University of Brasilia

Institute of Exact Sciences
Department of Computer Science

Deep Active Learning Approaches to the task of Named Entity Recognition

José Reinaldo da Cunha S. A. V. da Silva Neto

Document presented as a requisite for the completion
of the master's program in informatics

Supervisor
Prof. Dr. Thiago de Paulo Faleiros

Brasilia
2021

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

dd

da Cunha S. A. V. da Silva Neto, Jose Reinaldo
Deep Active Learning Approaches to the task of Named Entity Recognition / Jose Reinaldo da Cunha S. A. V. da Silva Neto; orientador Thiago de Paulo Faleiros. -- Brasília, 2021.
65 p.

Dissertação (Mestrado - Mestrado em Informática) -- Universidade de Brasília, 2021.

1. Deep active learning. 2. Self-learning. 3. Named entity recognition. 4. Deep learning. I. de Paulo Faleiros, Thiago, orient. II. Título.

University of Brasilia

Institute of Exact Sciences
Department of Computer Science

Deep Active Learning Approaches to the task of Named Entity Recognition

José Reinaldo da Cunha S. A. V. da Silva Neto

Document presented as a requisite for the qualification
to the master's program in informatics

Prof. Dr. Thiago de Paulo Faleiros (Supervisor)
CIC/UnB

Prof. Dr. Vinicius Ruela Pereira Borges
CIC/UnB

Dr. Ricardo Marcondes Marcacini
ICMC/USP

Prof. Dr. Genaina Nunes Rodrigues
Informatics Graduate Program's Coordinator

November 18, 2021

Acknowledgements

Throughout my master's research period, I've been helped quite often. In this section I hope to extend my feelings of gratitude to all those who made this dissertation possible. In particular, I'd like to thank my supervisor Professor Thiago de Paulo Faleiros, who guided me throughout my master's research, always encouraging and helping me. At the times I found myself unsure of which path to take, our discussions were always reassuring and your takes were always helpful. I'd also like to thank the members of the examination board, Professor Vinicius Borges and Professor Ricardo Marcacini, for revising and providing valuable advice on how to improve my thesis.

During my master's research, I had the great opportunity of joining project KnEDLe at the University of Brasilia. I believe this decision greatly and positively impacted my final thesis. Therefore, I'd like to thank all my colleagues and professors from project KnEDLe, from whom I learned many valuable lessons during our weekly interactions. In particular, I'd like to thank professor Teofilo de campos for being part of the examination committee on my master's qualification exam, and providing insightful advice on how to improve my research.

I also extend my thanks to my family and friends that helped me maintain the balance between my academic and personal lives. In special, I thank my parents Marcos Claudio and Joana Angelica, as well as my sister Marina for being my emotional anchors during the tough times. I also thank my grandparents Jose Reynaldo and Evani, for always being supportive of my research path.

This thesis was partly financed by a research scholarship granted by FAP-DF (*Fundação de Apoio a Pesquisa do Distrito Federal*), through project KnEDLe (*Knowledge Extraction from Documents of Legal content*).

This work was also supported by the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior* - Brazil (CAPES), by granting the author access to the CAPES' *Portal de Periódicos*.

Resumo expandido

Título: Soluções de aprendizagem ativa para a tarefa de reconhecimento de entidades nomeadas

Introdução

Redes neurais profundas são o atual estado da arte para uma grande variedade de desafios em áreas como processamento de linguagem natural e visão computacional, mas necessitam de uma grande quantidade de dados rotulados para serem treinadas para atingir tais resultados. Algoritmos de aprendizagem ativa baseados em redes neurais profundas foram projetados para reduzir a quantidade de dados rotulados que são necessários para treinar estes modelos. Nesta dissertação, nós investigamos a literatura de aprendizagem ativa, buscando pontos a serem trabalhados.

Da nossa investigação da literatura, identificamos que os trabalhos atuais utilizam conjuntos de validação para a realização de *early stopping* do treinamento do modelo durante a execução do algoritmo de aprendizagem ativa. Em cenários onde possuimos poucos dados rotulados, especialmente no começo da execução do algortimo de aprendizagem ativa, não é desejável utilizar dados rotulados para a criação de um conjunto de validação que não será efetivamente utilizado para o treinamento do modelo. Desta forma, um dos objetivos deste trabalho é apresentar uma possível solução para substituir a técnica de *early stopping* com conjunto de validação.

Uma segunda motivação para este trabalho é reduzir o custo de anotação manual de dados durante o algoritmo de aprendizagem ativa. Para isto, iremos investigar a possibilidade de utilizar o modelo treinado para realizar rotulação automática de alguns dados não rotulados. Trabalhos atuais da literatura propuseram soluções de rotulação automática a nível de sentenças, onde sentenças completas são selecionadas para serem rotuladas pelo modelo. Nesta dissertação iremos avaliar também a auto rotulação a nível de palavras, que permite que o modelo e o humano rotulem palavras de uma mesma sentença.

Dadas as motivações apresentadas, propusemos 4 hipóteses de pesquisa como possíveis soluções. A primeira hipótese propõe uma estratégia de *early stopping* que não utiliza o

conjunto de validação. A segunda e terceira hipóteses são relacionadas à investigação da rotulação automática a nível de sentenças. A quarta hipótese é relacionada à investigação da rotulação automática a nível de palavras. As hipóteses propostas são:

1. Nós propomos a estratégia DUTE para a realização de *early stopping* sem a utilização de um conjunto de validação. Nós esperamos que a estratégia DUTE proposta seja competitiva com técnicas tradicionais de *early stopping* mas sem utilizar dados de validação.

2. Nós propomos um algoritmo de aprendizagem ativa com auto rotulação a nível de sentenças que é robusto à escolha do conjunto inicial de dados rotulados. Nós hipotetizamos que nosso algoritmo proposto terá um desempenho superior ao da literatura, tanto em desempenho do modelo treinado quanto em qualidade dos dados rotulados automaticamente.

3. Nós propomos substituir a técnica de auto rotulação tradicional, por técnicas de *self-training* mais sofisticadas da literatura semi supervisionada. Nós hipotetizamos que a utilização de técnicas mais sofisticadas de *self-training* irá melhorar o desempenho do modelo treinado.

4. Nós propomos a utilização de auto rotulação somente nas palavras para as quais o modelo possui grande confiança nas suas predições, ao invés de rotular sentenças completas. Nós esperamos que ao identificarmos palavras que podem ser rotuladas pelo modelo de forma segura em uma sentença selecionada para o humano anotar, é possível reduzir de forma significativa o custo de anotação manual do algoritmo de aprendizagem ativa.

Nós propomos um experimento para cada hipótese. Os quatro experimentos são descritos a seguir.

Experimento 1

O primeiro experimento compara o impacto de diferentes técnicas de *early stopping* em um algoritmo de aprendizagem ativa baseada em redes neurais. Nós comparamos técnicas de *early stopping* tradicionais baseadas em métricas que utilizam o conjunto de validação (e.g. f1-score, *loss*) e a técnica *batch gradient disparity* proposta na literatura com a nossa estratégia DUTE. Dos resultados apresentados, identificamos que nossa técnica utiliza mais épocas de treinamento quando comparada às técnicas tradicionais. No entanto, a estratégia DUTE possui melhor desempenho quando comparada à técnica *batch gradient disparity*, que também não utiliza dados de validação. Desta forma, demonstramos que a

nossa estratégia proposta pode ser utilizada em cenários de poucos recursos onde dados rotulados são escassos e a criação de um conjunto de validação é indesejável.

Experimento 2

No segundo experimento, nós propomos um algoritmo de aprendizagem ativa com rotulação automática a nível de sentenças que é robusto à escolha do conjunto inicial de dados rotulados. Nosso algoritmo possui duas diferenças significativas, quando comparado ao algoritmo da literatura. A primeira diferença é que os dados rotulados pelo humano são separados dos dados rotulados pelo modelo. Isto nos permite dar um peso menor para os dados rotulados automaticamente durante o treinamento do modelo, pois estes podem ser ruidosos. A segunda diferença é que os dados rotulados de forma automática são devolvidos ao conjunto de dados não rotulados após o treinamento do modelo, permitindo a reanotação destes dados. O experimento 2 consiste, então, na comparação entre o algoritmo da literatura e o nosso algoritmo proposto, ambos com auto rotulação a nível de sentenças. Para demonstrar a sensibilidade do algoritmo da literatura ao conjunto inicial de dados rotulados, nós esperamos que uma porcentagem do conjunto de treinamento seja rotulado de forma manual antes de permitir a auto rotulação pelo modelo. Nós realizamos testes com a auto rotulação iniciando com 1%, 5%, 10% e 15% do conjunto de treinamento rotulado. Os resultados do experimento mostraram que tanto o desempenho do modelo final quanto a qualidade dos dados rotulados automaticamente crescem de acordo com o tamanho do conjunto inicial de dados rotulados manualmente. Também observamos que o nosso algoritmo proposto é robusto à escolha do conjunto inicial de dados rotulados. Ele é capaz de treinar um modelo com desempenho superior aos modelos treinados pelo algoritmo da literatura, e de rotular menos dados de forma incorreta.

