errors that the system can produce. To this end, we present table 3.1.

Ground truth	System output	Type of error
Home	<placeName>Home </placeName>	The system identified an entity when there should be none.
<persName>Asunción </persName>	Asunción	The system missed an entity.
<persName>Asunción </persName>	<placeName>Asunción </placeName>	The system assigned a wrong label to an entity.
<persName>María Asunción </persName>	<persName>María </persName>	The system got the entity's boundaries wrong.
<persName>María Asunción </persName>	<placeName>María </placeName>	The system assigned a wrong label to an entity and missed its boundaries.
<persName>María Asunción </persName>	<persName>María González </persName>	The system obtained a transcription with mistakes.
<persName>María, born in <placeName> Spain </placeName> </persName>	<persName>María, born in <placeName> Spain </persName> </placeName>	The system did not close correctly the nested tags.
<persName>María </persName>.	<persName>María.	The system did not close an open tag upon line ending.
<date>April 14th </date>	April 14th </date>	A tag was closed without a matching opening tag.

Table 3.1: Types of errors to be expected in the produced output.

The system must be able to tolerate nested Named Entities as well as avoiding incurring in several decoding mistakes. However, these errors are not the only ones to account for. Via an analysis of the training and validation data, we observed that nested Named Entities follow specific structures. What we found was that two tags of the same type could not be nested one inside the other. As an example, a sequence that resembled something like "<placeName> ... <placeName> ... </placeName> </placeName>" would be impossible to find. Therefore, we decided to consider the production of these type of structures as a mistake.

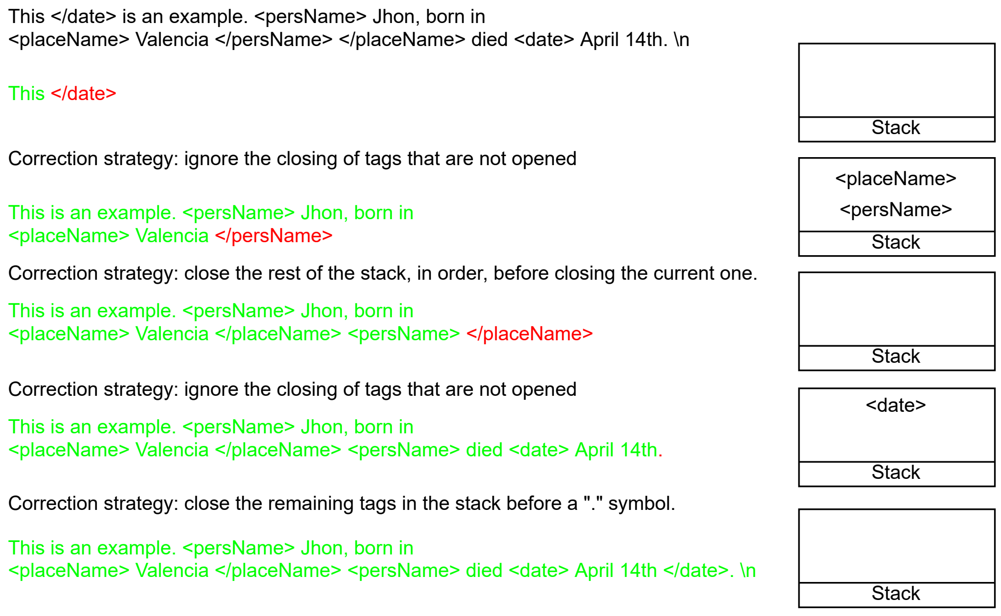
The error correcting strategies that we have designed and implemented try to amend some of the errors that were listed. More specifically, we will try to identify sequences that are syntactically wrong and correct them. As such, we will only deal with cases of incorrect closing of nested tags, tags that remain opened upon line ending, tags that were closed without a matching opening tag and sequences of nested Named Entities of the same type.

3.3.2. Heuristic approach

The first approach that may come to mind when trying to deal with syntactical errors would be to use Regular Expressions to search for erroneous structures and substitute them with the correct ones. Such approach may appear as valid as the language of the expected output is regular, having all nested Named Entities following specific structures that can be identified. However, the language of the actual output could be of greater complexity. As such, we would require a sizeable collection of regular expressions to identify each and every type of error; even in that case, the solution would be susceptible to erroneous structures which were not identified during the implementation.

A slightly more elegant solution would be to have some kind of algorithmic method that would be able to, by analyzing the output, identify the errors and apply corrections where needed. In this work, we attempted to implement such method by doing a left-to-right analysis on the output with an auxiliary stack to store state information.

To give an intuition on how the proposed method works, we give an example of its operation in figure 3.7. Note that the errors are detected by inspecting the state of the stack, and that the error correction strategy depends on the type of error being detected. If no error was detected, the method would produce the same output as its input line. Even if the proposed method is built to work with lines of text, it should be able to deal with whole paragraphs by construction.



This </date> is an example. <persName> Jhon, born in
<placeName> Valencia </persName> </placeName> died <date> April 14th. \n

This </date>

Correction strategy: ignore the closing of tags that are not opened

This is an example. <persName> Jhon, born in
<placeName> Valencia </persName>

Correction strategy: close the rest of the stack, in order, before closing the current one.

This is an example. <persName> Jhon, born in
<placeName> Valencia </placeName> <persName> </placeName>

Correction strategy: ignore the closing of tags that are not opened

This is an example. <persName> Jhon, born in
<placeName> Valencia </placeName> <persName> died <date> April 14th.

Correction strategy: close the remaining tags in the stack before a "." symbol.

This is an example. <persName> Jhon, born in
<placeName> Valencia </placeName> <persName> died <date> April 14th </date>. \n

Figure 3.7: An example on how the stack error correction operates, step by step.

However, this method is not exempt from errors. While its application appears promising, being theoretically able to amend some of the syntactical mistakes, the experimentation has revealed a degradation in system performance with the usage of this method. The possible reasons for this will be revealed in chapter 5.

3.3.3. Exploration of the n-best hypotheses

A different approach to try and improve the performance of the system would be to forgo additional error correction over the output and, instead, to improve the decoding process by including the restrictions of the desired output. The main idea here would be to force the system to extract hypotheses which are syntactically sound via an analysis of the output, operating with an auxiliary stack similarly as done before. However, in this case we do not correct the wrongful hypotheses, we discard them instead and search for correct hypotheses in their place.

The way in which such idea is implemented is simple. Instead of obtaining the best hypothesis during the decoding phase, we obtain a graph containing a sizeable number of hypotheses. This graph is called Lattice and it can be directly obtained with the usage of Kaldi.[1] This graph can be pruned to obtain a smaller version containing only the n-best solutions, n being a parameter which we can adjust. From this reduced Lattice we can directly obtain the sequence of the n-best hypotheses for an input line.