Experimento 3

No terceiro experimento, nós investigamos o impacto de diferentes técnicas de *self-training* no nosso algoritmo proposto no experimento 2. Nós avaliamos três técnicas de *self-training* da literatura semi-supervisionada, sendo elas: (1) *cross-view training*[14], (2) *virtual adversarial training*[40], e (3) *word dropout*[14]. Dos resultados obtidos, nós pudemos observar que nenhuma das técnicas obteve resultados consistentemente superiores à *baseline* que é o algoritmo de aprendizagem ativa sem *self-training*. Algumas técnicas como a *cross-view training* e a *virtual adversarial training* obtém resultados melhores em iterações iniciais do algoritmo quando comparadas à *baseline*, mas acabam obtendo resultados piores nas iterações finais.

Experimento 4

O quarto experimento investiga a possibilidade de realizar a auto-rotulação a nível de palavras. A auto-rotulação a nível de sentenças, utilizada nos experimentos anteriores, identificava sentenças não rotuladas que poderiam ser completamente anotadas pelo modelo de forma confiável. Neste experimento, nós iremos identificar as palavras, dentro das sentenças selecionadas para rotulação manual, que podem ser rotuladas pelo modelo de forma segura. Desta forma, o humano não precisa rotular todas as palavras das sentenças selecionadas pelo algoritmo de aprendizagem ativa, pois algumas das palavras serão rotuladas de forma automática. A *baseline* para comparação será o algoritmo de aprendizagem ativa sem rotulação automática. Os resultados do experimento 4 demonstraram que a solução de auto rotulação a nível de palavras foi capaz de treinar um modelo com desempenho similar ao treinado pela *baseline* mas com uma redução significativa na quantidade de dados rotulados manualmente. Mais especificamente, para os datasets CoNLL2003, OntoNotes5.0 e Aposentadoria, a redução foi de $29,24\%$, $14,37\%$, e $3,95\%$, respectivamente.

Conclusão

Dos quatro experimentos realizados percebemos que a estratégia DUTE é uma solução viável para substituir técnicas de *early stopping* em algoritmo de aprendizagem ativa. Das desvantagens desta estratégia, podemos citar que ela não é capaz de identificar *overfitting* do modelo, uma vez que ela foi projetada para acelerar a simulação do algoritmo de aprendizagem ativa. Desta forma, a definição dos parâmetros do modelo neural e do treinamento supervisionado (e.g. épocas de treinamento máximo) devem ser escolhidos de forma cautelosa.

O segundo experimento mostrou que nosso algoritmo de aprendizagem ativa com rotulação automática a nível de sentenças é mais robusto à escolha do conjunto inicial de dados rotulados, quando comparado ao algoritmo da literatura. Ao contrário do esperado, nosso algoritmo proposto não é capaz de melhorar significadamente o desempenho do modelo com menos dados rotulados, como mostrado no Experimento 3. Mesmo técnicas mais sofisticadas de *self-training*, não foram capazes de melhorar o desempenho do modelo treinado ao utilizar os dados não rotulados.

O quarto experimento, no entanto, nos mostra que é possível utilizar rotulação automática a nível de palavras para reduzir de forma significativa o custo de anotação manual. O algoritmo proposto foi capaz de treinar um modelo neural ao seu pico de desempenho utilizando até $29,24\%$ menos dados rotulados manualmente quando comparado ao algoritmo de aprendizagem ativa sem auto rotulação.

Trabalhos futuros

Grande parte dos trabalhos atuais da literatura em aprendizagem ativa estudam funções de sampling, estratégias para selecionar os dados mais interessantes do conjunto de dados não rotulados. Estes trabalhos focam em acelerar a convergência dos algoritmos de aprendizagem ativa, treinando modelos ao seu pico de desempenho com a menor quantidade de dados rotulados possível. No entanto, algoritmos de aprendizagem ativa possuem uma série de questões práticas de implementação ainda não resolvidas. Um dos problemas mais sérios é a seleção dos hyperparâmetros do modelo e do treinamento supervisionado. No início do algoritmo de aprendizagem ativa normalmente não há dados de validação para identificar estes parâmetros. Desta forma, áreas de pesquisa como autoML e tuning de parâmetros de forma não supervisionada estão fortemente relacionadas à implementação de algoritmos de aprendizagem ativa em cenários reais.

Outra direção de pesquisa é a busca por métricas capazes de identificar overfitting do modelo, sem a utilização de dados de validação. Desta forma seria possível realizar o *early stopping* do treinamento do modelo de forma confiável, sem a necessidade de um conjunto de validação.

Podem ser realizados, também, outros experimentos com auto rotulação a nível de palavras. Uma possibilidade é estender a técnica de refinamento de predições para uma versão iterativa, capaz de reduzir a quantidade de tokens incorretos.

Palavras-chave: Aprendizagem ativa, Auto-aprendizagem, classificação sequencial, Redes neurais profundas, Reconhecimento de entidades nomeadas

Abstract

Deep neural networks are the current state-of-the-art for a variety of challenging tasks in fields such as natural language processing and computer vision, but they rely on big labeled datasets to be trained to achieve such results. Deep active learning algorithms have been designed to reduce the amount of labeled data to train these models. This dissertation identifies shortcomings of the current works from the literature on deep active learning algorithms applied to the task of named entity recognition, and proposes potential solutions to them. In particular, current works from the literature rely on validation sets to apply early stopping of the model training during the active learning process. In low resource scenarios, however, separating labeled samples in order to create a validation set is undesirable. Therefore, we propose the *Dynamic Update of Training Epochs (DUTE)* strategy that acts as an unsupervised early stopping technique. Experimental results suggest that the proposed DUTE strategy is capable of maintaining the trained model's performance, when compared to traditional early stopping techniques, while not relying on validation sets. We also investigate self-labeling as a viable option to further reduce the annotation costs in active learning scenarios. In particular, we experiment with sentence-level and token-level self-labeling strategies. It was observed that despite significant efforts, sentence-level self-labeling did not incur a significant improvement over previous works from the literature. However, token-level self-labeling has shown promising results by training models that achieve similar performance to the current state-of-the-art works on deep active learning from the literature while requiring significantly less hand annotated data. More specifically, experiments performed on the CoNLL2003 dataset have shown that the proposed token-level self-labeling strategy trained a neural model to near peak performance using 29.24% less hand annotated data.

Keywords: Deep active learning, Self-learning, Sequence tagging, Deep neural networks, Natural language processing, named entity recognition

Contents

List of Figures

List of Tables

Acronyms

AL Active Learning.

ASL Active-Self Learning.

AT Adversarial Training.

BGD Batch Gradient Disparity.

CRF conditional Random Field.

CVT Cross-View Training.

DAL Deep Active Learning.

DASL Deep Active-Self Learning.

DUTE Dynamic Update of Training Epochs.

ES Early Stopping.

LC Least Confidence.

LSTM Long Short-Term Memory.

MLM Masked Language Model.

MNLP Maximum Normalized Log-Probability.

NER Named Entity Recognition.

NLP Natural Language Processing.

SC Stopping Criterion/Criteria.

SOTA State of the Art.

SVM Support Vector Machine.

VAT Virtual Adversarial Training.

Chapter 1

Introduction

There are many tasks in the natural language processing (NLP) domain that can be modeled as problems of sequence tagging or sequence labeling. The most well-known examples are the tasks of named entity recognition (NER)[49] and part-of-speech (POS) tagging[53]. The task of joint text segmentation and segment labeling can also be modeled as a sequence tagging problem[6], although it appears less often in the literature. The task of named entity recognition will be the focus of this dissertation.

The NER task is widely used for information extraction on natural language texts, with trained machine learning models being able to directly extract named entities from an unstructured text. The extracted information can be used to create databases of structured information, as well as help in solving more complex NLP challenges, such as text summarization and relationship extraction.

Deep neural-based models have been the state-of-the-art (SOTA) for solving the task of NER [14, 4, 6]. But to be trained to achieve the state-of-the-art , they require a significant amount of labeled data. In many practical scenarios it is infeasible for huge amounts of data to be manually labeled. One solution to this problem is the use of active learning (AL) algorithms, that enable us to identify a subset of data samples that reliably represents the label distribution of the entire dataset. The main motivation behind active learning is that a model trained on the entire subset and another trained on this smaller representative subset will achieve similar performance. Thus, annotation costs are reduced to only the small subset of representative data samples.

AL research for sequence tagging tasks are not recent[55], but Deep Active Learning (DAL) applied to them started to appear much more recently[57]. As such, they still have problems that need to be solved in order for them to be reliable in real-world scenarios, some of which are presented in the next section.

1.1 Motivation

Even though neural-based models achieve state-of-the-art performance for NER and active learning algorithms have been shown to work reliably for various tasks, DAL brings with it new challenges to be solved. Two of these problems are evident in the literature as presented in section 3.1.1: (1) Validation sets are used for tuning hyperparameters for the neural model and the optimization method (e.g. learning rate); and (2) current DAL algorithms for NER rely on validation sets for early stopping of the supervised training.

Besides the drawbacks of current works on DAL from the literature, another aspect to be investigated in this dissertation is whether self-learning can be used to enhance the performance of active learning algorithms. The self-learning may be implemented by using the trained model to annotate all tokens in an unlabeled sentence (i.e. sentence-level self-labeling) or to annotate specific tokens only (i.e. token-level self-labeling). Current works on the literature on semi-supervised and active-self learning algorithms rely on sentence-level self-labeling, as presented in section 3. This dissertation will, however, investigate both methods of self-labeling applied in deep active learning scenarios.

The main motivation of this dissertation is to address the problem that current DAL algorithms rely on validation sets, and investigate the possibility of using self-labeling to reduce the human annotation efforts for deep active learning algorithms.

1.2 Objectives

Section 1.1 presented two main problems that deep active learning algorithms may face, and proposed an investigation of self-learning to enhance the performance of models trained in DAL scenarios. In this dissertation, we will address one of the drawbacks of current DAL algorithms, namely the one related to the use of validation sets for early stopping. We will also investigate the possibility of using self-labeling in order to enhance the performance of DAL algorithms. The remaining drawback, namely the lack of prior knowledge about the optimal structure and hyperparameters of the neural model, will be left for future works. Thus, the questions to be answered in this dissertation are:

1. How to avoid using validation sets throughout the DAL procedure?
2. Can sentence-level self-labeling be used to enhance the performance of DAL algorithms?
3. Can token-level self-labeling be used to enhance the performance of DAL algorithms?

In order to address these questions, we hypothesize on potential solutions which we present in the next Section 1.3.

1.3 Hypotheses

In order to address the problems presented in section 1.2, we propose four hypotheses. The first hypothesis is related to the problem of current DAL algorithms relying on validation sets for early stopping. Hypotheses 2 and 3 are related to the investigation of sentence-level self-labeling. And hypothesis 4 is related to the investigation of token-level self-labeling. They are formally defined as follows:

1. An early stopping strategy to dynamically change the number of training epochs at each active learning iteration based on the model's confidence on the unlabeled set can replace traditional early stopping techniques that rely on validation sets.
2. It is possible to design a sentence-level active-self learning algorithm that is less sensitive to the initial set of labeled samples, when compared to previous algorithms from the literature [62]
3. Replacing the traditional self-training technique from the sentence-level active-self learning algorithm with more sophisticated techniques from the semi-supervised literature can improve the performance of the trained model.
4. Token-level self-labeling, instead of sentence-level, can enable a deep active learning algorithm to train a NER model to peak performance with significantly less hand annotated data.

Four experiments are developed in order to demonstrate the validity of each of the proposed hypotheses. These experiments are presented in more details in chapter 4.

1.4 Contributions

The main contributions of this dissertation are as follows:

1. We proposed an early stopping technique for active learning algorithms that does not rely on validation sets, and that is competitive with traditional techniques that do rely on them.
2. We proposed an active-self learning algorithm with sentence-level self-labeling that is less sensitive to the initial set of labeled data, when compared to previous works from the literature.
3. We have shown that more sophisticated self-training techniques do not consistently improve the active-self learning algorithm, as we initially expected.

4. Our proposed token-level active-self learning algorithm achieves the state-of-the-art on three NER datasets, being capable of training a neural model to peak performance using the least amount of human annotation, when compared to previous works from the literature.

Compiling the results achieved by contributions 1 and 2, we published the paper:

- C. S. A. V. da Silva Neto, J. R. and de Paulo Faleiros, T. *Deep Active-Self Learning Applied to Named Entity Recognition* [9]

1.5 Outline of the document

This section presents an introduction to the problem of deep active learning for sequence tagging tasks, along with the motivation and objectives of this research. Chapter 2 introduces the theoretical foundations needed to better understand the experiments to be designed and executed in following sections. Chapter 3 presents works from the literature related to active learning algorithms applied to NER, as well as stopping criteria for AL not restricted to sequence tagging tasks, and semi-supervised techniques applied to NER. Chapter 4 presents the methodology for the experiments, including design choices and evaluation procedures to be used. Chapter 5 shows the results of the experiments, with brief observations about them. Finally, Chapter 6 concludes this dissertation, pointing out the most important results achieved and possible future research directions.

Chapter 2

Background

This chapter introduces the theoretical foundations of named entity recognition tasks, its related machine learning models, and active learning algorithms, that are deemed necessary for a better understanding of the following sections. It starts with an introduction to the NER task, including annotation schemes and performance metrics; followed by commonly used machine learning models for NER; a general review of active learning algorithms, focusing on those designed for pool-based scenarios is presented next; followed by a brief introduction to self-learning algorithms.

2.1 Named Entity Recognition Task

The expression named entity was first used in the sixth message understanding conference (MUC-6) for its named entity recognition task[23]. At that time, three types of entities were to be identified: (1) Names (organizations, persons and locations), (2) times (date, hour) and (3) quantities (monetary values and percentages). Ever since this first challenge, interest in named entity recognition has grown significantly as named entity information has been proven useful for a variety of challenging NLP tasks such as question answering[42, 59], automatic summarization[45] and machine translation[3].

The named entity recognition task is a sequence labeling problem where each word in a sentence must be classified into one of many predefined classes. As such, the NER datasets come with word level annotations, assigning to each word its true label. Next, we present how this annotation is presented for named entities.

2.1.1 Annotation Schemes

Named entity recognition is a classification task where each word in a sentence is assigned to one named entity class. Many times, a named entity is composed of multiple words

and named entities of the same class may be adjacent to one another, making it harder to properly identify them. To address this problem, annotation schemes were designed to clearly define named entity boundaries. Each scheme is formed by a set of characters that adds boundary information to each entity class. The characters used in such schemes are shown in Table 2.1, where the character O marks a word as not belonging to any

Table 2.1: Special characters to define boundaries for named entities

Character	I	O	B	E	S
Meaning	Inside	Outside	Beginning	End	Single

of the desired classes of named entities, the character I indicates that a word is inside a named entity, the characters B and E indicate the first and last word of an entity, respectively, and the character S is used for entities composed of a single word. The annotation schemes may use some or all of these characters to define entity boundaries.

Three common annotation schemes that use the characters presented in Table 2.1 are: IOB1, IOB2 and IOBES. The IOB1 notation uses the {I, O, B} characters and separates adjacent named entities of the same class by marking the first word of the second entity using the letter B. The IOB2 notation uses the same set of characters as the IOB1, but the letter B is used to indicate the first word of all entities, not only those that are adjacent to another. The IOBES is the most complete, and uses the whole set of characters to define entity boundaries. Table 2.2 shows how different annotation schemes are used in a sample sentence extracted from the CoNLL2003 NER dataset, where two named entities of the class *location* (identified as LOC) are adjacent to each other.

Table 2.2: Different annotation schemes for named entity recognition

	Above	summer	rainfall	in	the	U.S.	High	Plains	has	produced
IOB1	O	O	O	O	O	I-LOC	B-LOC	I-LOC	O	O
IOB2	O	O	O	O	O	B-LOC	B-LOC	I-LOC	O	O
IOBES	O	O	O	O	O	S-LOC	B-LOC	E-LOC	O	O

2.1.2 Performance Metrics

Accuracy, precision, recall and f1 score are metrics often used for the evaluation of named entity recognition models. These metrics are presented by equations 2.1, 2.2, 2.3, 2.4.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \tag{2.4}$$

Where TP stands for true positive, TN for true negative, FP for false positive and FN for false negative.

One question that comes up when evaluating NER models is whether to compute these metrics on a word or on an entity level. Many entities are formed by multiple words, and maybe some of them are correctly predicted while the rest is not. The *exact match* concept introduced in the 2002 Conference on natural language learning (CoNLL-2002)[52] states that named entities are correctly classified if both their class and boundaries are predicted accurately. That concept gives birth to *strict* and *relaxed* metrics, where *strict* refers to metrics that use the exact match definition (entity level predictions), while *relaxed* do not use the exact match, allowing for entities to be partially predicted.

Another concern to be addressed is how to calculate the overall metric function for multiple classes. The final metric can be macro or micro averaged. Using a macro average means to compute the metric function for each class separately and averaging the result to obtain the overall metric. On the other hand, micro average computes the metric function for all predictions over all classes at once.

2.2 Machine learning approaches to named entity recognition tasks

Machine learning models for named entity recognition tasks can be separated into two classes: traditional and deep learning models. Traditional models for NER tasks include shallow machine learning models such as logistic regression[31], support vector machines[56], and linear-chain conditional random field[18]. Many works still use these models for specific scenarios, such as in low resource scenarios where active learning algorithms thrive. Deep learning models, on the other hand, are the current state-of-the-art on complex NER datasets where plenty of labeled data is available.

Next, section 2.2.1 presents the linear-chain conditional random field model, which is still frequently used for solving low-resource NER tasks. Section 2.2.2 presents some of the components used to build up deep learning models.