The decoding process for a specific line, l, starts with an ordered exploration, from the best hypothesis to the n-best. In such exploration, each hypothesis is evaluated in a process similar to the stack error correction. More specifically, the strategy consists of doing a left-to-right analysis with a stack that represents the state of the hypothesized Named Entities to detect syntactical errors. In other words, a hypothesis is valid when the output from the stack error correction matches the input. In case of finding an error, the i-best hypothesis is disregarded and the exploration continues. This search for the best valid hypothesis concludes when a hypothesis is found such that no error was detected via the stack error detection.

It may also happen that, during the search, no valid hypothesis is found. This is an exception that will happen less often as the parameter n increases, but it should still be considered. In those cases, the applied policy will be to keep the 1-best hypothesis as the selected output. Other policies, such as selecting the hypothesis which would require the least amount of corrections via the stack error correction method, may be considered. However, for consistency reasons, we are going to favor simplicity in the implementation.

The result of the customized decoding process will be the best valid hypothesis for each of the lines or, in case of not having found any correct output, the one with the most confidence. This error correction strategy differs from the previously mentioned one because it includes the structural requirements during the decoding process, instead of relying on the correction of the best hypothesis.

To give an intuition on how the syntactic requirements influence the decoding process, the effect is similar to interpolating the probabilities of the hypotheses with a Stochastic Context-Free Grammar (SCFG) learned from the training data. Such SCFG would give better probabilities to valid hypotheses, while lowering the probabilities for syntactically wrong outputs. This variation in probabilities would make the system prioritize valid outputs while still taking into account the probability that was estimated by the combined model, which is the same effect that the proposed n-best decoding has.

[1]Kaldi's documentation is available at: http://kaldi-asr.org/doc/

CHAPTER 4

Experimental setup and obtained results

4.1 Experimental setup

4.1.1. Description of the corpus

Our experiments have been carried out on a corpus of handwritten medieval charters, which was presented and thoroughly described in [6]. This corpus consists of 499 letters written by different authors in three different languages: Latin, Czech and German. Even though the letters date from 1145 to 1491, their ink and paper are well preserved as we can see in figure 4.1.

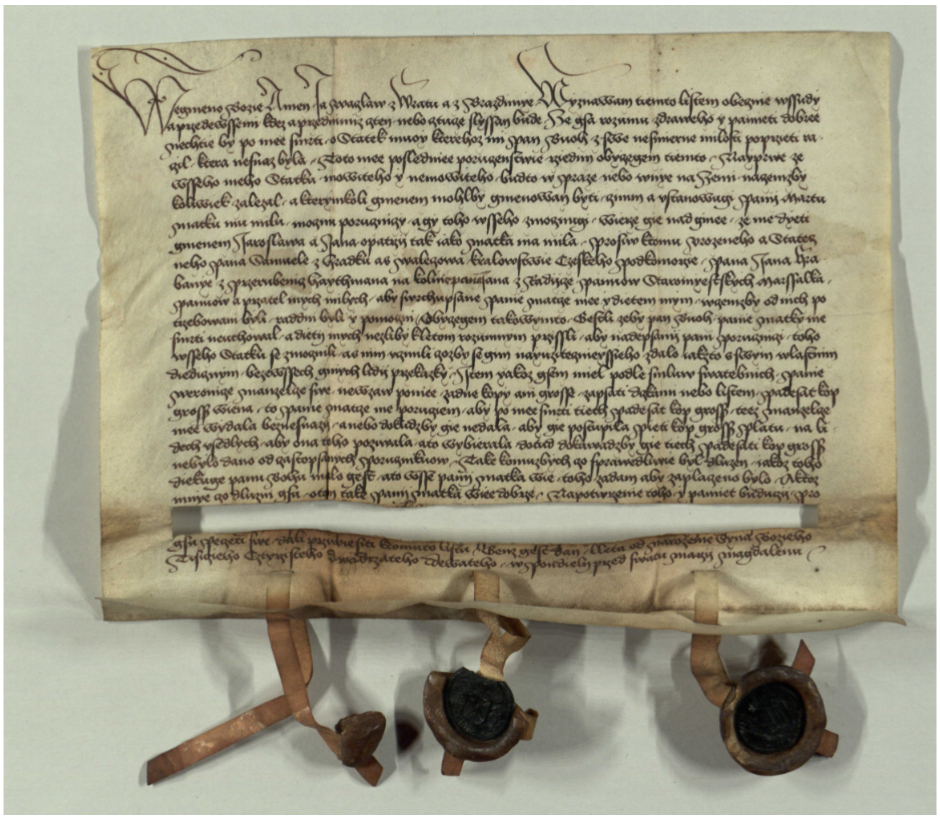

[illegible]

Figure 4.1: Scanned version of one of the available Czech charters.

As we have previously mentioned, the way in which the HTR and NER task is going to be carried out is via a combined model which performs both processes in one step. In order to train such model, we need data that contains both the transcriptions and its associated tags that identify the present Named Entities. This corpus has such kind of data available, as it can be seen in figure 4.2.

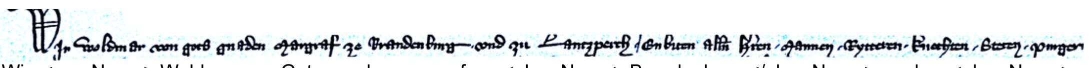

Wir <persName> Woldmar von Gots gnaden margraf ze <placeName> Brandenburg </placeName> und zu <placeName> Lantzperch </placeName> </persName> enbiten allen herren, mannen, rytteren, knechten, steten, purgern,

Figure 4.2: A single extracted line and its reference tagged transcription.

In order to be able to compare our work to that of the state of the art, the experiments should be carried out on even ground. For that reason, we are going to train and evaluate our models with the same partition scheme that was presented in the original work. Therefore, the available data is split into three parts: a training set containing 80% of the charters, a validation set with a 10% of the letters and a testing set with the remaining 10% of the samples. The details on how the data is partitioned are shown in table 4.1.

	Number	**Czech**	**German**	**Latin**	**All**
Train	pages	161	138	99	398
	lines	2,905	2,556	1,585	7,046
	tokens	52,708	60,427	28,815	141,950
	N. Entities	4,973	6,024	2,809	13,806
Validation	pages	21	18	12	51
	lines	300	252	150	702
	tokens	5,997	5,841	2,467	14,305
	N. Entities	440	461	188	1,089
Test	pages	20	17	13	50
	lines	381	388	229	998
	tokens	6,891	9,843	3,995	20,729
	N. Entities	467	744	295	1,506

Table 4.1: Experimental dataset split, equal to that proposed in [6].