2.2.1 Linear-chain Conditional Random Field

The linear chain conditional random field[61] is a discriminative model often employed in sequence classification tasks (e.g. named entity recognition, part-of-speech tagging). It uses a conditional probability to identify the most likely sequence of tags given a sequence of inputs. The conditional probability of a sequence of tags Y given a sequence of inputs X is modeled as

$$\begin{aligned} P(Y|X) =& \frac{\prod_{i=1}^{L}\Big(exp\Big(e(y_i,x_i)\Big)exp\Big(tr(y_{i-1},y_i)\Big)\Big)}{Z(X)} \\ =& \frac{\prod_{i=1}^{L} exp\Big(e(y_i,x_i)+tr(y_{i-1},y_i)\Big)}{Z(X)} \\ =& \frac{exp\Big(\sum_{i=1}^{L}\Big(e(y_i,x_i)+tr(y_{i-1},y_i)\Big)\Big)}{Z(X)} \end{aligned} \tag{2.5}$$

where $e(x_i, y_i)$ is called the emission score, it represents how likely the input x_i is to be tagged as y_i. $tr(y_{i-1}, y_i)$ is called the transition score, representing how likely it is for the current element to be tagged y_i if the previous element was tagged y_{i-1}, and L represents the length of the input sequence. Finally, $Z(X)$ is called the partition function and it effectively normalizes equation 2.5, ensuring that the resulting value stays in the interval $[0, 1]$. This partition function considers all possible sequences of tags, and can be calculated as

$$Z(X) = \sum_{y'_1}^{C}\sum_{y'_2}^{C}\cdots\sum_{y'_{L-1}}^{C}\sum_{y'_L}^{C} exp\Big(\sum_{i=1}^{L}\Big(e(y'_i,x_i)+tr(y'_{i-1},y'_i)\Big)\Big). \tag{2.6}$$

Where C is the total number of classes. Notice that here we use y'_i instead of y_i so as to differentiate the labels being used to calculate the partition function from the labels of the sequence Y being used to calculate the numerator in Equation 2.5. Finally, the complete equation for the conditional probability is

$$P(Y|X) = \frac{exp\Big(\sum_{i=1}^{L}\Big(e(y_i,x_i)+tr(y_{i-1},y_i)\Big)\Big)}{\sum_{y'_1}^{C}\sum_{y'_2}^{C}\cdots\sum_{y'_{L-1}}^{C}\sum_{y'_L}^{C} exp\Big(\sum_{i=1}^{L}\Big(e(y'_i,x_i)+tr(y'_{i-1},y'_i)\Big)\Big)} \tag{2.7}$$

Efficient computation of the partition function

A naive implementation of Equation 2.6 has exponential time complexity to be computed. Fortunately, the linear-chain restriction of the linear-chain CRF allows us to apply a dynamic programming strategy to efficiently compute the partition function in polynomial time. Thanks to the chain structure of the linear-chain CRF, we can re-write the equation for the partition function by pulling out the emission and transition scores up to the summation they are dependent on:

$$Z(X) = \sum_{y'_L} exp\Big(e(y'_L, x_L)\Big) \Bigg[\sum_{y'_{L-1}} exp\Big(e(y'_{L-1}, x_{L-1}\Big) + tr(y'_L, y'_{L-1}) \cdots \underbrace{\Bigg[\sum_{y'_2} exp\Big(e(y'_2, x_2) + tr(y'_3, y'_2)\Big) \underbrace{\Bigg[\sum_{y'_1} exp\Big(e(y'_1, x_1) + tr(y'_2, y'_1)\Big) \Bigg]}_{\alpha_1(y'_2)} \Bigg]}_{\alpha_2(y'_3)} \cdots \Bigg] \tag{2.8}$$

Now we need to compute all α values. We start by computing $\alpha_1(\cdot)$ for each possible y'_2 label. We then proceed to compute $\alpha_2(\cdot)$ using the values of $\alpha_1(\cdot)$, and repeat this process until we reach to $\alpha_L(\cdot)$. The partition function $Z(X)$ is calculated as the sum over $\alpha_L(\cdot)$ for all possible classes. Figure 2.1 shows a representation of the dynamic programming matrix that is to be filled in order to compute the partition function.

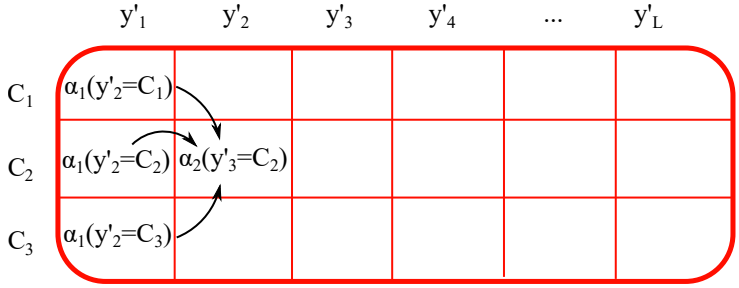

Figure 2.1: Dynamic programming matrix that must be completed in order to calculate the partition function $Z(X)$. Note that C_k represents the k-th label class, and y'_k indicates the label of the k-th element of the sequence.

Inference

Up until now, we have seen how to compute the conditional probability of a sequence of elements X given the correct sequence of tags Y for a linear-chain CRF. Now we wish to understand how to find the most likely sequence of tags $\hat{Y}$, given a trained model and a sequence of elements X. The most used approach is to employ the viterbi

decoding algorithm[61]. The viterbi algorithm for decoding consists of a pair of forward and backward passes. The forward pass identifies the transition that has the highest score for each possible tag for the current element of the sequence when considering all tags of the previous element. The backward pass consists of, given the best transitions computed in the forward pass, identifying the most likely sequence of tags. Figure 2.2 presents diagrams to better explain both the forward and backward passes.

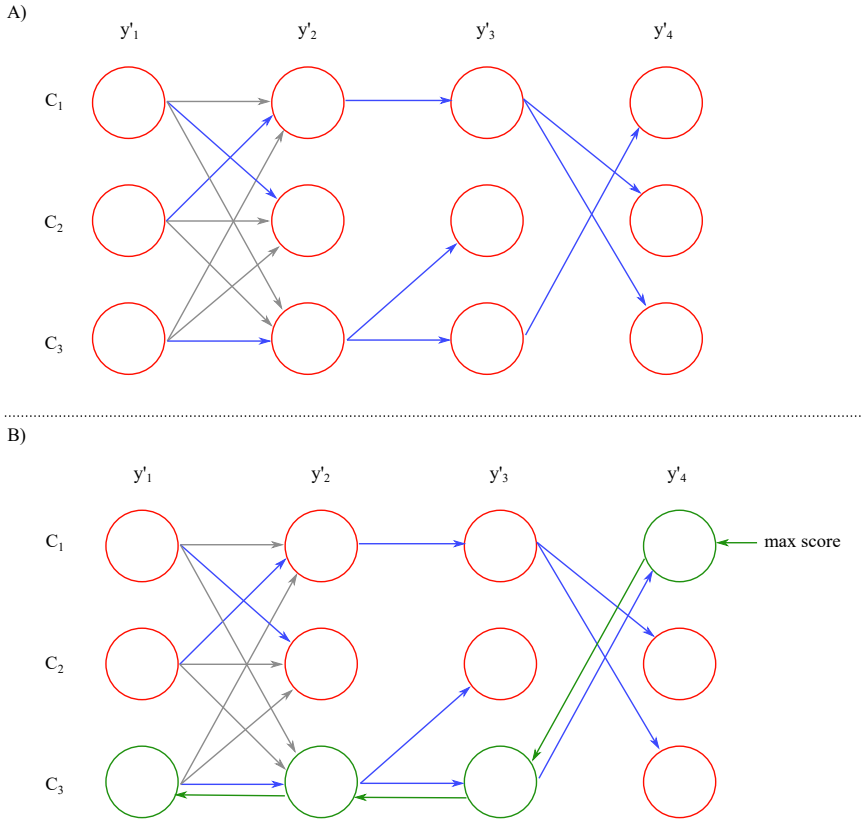

Figure 2.2: Diagram of the viterbi decoding algorithm, for a sequence of four elements and three possible labels for each element. Diagram A represents the forward pass of the viterbi algorithm, where the best transitions from the y'_{k-1} to the y'_k label is computed for each possible class (blue arrows represent the best transitions). Diagram B represents the backward pass, where the best sequence of tags is identified (in green).

2.2.2 Neural models

Neural models are the current state-of-the-art on challenging NER datasets, such as the OntoNotes5.0[49] and the CoNLL03[53]. The main components for such models are presented in the next sections.

Embeddings

Machine learning models working in natural language processing tasks cannot, in general, receive raw text as its input. A preprocessing step is required to transform the text into numeric values that the model can understand. A naive approach to this preprocessing is to one-hot encode each word of your vocabulary. By one-hot encoding the text, each word can now be represented by a sparse vector $\overrightarrow{w} \in \{0,1\}^{|V|}$, where V is the vocabulary set. This naive solution generates a very sparse representation of the input data. To avoid sparse representations, various word embedding techniques were proposed to try and generate more dense and meaningful vector representations for each word. There are two types of embeddings used for NER tasks: (1) non-contextual word embeddings and (2) contextual embeddings.

non-contextual word embeddings refer to techniques where each word has a fixed numeric vector assigned to it. The parameters of this vector are usually learned through an unsupervised pretraining on a large training corpora, and later fine-tuned for a specific downstream task (e.g. NER, POS-tag). Word2Vec[39], FastText[8], and Glove[47] are examples of non-contextual word embedding techniques. These non-contextual word embeddings are able to retain the relationship between words[39], as depicted in Figure 2.3 where it can be seen that there is a clear relationship between the word embeddings of the names of countries and their respective capital cities.

Contextual embeddings are more recent, when compared to non-contextual word embeddings, and their main characteristic is that a word does not have a fixed numeric vector representation. Instead, a word's embedding will depend on its surrounding context in a text. The vector representation for contextual embeddings are obtained from inner states of neural models trained in language modeling tasks. BERT-based[15], ELMo[48] and Flair[1] are examples of contextual word embeddings.

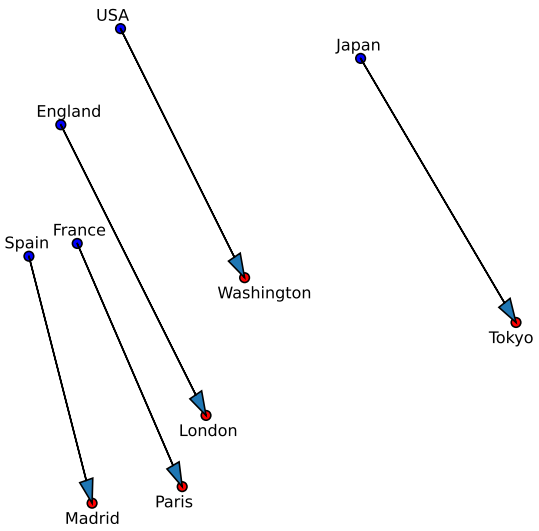

Figure 2.3: Demonstration on the relationship between names of countries and their respective capital cities using word2vec embeddings pretrained on the Googlenews corpora. Principal component analysis was performed to reduce the embedding dimension from 300 to 2, allowing for a 2 dimensional plot.

Convolutional layers

A convolution is defined as an operation between two real-valued functions, where the output represents how one function behaves as another slides across it. Formally, a convolution between continuous functions $f(\cdot)$ and $g(\cdot)$ is given by

$$c(t) = (f * g)(t) = \int f(\tau)g(t-\tau)d\tau. \tag{2.9}$$

This definition can also be applied to discrete functions, where a convolution between the discrete functions $a(\cdot)$ and $b(\cdot)$ is given by

$$c(k) = (a * b)(k) = \sum_i a(i)b(k-i) \tag{2.10}$$

A convolutional layer in deep learning models executes a discrete convolution, where one discrete function is the data fed to the convolutional layer, and the function that slides over the first one is called a filter. In most implementations of convolutional layers in deep learning frameworks, the operation performed is actually the cross correlation, which is similar to the discrete convolution but without the horizontal shift of the sliding

function. The cross correlation between discrete functions $a(\cdot)$ and $b(\cdot)$ is given by

$$cr(k) = \sum_i a(i)b(k+i) \tag{2.11}$$

Generically, both the input and the filter are matrices that can have 1 or more dimensions. Figure 2.4 represents a 2 dimensional convolution.

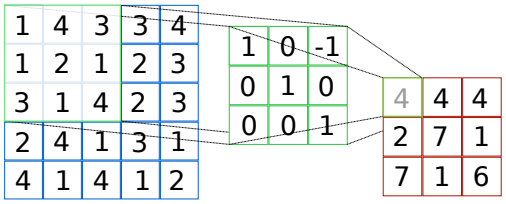

Figure 2.4: Example of a 2 dimensional convolution with an input matrix of size 5×5 and a filter with kernel size of 3×3. In this example, the convolution is performed with stride of 1 and padding equal to 0, generating an output matrix of dimensions 3×3. The input, filter and output matrices are represented in blue, green and red, respectively.

For named entity recognition tasks, convolutional layers are used to generate character level word embeddings, and to encode sentence information from embeddings[57]. Figure 2.5 presents how a character level word embedding can be generated from a 1 dimensional convolution. Note that pooling techniques are generally used to generate fixed size word embeddings.

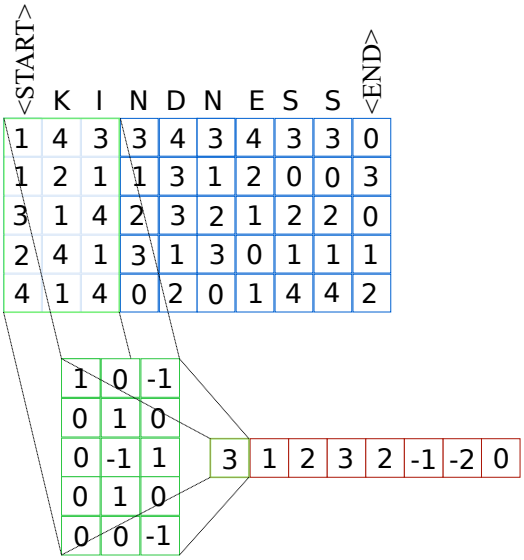

Figure 2.5: Representation of a character level word embedding using a 1 dimensional convolution. <START> and <END> are special characters that indicate the beginning and end of a word. Each character is represented by a numeric vector, and the 1 dimensional convolution takes place as the filter, shown in green, slides from left to right over the character embedding matrix.

Long-short term memory layers

Recurrent neural networks have been the go to method to solve hard problems that involve modeling sequential data. The vanilla recurrent network consists of a feedforward neural network augmented with context layers. These context layers are responsible to create feedback loops in the network, allowing networks to model short range relationships between different elements of a sequence. One issue vanilla recurrent networks encountered were vanishing gradients as sequences became longer, making it difficult to model long range dependencies[27]. To address this issue, Hochreiter and Schmidhuber[27] proposed the Long Short-Term Memory (LSTM) model, that facilitate the flow of the backpropagated gradient through longer sequences, thus mitigating the effects of vanishing gradients. In order to facilitate the flow of the backpropagated gradient, the LSTM cell creates pathways that allow for previous hidden and cell states to be forwarded to the cell's current output generating an effect similar to that of residual connections[25] in deep feedforward networks. The flow of information inside an LSTM cell is controlled by three gated units, namely input, forget and output gates. The input gate is responsible for controlling how much of the current input and previous hidden state are to be used for the computation of the new cell state. The forget gate determines how much of the previous cell state is to be used to calculate new hidden and cell states. The output gate controls how much

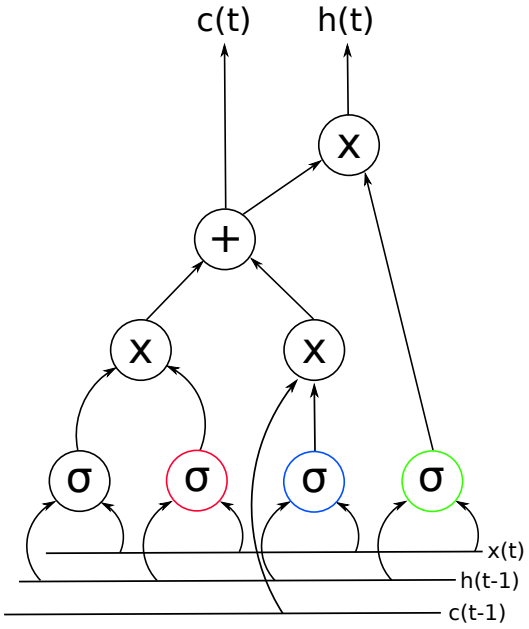

Figure 2.6: Diagram of an LSTM cell. The hidden and cell states at time-step t are represented by $h(t)$ and $c(t)$, respectively. $x(t)$ is the input data at time-step t. The input, forget and output gates are represented in colors red, blue and green, respectively. σ represents a sigmoid function.

of the current cell state is to be used to compute the hidden state. Figure 2.6 presents a diagram of an LSTM cell.

Attention mechanism

Attention in neural models applied to natural language processing was first proposed by Bahdanau et al[5]. But has seen a rise in popularity ever since the work from Vaswani et al[63] proposed the scaled dot-product attention and the transformer network.

Since then, it has become the norm to see attention mechanisms as probabilistic retrieval systems, where the attention module retrieves a value $v \in V$ according to a key $k \in K$ given a query $q \in Q$. In general, a similarity measure is computed between the keys and the query, and a linear combination of the values whose keys are most similar to the query is retrieved. The scaled-dot product attention is one example of attention, and for a set of values $V \in \mathbb{R}^{N \times d_v}$, keys $K \in \mathbb{R}^{N \times d_k}$ and a query $Q \in \mathbb{R}^{d_k \times 1}$, it is computed as

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{2.12}$$

where d_k is the dimension of a key vector, d_v is the dimension of a value vector and N is the number of values and keys. Note that the scaled-dot product attention uses the dot product between the query and the keys to measure similarity, and uses a softmax

function to generate the weights, or importance scores, that are then multiplied by the values related to each key. In other words, Equation 2.12 can be interpreted as assigning an importance score to each value $v_i \in V$ given how close its key vector $k_i \in K$ is to the query vector Q. The softmax function normalizes the importance scores to a value in the interval $[0, 1]$ and ensures that all scores sum up to 1, meaning that the output to the attention module is a weighted averaged sum of the values given their importance scores.