Throughout the entirety of the corpus, we can only find three types of Named Entities: the name of a person, the name of a place and a date. As such, the model must be able to generate the corresponding opening and closing tags for each type of Named Entity. The number of different syntactical structures that can be generated, however, is potentially infinite due to the appearance of nested Named Entities.

During previous experimentation, the authors assumed two important simplifications over the original problem. First of all, Named Entities spanning over different lines (continued Entities) were split so that their apparitions are always contained within single lines. The other assumption regarded nested Named Entities. In this case, the authors decided to remove the nested entity. As an example, a date contained within a proper name would have its tag removed, so that there is no nesting.

Our work is based on a combined model that works at line level. Therefore, we are going to maintain the simplification assumed in the original work which consisted of splitting the continued Entities as not doing so would lead to a noticeable performance decrease. However, we are going to forgo the other simplification and try to deal with nested Named Entities. Our reasoning for this choice is that the combined model should be capable of dealing with this kind of structural complexity without compromising the performance.

4.1.2. Description of the chosen model

The employed architecture is based on the combined approach that was used in [6]. Therefore, we scale the line images to a height of 64 pixels and apply contrast enhancement and noise removal as described in [49]. No additional preprocessing is applied to the input.

The optical model consists of a CRNN with four convolutional layers, where the n-th layer has $16n$ 3×3 filters, and a LSTM unit of three layers of size 256 plus the final layer with a Softmax activation function. The values for the rest of the hyperparameters are the same that were used in [37]. Such model is implemented and trained with the PyLaia Toolkit [31].

The employed language model is a character 8-gram with Kneser-Ney back-off smoothing. The estimation of its probabilities is done with the SRILM Toolkit [47] by considering the tagged transcriptions in the training partition.

Both the CRNN and the character 8-gram are combined into a Stochastic Finite State Automata (SFSA) with the Kaldi Toolkit [36]. The OSF and WIP are, then, adjusted over the validation partition with the usage of the Simplex algorithm provided by the SciPy library.[1] Finally, the resulting SFSA is used to obtain the hypothesized tagged transcription for each line in the test partition.

4.2 Obtained results

Table 4.2 shows the best results obtained with each approach. Note that the first column corresponds to the previous results, which were taken as a reference for the implementation of our model. Therefore, it is impossible to evaluate the system with the proposed ECER and EWER metrics. The second column reports the results obtained with our version of the combined approach. The third and fourth column show the results obtained with the application of the error correcting strategies.

Metric	Boroş, Emanuela et al. [6]	Combined model (nested NEs)	Combined model + stack error correction	Combined model + 2500-best decoding
CER (%)	**8.00 ±1.68**	9.23 ±1.80	10.38 ±1.89	9.24 ±1.80
WER (%)	**26.80 ±2.75**	28.20 ±2.79	32.94 ±2.92	28.14 ±2.79
Precision (%)	**49.25 ±3.10**	43.14 ±3.07	31.48 ±2.88	40.05 ±3.04
Recall (%)	37.08 ±3.00	37.58 ±3.00	31.54 ±2.88	**39.97 ±3.04**
F1 (%)	**42.30 ±3.07**	40.17 ±3.04	31.51 ±2.88	40.01 ±3.04
ECER (%)	-	31.94 ±2.89	34.70 ±2.95	**28.69 ±2.81**
EWER (%)	-	46.62 ±3.10	56.46 ±3.08	**44.42 ±3.08**

Table 4.2: Reference results and evaluation scores of the best proposed models, with 95% confidence intervals.

[1]The documentation for the employed Simplex implementation is available at: https://docs.scipy.org/doc/scipy/reference/optimize.linprog-simplex.html

As we mentioned earlier, the exploration of the n-best hypotheses introduces a new parameter to adjust. In our experimentation, we have evaluated the effect that this parameter has by giving it different values and testing the performance of the resulting system. Such performance has been tested with each metric that has been presented, even though our focus is to optimize the proposed ECER and EWER metrics. Figures 4.3, 4.4 and 4.5 show the evolution of the different metrics with respect to the value of n.

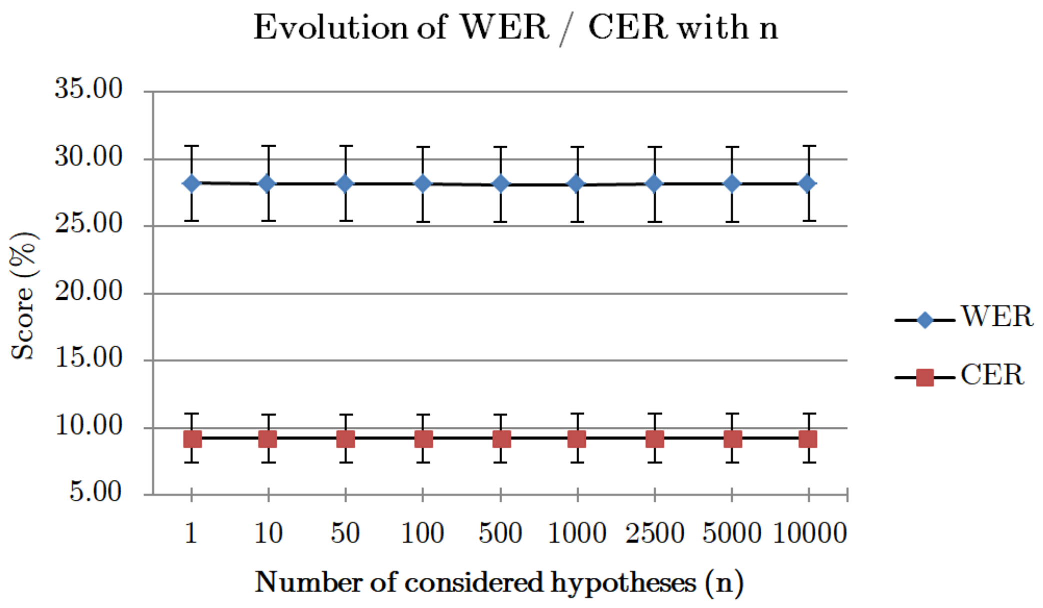


Figure 4.3: Evolution of CER and WER with different values of n. For each point in the plot the 95% confidence interval is shown.