Transformers

The transformer network was first proposed by Vaswani et al[63]. It is an attention-based neural architecture that models sequential data without explicit recurrences and achieves high performance in sequential tasks. The original transformer was proposed for the task of neural translation, and has an encoder-decoder architecture as depicted in Figure 2.7.

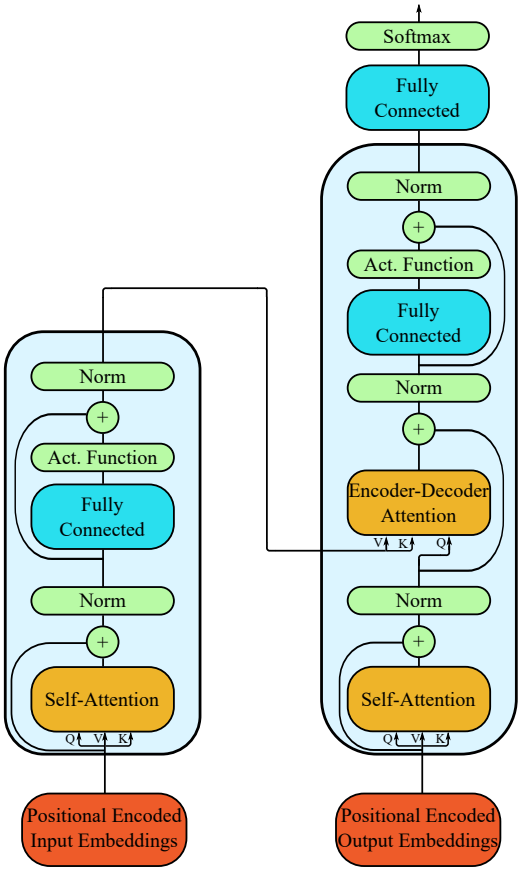

Figure 2.7: Diagram of the transformer network proposed by Vaswani et al[63].

The attention mechanism is a bag-of-words model, that is unable to make use of information about the position of each element in a sequence by itself. Because of that, positional encoding is used to add information about the relative position of an element

in a sequence to its respective embedding. The transformer uses two types of attention mechanisms, namely self-attention and encoder-decoder attention. The self-attention component uses an attention mechanism where the keys, values and queries come from the same sequence, and are able to identify which elements of a sequence relate the most to a specific element from that same sequence. For the encoder-decoder attention, on the other hand, keys and values are generated from the encoder output, while queries come from the output of the decoder self-attention module. The encoder-decoder attention is used to identify which elements from the input sequence are the most relevant for predicting the next output, given prior predicted outputs. Attention mechanisms are inherently linear, and because of that, fully-connected layers with non-linear activation functions are used after attention modules to introduce nonlinearities inside the model.

2.2.3 Early stopping

Large deep neural models have a tendency to overfit during supervised training, as their representational capabilities are sufficient to learn random noise that are specific to the training set. The overfitting makes the trained model to be very proficient in its task when being used on the training set, but poor generalization makes it harder for this model to be used in unseen data. One way to identify when overfitting takes place is to observe the performance (e.g. loss, accuracy, f1-score) of the model on a set of data that was withheld during training, typically called validation or development set. When the performance of the model on this validation set starts to worsen, even though the training loss gradually diminishes, the model is most likely starting to overfit. Early stopping is one solution to this problem, where the training is halted when the model's performance on the validation set hasn't improved over a predefined number of training epochs[21]. This predefined number of training epochs is be named the *patience* of the early stopping technique. Throughout the training process, the parameters of the model that achieved the best performance on the validation set are saved to be used when training is halted. It is expected that the best performing model on the validation set will also perform well on a test set.

While early stopping techniques based on performance computed on validation sets are the most commonly used, due to their efficacy and simplicity, there are situations where separating a validation set is costly and undesirable. One example are the active learning scenarios, where we have little labeled data especially in the earlier rounds of the AL process. For such situations, an early stopping technique that does not rely on validation sets is desired. The batch gradient disparity (BGD) technique proposed by Forouzesh and Thiran, which doesn't rely on validation sets, is presented next.

Batch gradient disparity early stopping

Forouzesh and Thiran[17] proposed an early stopping criterion that doesn't require a validation set. The intuition behind this technique is presented next. Suppose we have two training mini-batches S_1 and S_2. At any given iteration of the training algorithm we may select one of them to train the model with. Suppose we select mini-batch S_1, compute the loss $\mathcal{L}_{S_1}(\theta)$ for the model with parameters $\theta \in \mathcal{R}^d$, where d is the number of parameters of the model, and minimize this loss obtaining a model with parameters θ_{S_1}. If mini-batch S_2 was selected for training instead, the model's parameters would have been θ_{S_2}. At this point, the authors propose a theoretical generalization penalty R, given by:

$$R_2 = L_{S_2}(\theta_{S_2}) - L_{S_2}(\theta_{S_1}), \tag{2.13}$$

that measures the difference on the loss function on mini-batch S_2, if S_2 had been chosen for training instead of S_1. The motivation for this penalty is that an increase in the value of R indicates that the model is learning data structures that are specific to one of the mini-batches, therefore overfitting. The generalization penalty R is a theoretical penalty, because during training there are often several mini-batches, not only two as in the example, hence it is intractable to train the model one time for each mini-batch from the same starting parameters. Because of that the authors use the PAC-Bayesian framework[38] to define an upper bound on the sum of the expected penalties for each mini-batch as:

$$\mathcal{E}[R_1] + \mathcal{E}[R_2] \leq \sqrt{\frac{2KL(Q_2||Q_1) + 2ln\frac{2m_2}{\delta}}{m_2 - 2}} + \sqrt{\frac{2KL(Q_1||Q_2) + 2ln\frac{2m_1}{\delta}}{m_1 - 2}}, \tag{2.14}$$

where Q_1 and Q_2 are the posterior distribution of the model conditioned on mini-batches S_1 and S_2, respectively. m_1 and m_2 are the number of training samples in each mini-batch, and δ is the expected probability of selecting each mini-batch. The authors show that assuming the conditioned posterior distributions to be Gaussian, the Kullback-Leibler divergence between them are driven by the l_2 norm $||g_1 - g_2||_2$, where g_1 is the gradient of the model's parameters when trained using mini-batch S_1. Motivated by this fact, the authors propose the metric called *gradient disparity*, given by:

$$D_{i,j} = ||g_i - g_j||_2. \tag{2.15}$$

Where g_i is the gradient of the trained model computed on mini-batch S_i. Finally, the early stopping criterion is defined as:

$$\overline{D} = \sum_{i=1}^{S} \sum_{j=1, j \neq i}^{S} \frac{D_{i,j}}{S(S-1)}, \tag{2.16}$$

where S is the number of mini-batches. From the experiments performed on the original work, it has been shown that using only five mini-batches to compute the criterion of Equation 2.16 is enough to achieve good results. The authors also experiment with the consecutive BGD where the training is stopped when the BGD metric increases over consecutive epochs.

2.3 Active Learning

Machine learning models achieve high performance in various tasks. In order to do that, however, these models often require a big set of labeled data to be trained via supervised learning. There are two problems that may appear in this approach: (1) Difficulty on acquiring new samples due to the nature of the task, thus making it impractical to create big datasets, and (2) the cost of labeling all data samples exceeds the available budget for the project. Active learning comes as a solution for the latter. It works on the assumption that a big dataset can be well represented by a smaller subset of its samples, and ideally a model trained with the full dataset or with this smaller representative subset achieves similar performance. Therefore, active learning based models are designed to find a representative subset of samples that should be labeled by an oracle (e.g. human). And by using this subset of samples, it is possible to train models with good performance and reduced labeling cost as only a smaller subset of samples needs to be annotated.

There are two main scenarios where active learning algorithms may be used[29]: (1) Stream-based scenario, where data comes from a data source in consecutive fashion and samples are selected to be labeled as they are generated, and (2) Pool-based scenario, where a pool of unlabeled data is available and samples are ranked by the sampling function and the most informative ones are selected to be labeled. This dissertation focuses on the pool-based scenario, and Figure 2.8 shows a diagram for a pool-based active learning scenario.

Active learning algorithms are composed of three main parts: (1) A strategy to create the initial labeled subset, (2) a sampling function to choose the most informative unlabeled samples, and (3) a stopping criteria to stop the active learning algorithm. More details are given about each of these parts in the following sections.

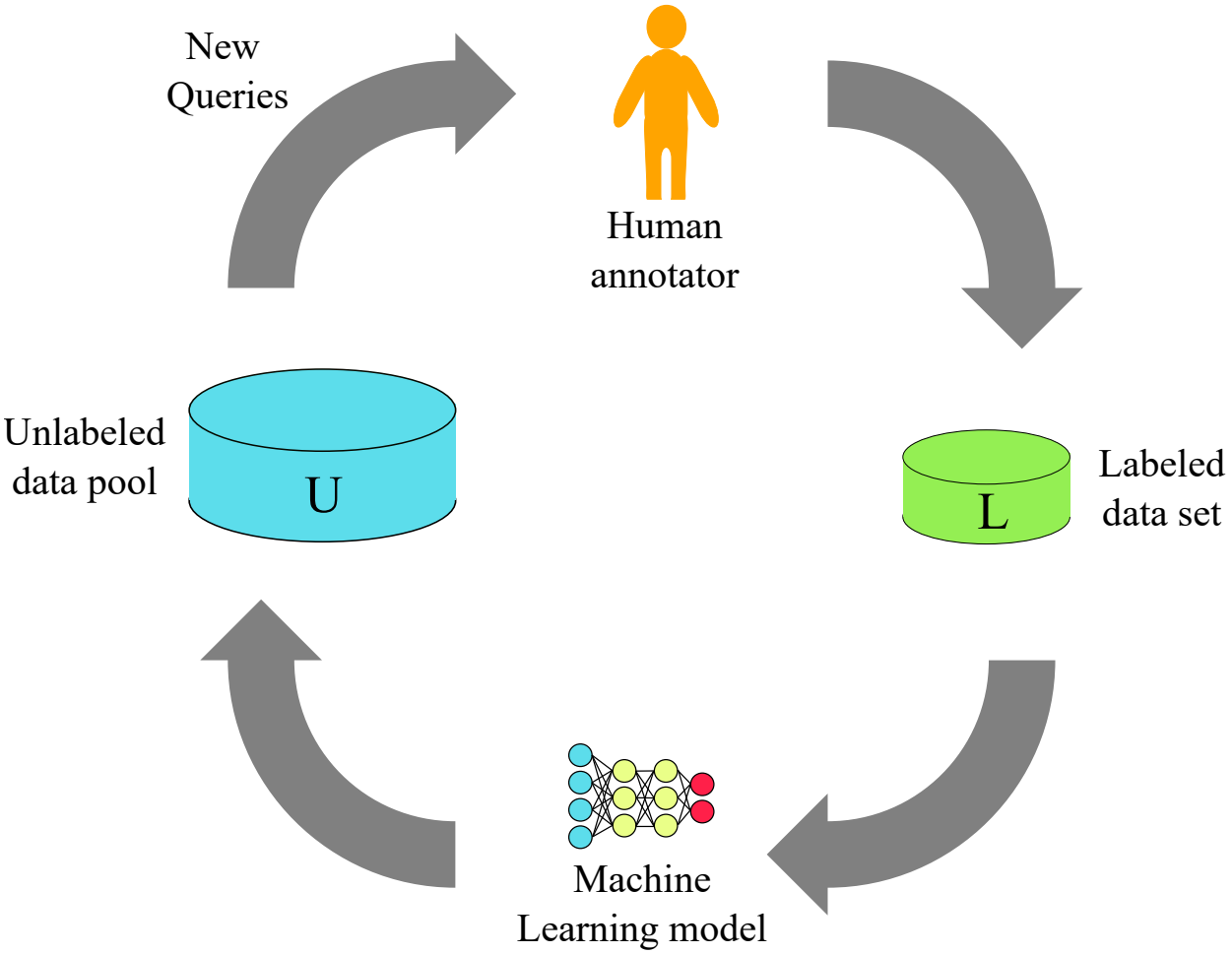

Figure 2.8: Diagram of a Pool-based active learning algorithm.

2.3.1 Creating the initial labeled subset

Active learning algorithms often work on a cold-start scenario, where no data is labeled from the start. Many works generate the initial labeled set by randomly querying samples from the unlabeled set. A random strategy, however, brings about the potential of generating a homogeneous and non-diverse initial labeled set. With that in mind, strategies to select the initial labeled subset are designed. They rely, in most part, on diversity metrics of the data itself. For instance, Gao et al [18] uses a measure of mutual information between unlabeled samples and clusters them with a K-means++ algorithm [2]. By doing that they create the initial labeled set by selecting at least one sample from each cluster, therefore increasing diversity of the first selected samples to be labeled.

2.3.2 Sampling functions

Sampling functions, or query strategies, are the core component of any active learning algorithm. They are responsible for selecting the most informative samples from the unlabeled pool of data and query them to be labeled by the oracle. Sampling functions may be separated into three categories: *Uncertainty sampling*, *diversity sampling*, and combinations of uncertainty and diversity measures.

Uncertainty sampling works on the assumption that the unlabeled samples that could bring the most new information to the model are those for which the model's predictions are most uncertain about. The Least confidence is one of the most common uncertainty-based sampling functions, where the unlabeled sample for which the model has the least confidence in its predictions is queried for the oracle.

Diversity sampling uses characteristics inherent to the data and chooses unlabeled samples that increase the diversity of the labeled set. The core idea is that the most different unlabeled samples will bring the most information to the labeled set as they are annotated.

2.3.3 Stopping criteria

All active learning algorithms have the same objective, maximize a model's performance with the least amount of annotation cost. Therefore, a stopping criterion is designed to stop the training process when the annotation of new unlabeled samples no longer increases the performance of the model in a significant manner. One major problem is how to compute the performance of a model. Active learning is implemented in environments with scarce annotated data. Because of that, creating a test set to evaluate performance is unlikely as it will either increase the annotation costs or decrease the model's performance by reducing the number of training data. Motivated by not using a test set, new stopping criteria were designed. Stopping criteria may be separated into 4 classes: (1) *Performance-based* criteria employs a test set and stops training when the performance of the model on the test set surpasses a predefined threshold; (2) *gradient-based* criteria stops the training of the model when the gradient during training is small, indicating that little new information is being brought from the unlabeled set; (3) *confidence-based* criteria stops the learning algorithm when the classifier has enough confidence on its predictions on the unlabeled set; and (4) *model specific* criteria depends on characteristics specific to the learner model to decide when to stop training.

2.4 Self-learning algorithms

Supervised training has been shown to achieve excellent results in deep learning, but requires big sets of labeled data to do so. Scenarios where a huge pool of unlabeled data is available is often found, and several self-learning algorithms were designed in order to take advantage of them. These algorithms are the core of semi-supervised learning, which mix supervised and self learning strategies. This work focuses on the teacher-student self-training techniques proposed in the literature.

Teacher-student self learning strategies use a model, initially trained on labeled data, to pose as a teacher that makes predictions on unlabeled data, which are later used as targets for the same model, now acting as a student, to train with. This approach has been used for many works in the literature, and it is able to achieve better performance when compared to pure supervised learning[14]. The targets generated by the model's predictions can be classified as either hard-targets or soft-targets. Hard-targets use the classes predicted by the model as targets for future training. Soft-targets, on the other hand, use the output probability distribution of the model as targets, an approach similar to that of knowledge distillation[26].

There are many classes of self-training algorithms applied in semi-supervised scenarios in the literature. This work will focus on two of them, that have been applied to sequence tagging tasks. The first is the pseudo-label method, while the second is the consistency regularization method for self-training. Pseudo-label techniques rely on the predictions made by the model as ground-truth hard-targets, often using the same loss function for both supervised training and self-training. Consistency regularization, on the other hand, relies on soft-targets and tries to increase the consistency of the model's predictions when different types of perturbations are applied to the input data.

Chapter 3

Related works

This chapter compiles the works presented in the literature that are related to the research done in this dissertation. Section 3.1 presents works from the literature that propose active learning techniques, section 3.2 presents works from the semi-supervised literature that propose novel self-training techniques, and section 3.3 concludes the chapter by describing the works from the literature that we will use in this dissertation.

3.1 Active learning algorithms

This section is separated into two parts. The first presents works proposed in the literature for pool-based active learning algorithms applied to NER, while the second compiles a series of works that proposed stopping criteria for active learning algorithms not restricted to NER.

3.1.1 Active learning applied to named entity recognition

The Google Scholar[1] database was used to identify an initial batch of articles proposing active learning algorithms for named entity recognition tasks. This initial search used the keywords *active learning* and *named entity recognition*. The search was limited to papers published in 2016 or later, returning 1500 results from which the 30 most relevant were selected to be read. If a paper used an AL algorithm proposed by another, the cited paper was added to the pool of papers to be read, even if it was published prior to 2016. The relevant works presented next are separated by the machine learning model they employed for the active learning algorithm, as to depict which models are more/less popular in this research area. Works that introduced new datasets in less popular languages and presented little novelty on actual active learning algorithms research were not reported.

[1] https://scholar.google.com/

Conditional random fields

Chen et al[13] uses a CRF with clinical domain optimized features and compares the performance of uncertainty, diversity and random sampling functions. Uncertainty-based sampling was the most successful approach in the experiments, resulting in 66% of annotation cost reduction.

Gao et al[18] proposes an active learning algorithm with a cold-start technique. In order to select the initial batch to be labeled, the algorithm models mutual information between sentences and clusters them based on their similarities. This procedure allows the initial labeled batch to be diverse, as sentences are queried from all clusters. After the initial batch has been labeled, the algorithm proceeds with a committee-based sampling, as two CRF models are trained with two random splits of the labeled set at each active learning iteration and disagreement between the models is used to query new samples.

Tran et al[62] combines self-learning and active learning for training CRF models. The active learning algorithm uses a diversity sampling function to choose the most informative unlabeled sentences and asks for them to be labeled by a human annotator, while the self-learning selects sentences for whom the trained model has high confidence in its predictions and adds them with the predicted labels to the labeled training set. Experiments using a dataset of samples extracted from Twitter compared active learning algorithms with uncertainty and diversity sampling, both with and without self-learning. The AL algorithms with self-learning have shown, in general, significantly better performance in the experiments.