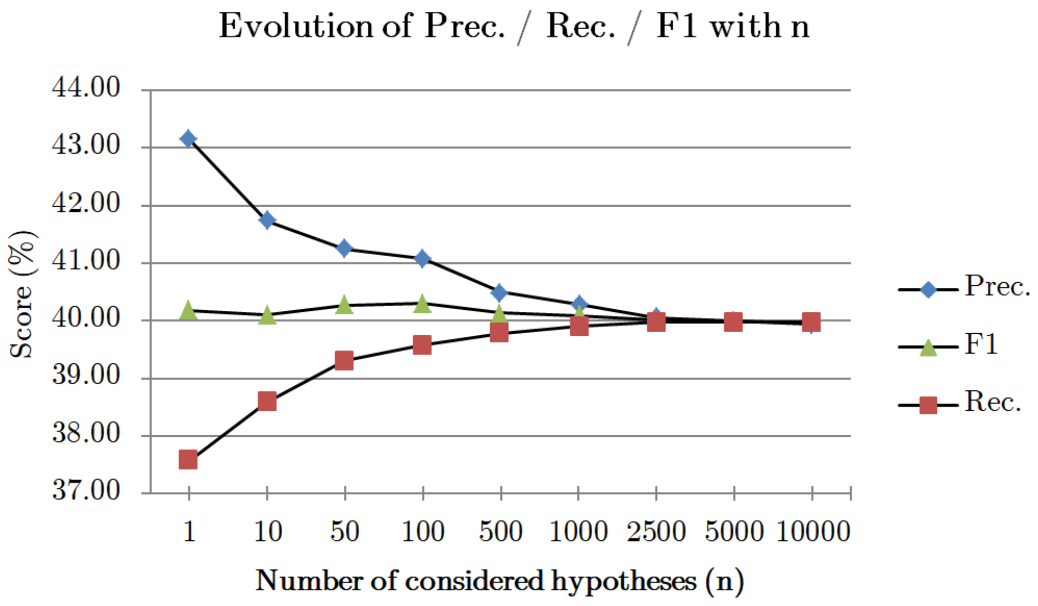


Figure 4.4: Evolution of Precision, Recall and F1-score with different values of n.

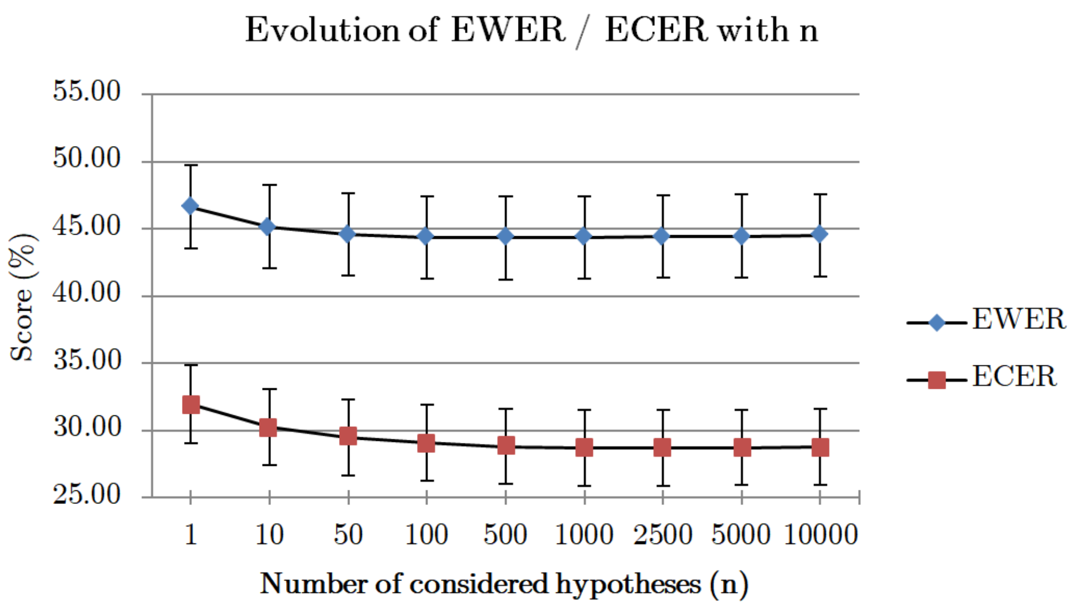


Figure 4.5: Evolution of ECER and EWER with different values of n. For each point in the plot the 95% confidence interval is shown.

CHAPTER 5

Discussion

During the design and implementation of the proposed solution, the possible effects of the proposed strategies were mere hypotheses. After the experimentation phase, some of those hypotheses have been discarded while others have been verified. In this chapter, we present the main ideas that can be deduced from the obtained results.

While we initially thought that the combined model would be able to deal with the increase in complexity due to the consideration of nested Named Entities, the results that were presented in table 4.2 show the contrary. There has been an important drop in performance in almost every metric which was considered in the previous work, although not a statistically significant one.

Our first attempt at improving the performance of the combined model has been to perform heuristic error correction. The idea behind this error correcting strategy is simple and seemingly well reasoned. However, the strategy fails at improving any of the proposed evaluation metrics.

The increase in both CER and WER with the usage of this correction strategy is due to the fact that we are adding additional symbols to the output of the system. While such symbols help guarantee the syntactical correctness of the output, the best hypothesis of the system did not have them for a reason. By adding those symbols, we are generating a much less likely output or, rather, a more error-prone one.

The decrease in Precision, Recall and F1 score is related to the mentioned fact, as well as to the process in which the Named Entities are extracted from the output. In such process, the Named Entities which were not properly opened or closed were disregarded, serving as some kind of error correction strategy. The results reveal, then, that ignoring the erroneous tagging in the output leads to better results than those obtained with active correction. The increase in the proposed ECER and EWER metrics is related to the same fact: the consideration of supposed Named Entities which the system could not properly tag.

The other strategy that we implemented is to consider the syntactical constraints during the decoding phase by obtaining the best valid transcription from a list containing the n-best hypotheses for each line. Our initial thought was that, by design, choosing less likely hypotheses would compromise the quality of the transcriptions in exchange for better ECER and EWER.

The obtained results show that that is not the case. In figure 4.3, we saw how there were slight variations in CER and WER with the increasing of n, but nothing significant. It turns out that the differences between the hypotheses are minor, as we can see in figure 5.1, and the transcription errors that may happen due to the consideration of a less likely hypothesis end up being offset by the imposition of correct tagging.

<persName>Woita otewec Ugolt, Ruda<placeName>Pocimus</persName>. Si qua in futurum ecclesiastica sclarisue persona in factum irritum facere temptaverit, vinculo

<persName>Woita otewec Ugolt, Ruda Pocimus</persName>. Si qua in futurum ecclesiastica sclarisue persona in factum irritum facere temptaverit, vinculo

Figure 5.1: Differences between the most likely hypothesis and the 11-best hypothesis, which is syntactically correct, in one of the test lines.

It is worth mentioning that, in figure 4.4, we saw how, with the increasing of n, the Precision worsened while the Recall kept increasing, both in a logarithmic fashion. Even if such difference is not statistically significant given the confidence interval that is being considered, it may be interesting to identify the cause for such trend.

The reason behind this trade-off is that the increase in the number of considered hypotheses, n, results in a noticeable increase of the number of Named Entities tagged by the system. With such increase, both the number of True Positives (TP) and False Positives (FP) rise, the second one at a faster rate than the first. As a consequence, Precision drops and Recall rises. The F1 score, however, does not vary significantly.

This evolution in the number of tagged Named Entities has its repercussion when we consider the proposed ECER and EWER metrics. In figure 4.5 we can see how there is an improvement in the performance of the system with both metrics. It must be mentioned, however, that such improvement is not statistically significant considering a 95% confidence interval. The usage of a bigger test partition may reveal the significance of such improvement.

There are two reasons for this increase in performance in both metrics. Firstly, we have seen an increase in the number of TP. Perfectly recognized Named Entities are greatly rewarded in the proposed metric by being substitutions with no cost. Moreover, the increase that we saw in FP may not be harmful to the quality of the hypotheses. Some of the Named Entities considered as FP may be correctly tagged and have small character errors. Such cases qualify as beneficial for the performance of the system as long as the CER or WER of the transcription is lower than 0.5, depending on the metric being used.

The decrease in ECER and EWER is logarithmic with respect to the increase of n, as we can see in figure 4.5. Logarithmic increases in performance were also spotted in the evolution of the Recall. To understand the reason behind this trend, we performed an analysis of the behavior of the system as the parameter n is increased.

The way in which such analysis was conducted is by recording, for each of the test lines, the index of the hypothesis which was selected as the output of the system. If the most likely hypothesis was the one being chosen because no valid hypothesis was found, the recorded index would be a 0, in contrast with the 1 that would be recorded if the 1-best hypothesis was valid. Having such data available, we calculated the frequency for each of the indexes and generated the histograms which can be seen in figures 5.2 to 5.6.

As we can see, by relying on the classical 1-best decoding, the system produces 248 syntactically incorrect outputs. With the usage of bigger values of n, the number of incorrect outputs decreases at a logarithmic rate. This trend justifies the logarithmic evolution that we have seen on the different evaluation metrics.

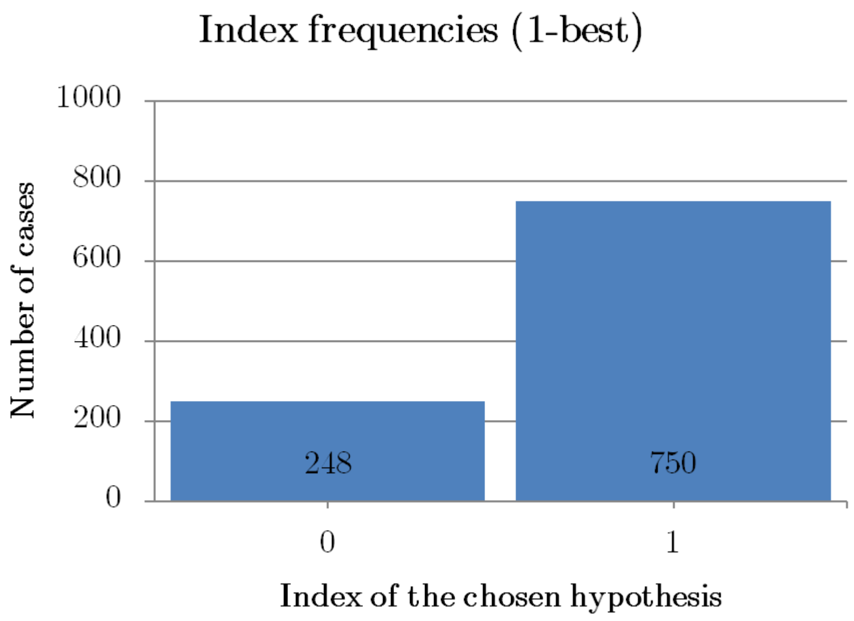


Figure 5.2: Frequency histograms of the indexes of the hypotheses with 1-best decoding.

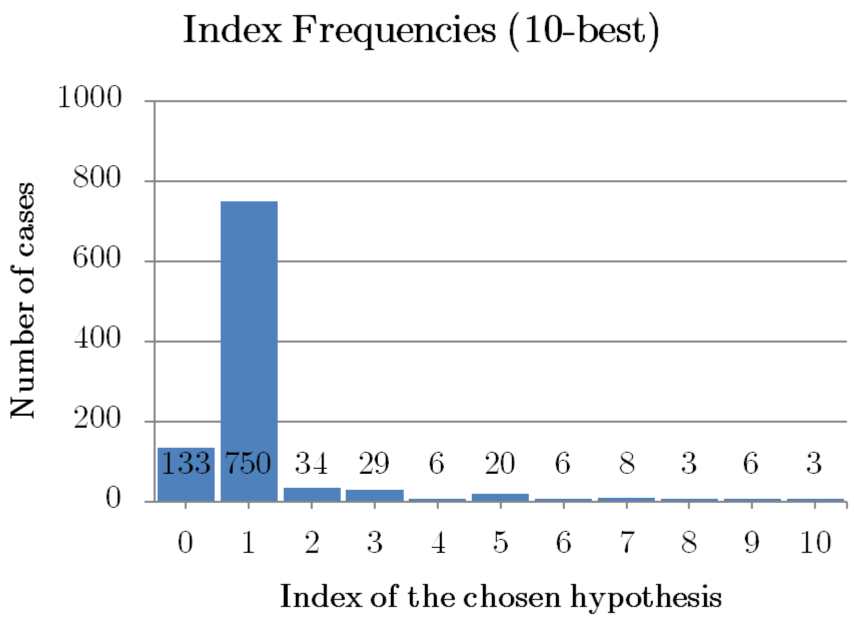


Figure 5.3: Frequency histograms of the indexes of the hypotheses with 10-best decoding.

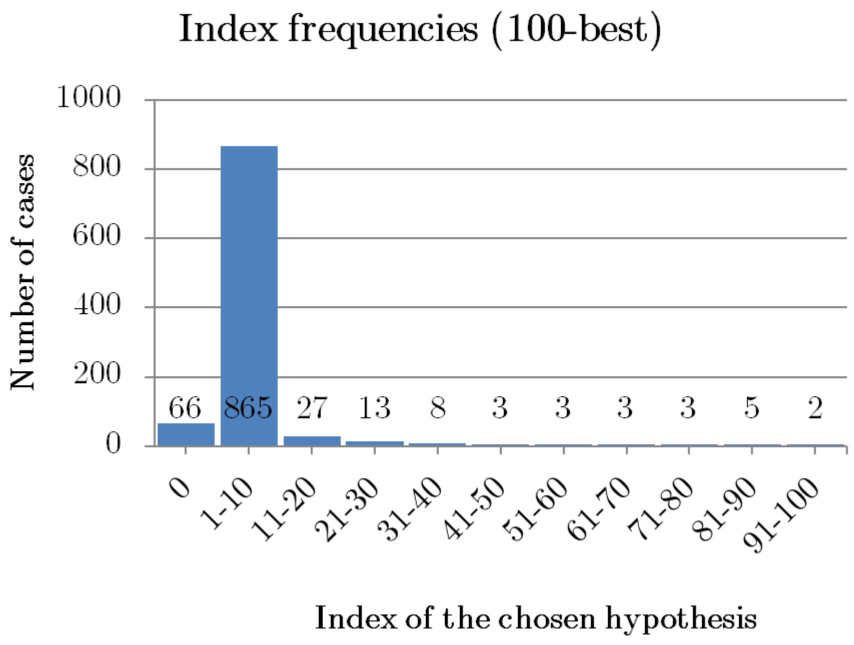


Figure 5.4: Frequency histograms of the indexes of the hypotheses with 100-best decoding.

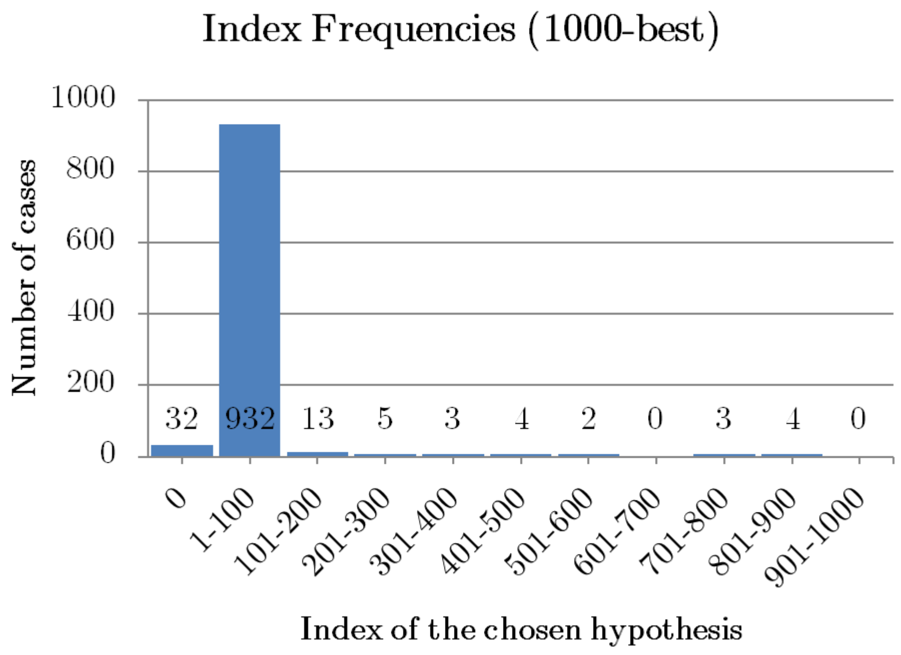


Figure 5.5: Frequency histograms of the indexes of the hypotheses with 1000-best decoding.

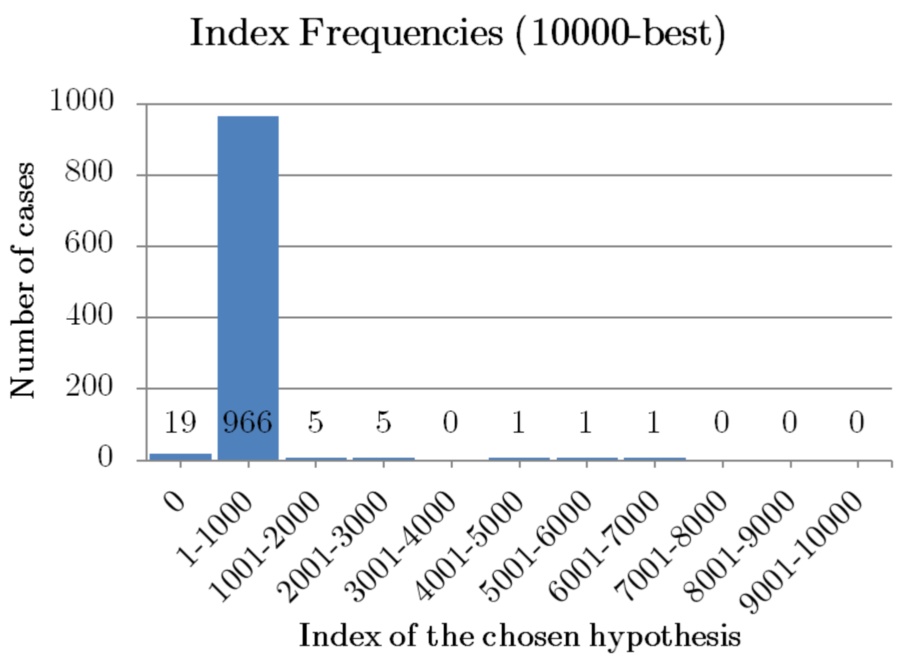


Figure 5.6: Frequency histograms of the indexes of the hypotheses with 10000-best decoding.

CHAPTER 6
Conclusions

In this chapter we summarize the information contained in each of the previous chapters of this document. We also perform a revision on the objectives that we listed, judging if they have been met and to which degree. Lastly, we propose some possible ways to expand the presented work.

6.1 Summary by chapters

In the first chapter we have presented the reasons that have prompted us to develop a combined system for the simultaneous HTR and NER task. Morever, we have motivated the choice for the corpus with which we have worked. In this chapter we have also listed the different objectives that we wanted to accomplish.

Afterwards, we have contextualized our work by presenting the current approaches being used in HTR and NER, as well as the previous work done in the combined HTR and NER task with the chosen corpus. With this chapter we have given some reasoning as to why we have chosen the combined model as the starting approach.

In the third chapter, we have briefly described the theoretical foundation on which the proposed solution is built. In this chapter we have also presented some of our contributions: the ECER and EWER metrics and two simple correction strategies.

In the fourth chapter we have presented with more detail the chosen corpus and described the experimental setup with which we have worked. Following such description, we have shown the results that were obtained with the experimentation.

Finally, in the fifth chapter we have analyzed the results and presented some of the main takeaways that arise from such discussion. We also mentioned how the error correction strategies could be seen as a real improvement over the performance of the combined model alone.

6.2 Accomplished objectives

We consider that we have successfully developed a combined model that is capable of performing HTR and NER over the chosen corpus, which was one of the objectives that we wanted to meet. The combined model obtains acceptable results by itself and replicates the architecture employed in previous experimentation.

The performance of such model has been adequately compared with that of the state of the art for the given corpus. Our experimentation shows that the performance of our implementation is slightly worse than the reference model. Then again, we are working on a slightly more complex task by not considering one of the simplifications that was made, as we already mentioned.

Another goal that we set was the design and implementation of a new metric that took into consideration the characteristics of the task. We believe that the proposed ECER and EWER metrics are valuable metrics to be employed in the evaluation of combined HTR and NER tasks, given their definition. Thus, we believe that this objective has been completed as well.

The last objective that we set was the improvement of the initial model with regard to the proposed metrics by using additional techniques which did not alter the structure of the system. The proposed stack error correction strategy proved to be unsuccessful for the reasons that we mentioned in chapter 5. The *n*-best decoding, on the other hand, improved the performance of the system by adding syntactical constraints in the decoding process. We, however, cannot state that the mentioned objective has been met as the usage of the *n*-best decoding technique introduces additional processes to the pipeline that is used to produce the system output.

6.3 Future work

There are many ways in which to expand the presented work. One of them would be to perform a more exhaustive experimentation, trying out additional configurations of the system in order to improve the results. The number of layers in the convolutional part of the neural network, as well as the number of filters in each layer, are different parameters of the optical model that could be varied. For the language model, we could increment the number of symbols that are considered as part of the context. Regarding the *n*-best decoding technique, a more exhaustive search for the local optimum may be conducted with values of *n* ranging from 1000 to 2500.

Another possible approach to improve the presented work is to forgo the simplification that was assumed regarding continued Named Entities. In short, the simplification consisted of splitting the Named Entities which spanned over different lines. If we opted to, instead, keep the tagging as in the original problem, we may need to switch the focus from line-level decoding to paragraph-level to obtain acceptable results. To this end, we could consider the technique presented in [46], which was mentioned in Chapter 2.

Another extension of the work would be to employ the presented architecture in other corpora to measure the robustness of the system. This could add the problem of having additional types of Entities for which we would have to train the combined model. In any case, it could be useful to test the performance of the *n*-best decoding in similar tasks to see if the results match the tendencies that we spotted during our experimentation.

The application of the combined model as the automatic transcriber and tagger in the construction of a linked collection of documents would be a far more ambitious project. However, before being able to utilize the combined model, it would be necessary to formally describe how such collection would operate and to implement such concept.

Bibliography

[1] Albawi, S., T.A. Mohammed, and S. Al-Zawi: *Understanding of a convolutional neural network*. In *2017 International Conference on Engineering and Technology (ICET)*, pp. 1–6, 2017.

[2] Andrés Moreno, J.: *Approximate search for textual information in images of historical manuscripts using simple regular expression queries*. Degree's thesis, Universitat Politècnica de València, 2020.

[3] Ang, J., Y. Liu, and E. Shriberg: *Automatic dialog act segmentation and classification in multiparty meetings*. In *Proceedings.(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, vol. 1, pp. I–1061. IEEE, 2005.

[4] Bengio, Y., R. Ducharme, P. Vincent, and C. Janvin: *A neural probabilistic language model*. The journal of machine learning research, 3:1137–1155, 2003.

[5] Bluche, T.: *Deep Neural Networks for Large Vocabulary Handwritten Text Recognition*. PhD thesis, Université Paris Sud-Paris XI, 2015.

[6] Boroş, E., V. Romero, M. Maarand, K. Zenklová, J. Křečková, E. Vidal, D. Stutzmann, and C. Kermorvant: *A comparison of sequential and combined approaches for named entity recognition in a corpus of handwritten medieval charters*. In *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp. 79–84. IEEE, 2020.

[7] Cai, J., L. Peng, Y. Tang, C. Liu, and P. Li: *Th-gan: Generative adversarial network based transfer learning for historical chinese character recognition*. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 178–183. IEEE, 2019.

[8] Carbonell, M., M. Villegas, A. Fornés, and J. Lladós: *Joint recognition of handwritten text and named entities with a neural end-to-end model*. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pp. 399–404. IEEE, 2018.

[9] Clausner, C., A. Antonacopoulos, N. Mcgregor, and D. Wilson-Nunn: *Icfhr 2018 competition on recognition of historical arabic scientific manuscripts – rasm2018*. In *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp. 471–476, 2018.

[10] Cortes, C. and V. Vapnik: *Support vector machine*. Machine learning, 20(3):273–297, 1995.

[11] Devlin, J., M.W. Chang, K. Lee, and K. Toutanova: *Bert: Pre-training of deep bidirectional transformers for language understanding*. arXiv preprint arXiv:1810.04805, 2018.

[12] Dyk, D.A. van and X.L. Meng: *The art of data augmentation*. Journal of Computational and Graphical Statistics, 10(1):1–50, 2001. https://doi.org/10.1198/10618600152418584.

[13] Forney, G.: *The viterbi algorithm*. Proceedings of the IEEE, 61(3):268–278, 1973.

[14] Gardner, M. and S. Dorling: *Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences*. Atmospheric Environment, 32(14):2627–2636, 1998, ISSN 1352-2310. https://www.sciencedirect.com/science/article/pii/S1352231097004470.

[15] Ghannay, S., A. Caubrière, Y. Estève, N. Camelin, E. Simonnet, A. Laurent, and E. Morin: *End-to-end named entity and semantic concept extraction from speech*. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pp. 692–699, 2018.

[16] Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio: *Generative adversarial networks*. Communications of the ACM, 63(11):139–144, 2020.

[17] Graves, A., S. Fernández, F. Gomez, and J. Schmidhuber: *Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks*. In *Proceedings of the 23rd international conference on Machine learning*, pp. 369–376, 2006.

[18] He, K., X. Zhang, S. Ren, and J. Sun: *Deep residual learning for image recognition*. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[19] Hu, J., J. Gauthier, P. Qian, E. Wilcox, and R.P. Levy: *A systematic assessment of syntactic generalization in neural language models*. arXiv preprint arXiv:2005.03692, 2020.

[20] Ingle, R.R., Y. Fujii, T. Deselaers, J. Baccash, and A.C. Popat: *A scalable handwritten text recognition system*. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 17–24, 2019.

[21] Jurafsky, D. and J.H. Martin: *Speech and Language Processing (Draft)*. Available at: https://web.stanford.edu/~jurafsky/slp3/, 2020.

[22] Kassis, M. and J. El-Sana: *Learning free line detection in manuscripts using distance transform graph*. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 222–227. IEEE, 2019.

[23] Keret, S., L. Wolf, N. Dershowitz, E. Werner, O. Almogi, and D. Wangchuk: *Transductive learning for reading handwritten tibetan manuscripts*. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 214–221. IEEE, 2019.

[24] Kim, G., V. Govindaraju, and S.N. Srihari: *An architecture for handwritten text recognition systems*. International Journal on Document Analysis and Recognition, 2(1):37–44, 1999.

[25] Kiros, R., R. Salakhutdinov, and R. Zemel: *Multimodal neural language models*. In *International conference on machine learning*, pp. 595–603. PMLR, 2014.

[26] LeCun, Y., B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel: *Backpropagation applied to handwritten zip code recognition*. Neural Computation, 1(4):541–551, 1989.

[27] Li, J., A. Sun, and S.R. Joty: *Segbot: A generic neural text segmentation model with pointer network*. In *IJCAI*, pp. 4166–4172, 2018.

[28] Lowerre, B.T.: *The harpy speech recognition system*. Carnegie Mellon University, 1976.

[29] Marti, U.V. and H. Bunke: *The iam-database: an english sentence database for offline handwriting recognition*. International Journal on Document Analysis and Recognition, 5(1):39–46, 2002.

[30] Mikheev, A.: *A knowledge-free method for capitalized word disambiguation*. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pp. 159–166, 1999.

[31] Mocholí Calvo, C.: *Development and experimentation of a deep learning system for convolutional and recurrent neural networks*. Degree's thesis, Universitat Politècnica de València, 2018.

[32] Mohit, B.: *Named entity recognition*. In *Natural language processing of semitic languages*, pp. 221–245. Springer, 2014.

[33] Nadeau, D. and S. Sekine: *A survey of named entity recognition and classification*. Lingvisticae Investigationes, 30(1):3–26, 2007.

[34] Namboodiri, A.M. and A.K. Jain: *Online handwritten script recognition*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(1):124–130, 2004.

[35] Pan, S.J. and Q. Yang: *A survey on transfer learning*. IEEE Transactions on Knowledge and Data Engineering, 22(10):1345–1359, 2010.

[36] Povey, D., A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, *et al.*: *The kaldi speech recognition toolkit*. In *IEEE 2011 workshop on automatic speech recognition and understanding*, no. CFP11SRW-USB. IEEE Signal Processing Society, 2011.

[37] Puigcerver, J.: *Are multidimensional recurrent layers really necessary for handwritten text recognition?* In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 1, pp. 67–72. IEEE, 2017.

[38] Quinlan, J.: *Decision trees and decision-making*. IEEE Transactions on Systems, Man, and Cybernetics, 20(2):339–346, 1990.

[39] Rabiner, L. and B. Juang: *An introduction to hidden markov models*. IEEE ASSP Magazine, 3(1):4–16, 1986.

[40] Radford, A., K. Narasimhan, T. Salimans, and I. Sutskever: *Improving language understanding by generative pre-training*. Open AI, 2018.

[41] Rajnoha, M., R. Burget, and M.K. Dutta: *Offline handwritten text recognition using support vector machines*. In *2017 4th International Conference on Signal Processing and Integrated Networks (SPIN)*, pp. 132–136. IEEE, 2017.

[42] Romero, V., J.A. Sánchez, V. Bosch, K. Depuydt, and J. de Does: *Influence of text line segmentation in handwritten text recognition*. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 536–540, 2015.

[43] Sánchez, J.A., V. Bosch, V. Romero, K. Depuydt, and J. De Does: *Handwritten text recognition for historical documents in the transcriptorium project*. In *Proceedings of the first international conference on digital access to textual cultural heritage*, pp. 111–117, 2014.

[44] Sekine, S. and C. Nobata: *Definition, dictionaries and tagger for extended named entity hierarchy.* In *LREC*, pp. 1977–1980. Lisbon, Portugal, 2004.

[45] Shen, Y., H. Yun, Z. Lipton, Y. Kronrod, and A. Anandkumar: *Deep active learning for named entity recognition*. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pp. 252–256, Vancouver, Canada, Aug. 2017. Association for Computational Linguistics. https://aclanthology.org/W17-2630.

[46] Singh, S.S. and S. Karayev: *Full page handwriting recognition via image to sequence extraction*. arXiv preprint arXiv:2103.06450, 2021.

[47] Stolcke, A.: *Srilm - an extensible language modeling toolkit*. In *Proc. 7th International Conference on Spoken Language Processing (ICSLP 2002)*, pp. 901–904, 2002.

[48] Sutton, C. and A. McCallum: *An introduction to conditional random fields*. Foundations and Trends® in Machine Learning, 4(4):267–373, 2012, ISSN 1935-8237. http://dx.doi.org/10.1561/2200000013.

[49] Villegas, M., V. Romero, and J.A. Sánchez: *On the modification of binarization algorithms to retain grayscale information for handwritten text recognition*. In *Iberian conference on pattern recognition and image analysis*, pp. 208–215. Springer, 2015.

[50] Walker, J., Y. Fujii, and A.C. Popat: *A web-based ocr service for documents*. In *Proceedings of the 13th IAPR International Workshop on Document Analysis Systems (DAS), Vienna, Austria*, vol. 1, pp. 21–22, 2018.

[51] Wang, D., X. Wang, and S. Lv: *An overview of end-to-end automatic speech recognition*. Symmetry, 11(8):1018, 2019.

[52] Wang, Q., B. Li, T. Xiao, J. Zhu, C. Li, D.F. Wong, and L.S. Chao: *Learning deep transformer models for machine translation*. arXiv preprint arXiv:1906.01787, 2019.

[53] Werbos, P.: *Backpropagation through time: what it does and how to do it*. Proceedings of the IEEE, 78(10):1550–1560, 1990.

[54] Xingjian, S., Z. Chen, H. Wang, D.Y. Yeung, W.K. Wong, and W.c. Woo: *Convolutional lstm network: A machine learning approach for precipitation nowcasting*. In *Advances in neural information processing systems*, pp. 802–810, 2015.

[55] Yujian, L. and L. Bo: *A normalized levenshtein distance metric*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 29(6):1091–1095, 2007.

[56] Zhang, S. and N. Elhadad: *Unsupervised biomedical named entity recognition: Experiments with clinical and biological texts*. Journal of biomedical informatics, 46(6):1088–1098, 2013.

[57] Zhou, G. and J. Su: *Named entity recognition using an hmm-based chunk tagger*. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 473–480, 2002.