Settles and Craven[55] compare uncertainty, diversity and compositions of both as possible sampling functions for active learning algorithms based on the CRF model. It was shown that using the information density sampling function, a mixture of uncertainty and diversity measures, achieved the most consistent results, sometimes achieving the best results and never performing poorly when compared to other sampling functions.

Support Vector Machines

Shen et al[56] proposed an active learning algorithm based on the combination of uncertainty and diversity measures for the sampling function for SVM models. The uncertainty function was to select samples that are closer to the decision boundaries of the trained SVM, while the diversity function selected samples by clustering them based on their information. A combination of the two is the proposed sampling function, and the experiments show that it achieves a significant reduction in labeling costs while still achieving a performance close to that of a fully supervised trained model.

Logistic regression

Kobayashi and Wakabayashi[31] state that selecting entire sentences to be labeled is a waste of annotation effort as not all words bring new information to the model. They propose to use a multi-class logistic regression model with point-wise predictions[44] to identify unlabeled words that could be highly informative to the trained model, and thus only a few words should be labeled instead of whole sentences.

Neural models

Shen et al[57] presents the first active learning algorithm based on neural networks applied to a sequence tagging task. It uses convolutional layers for character and word-level feature encoding and an LSTM layer with greedy decoding as the tag decoder. The authors propose the *Maximum Normalized Log-Probability (MNLP)* sampling function in order to minimize the effects the length of a sentence has on the model's confidence for it. Unlike prior works, the trained model isn't retrained for each active learning iteration. Instead, it is continually trained for a smaller number of epochs at each iteration, which relies on the validation set for early stopping of the supervised training to avoid overfitting and reduce training time. The experiments show that the proposed algorithm trains a model with performance close to the state-of-the-art using only 25% of the training set of the OntoNotes5.0 English dataset[49].

Siddhant and Lipton[58] extend the work done by Shen et al, they rely on the Bayesian Active Learning through Disagreement (BALD)[28] technique, which queries the unlabeled samples that generate the most disagreement from multiple passes on Bayesian neural networks and apply it to sequence tagging problems. In order to generate uncertainty inside the model, they propose two methods, the Monte Carlo dropout and the Bayes by backpropagation. The Monte Carlo dropout method makes several predictions on the same unlabeled sample with multiple dropout masks. The Bayes by backpropagation modifies specific layers of neural models turning its deterministic parameters to stochastic ones. Both of these methods are able to generate different outputs for the same input and the disagreement generated between them is used to query unlabeled samples. The experiments performed have shown that the proposed sampling functions perform consistently better than the MNLP proposed by Shen et al[57], but the gain is marginal. Similar to the work of Shen et al, they use the validation set for early stopping of the model training throughout the active learning process, but limit the size of the validation set to match the size of the current labeled training set.

Zheng et al[66] proposed a committee based active learning algorithm dependent on biLSTM-CRF models with attention mechanisms. The sampling function is based on

the disagreement of the committee of models. Each model needs to be retrained at each active learning iteration, which slows the proposed algorithm. The authors state that hyper-parameter tuning was not performed, as they used the default values from the deep learning framework they used. It is unclear whether the training hyperparameters, such as learning rate and batch size, were optimized using the validation set or not.

Radmard et al [50] propose a subsequence based active learning algorithm. The idea is that instead of querying full sentences for annotation, only the most uncertain non-overlapping subsequences are queried for annotation. The subsequences are queried without its surrounding context for annotation, and saved into a dictionary that maps a subsequence to its true labels. Then, this dictionary is used to propagate the labels to similar unlabeled subsequences. All sentences that contain annotations, either from the oracle or by label propagation, are used for training the model. Since sentences may be partially annotated, the authors propose to use backpropagation only on labeled tokens. The proposed algorithm has three main hyperparameters that need to be tuned, which are: The minimum length of a subsequence, the maximum length of a subsequence and the normalization factor $\alpha \in [0, 1]$. The baseline of comparison was the algorithm proposed by Shen et al[57]. The authors report that the proposed subsequence-based algorithm was capable of training the CNN-CNN-LSTM model to near peak performance using approximately 13% and 27% of the training set for the datasets OntoNotes5.0 and CoNLL2003, respectively.

Table 3.1 compares the works on deep active learning proposed in the literature. From it we observe that most works rely on validation sets for tuning of hyperparameters and early stopping. We also note that most of them require the oracle to annotate all tokens in the queried sentences.

3.1.2 A literature review on stopping criteria for active learning algorithms

Two databases were used to identify works proposing stopping criteria (SC) for active learning algorithms, namely *Google Scholar* and *Scopus*[2]. The keywords used in the search were *active learning* and *stopping criteria*, only papers published in 2016 or later were considered. This resulted in 799 papers from the Google Scholar and 22 from the Scopus databases. The 30 most relevant articles from the Google Scholar and all articles from the Scopus were selected for a first analysis. If a paper used a stopping criteria proposed by another work, the cited paper was added to the list of papers to be read. The stopping

[2]scopus.com

Table 3.1: Comparison between the current works on deep active learning from the literature.

Algorithm	Whole sentence annotation	Use validation set for tuning		Use validation set for early stopping	Drawbacks
		Model hyperparameters	Training hyperparameters		
Shen et al [57]	Yes	Yes	Yes	Yes	- Use validation set for early stopping - Annotate all tokens form queried sentences
Siddhant and Lipton [58]	Yes	Yes	Yes	Yes	- Use validation set for early stopping - Annotate all tokens from queried sentences
Zheng et al [66]	Yes	No	-	Yes	- Expensive to train committee of neural models
Radmard et al [50]	No	Yes	Yes	Yes	- Proposed sampling function requires multiple hyperparameters to be tuned - Subsequences are annotated without context information

criteria proposed in the literature are presented next, separated into 4 classes as presented in section 2.3.3.

Performance-based stopping criteria

Performance-based SC are, in general, not ideal for active learning as they usually require a set of labeled samples aside of training for the computation of the performance metric. Zhu et al[67] circumvented this problem by proposing the *selected accuracy* stopping criterion, a performance-based method that used the freshly queried batch of samples as a test set. This method verifies the model's accuracy using the new queried batch after it is labeled by the oracle, but before allowing the model to be trained using it. The active learning algorithm is stopped when the accuracy reaches a predefined threshold.

Confidence-based stopping criteria

Ghayoomi[20] proposes to use the mean and variance of the model's confidence on predictions of the most uncertain unlabeled samples (chosen by an uncertainty-based sampling function), and to interrupt the algorithm when the calculated mean starts to stagnate and variance starts to decrease.

Laws and Schütze[34] proposed two confidence-based SC, the *minimum absolute performance* and the *maximum possible performance*. The former estimates a performance metric (e.g. f1-score, accuracy) of the model on the unlabeled set of samples and stops the active learning procedure if it surpasses a predefined threshold. The latter works in a

similar fashion, only that it now stops the active learning algorithm when the estimated performance stagnates.

Vlachos[64] measures the model's confidence on its predictions for all the unlabeled samples after each supervised training step in the active learning algorithm, and dictates that training should be stopped when the confidence starts decreasing. It assumes that when that happens, no more informative labels are present in the unlabeled set.

Zhu et al[67] propose three confidence-based stopping criteria, namely *maximum uncertainty*, *overall uncertainty*, and *minimum expected error*. The maximum uncertainty proposes that if the confidence of the model on the most uncertain unlabeled sample is higher than a predefined threshold the training process can be stopped. The overall uncertainty works in a similar fashion, but it uses the mean of the model's confidence on all unlabeled samples. The minimum expected error computes the average uncertainty on the unlabeled set assuming that the true label for each sample is the prediction given by the current model, and states that training should be stopped when the expected error falls below a predefined threshold.

Gradient-based stopping criteria

Laws and Schütze[34] propose a gradient-based SC that stops the active learning algorithm when the gradient of the model during training vanishes, getting closer to zero. Wang et al[65] proposed another gradient-based SC that should also stop the algorithm when the gradient gets smaller than a predefined threshold, the main difference with prior works is that now the gradient is estimated using the unlabeled samples, and all possible labels are to be considered. The estimated gradient w.r.t. an unlabeled sample x is computed as

$$E\left\|\frac{\partial L(x)}{\partial \theta}\right\| = \sum_{\hat{y}\in Y} p(\hat{y}|x)\left\|\frac{\partial L(x,\hat{y})}{\partial \theta}\right\|. \tag{3.1}$$

Where $p(\cdot)$ is the probability given by the trained model, Y is the set containing all possible labels for the unlabeled sample x, and $L(\cdot)$ is the loss function given an input x and a label y. The active learning algorithm is stopped if the estimated gradient for all unlabeled samples is smaller than a predefined threshold.

Model specific stopping criteria

Model specific stopping criteria were presented for SVM models by Schon and Cohn[54] and Ertekin et al[16]. Both stopping criterions are based on the principle of margin exhaustion for support vector machines, dictating that the active learning algorithm should be stopped when the number of support vectors remains constant even though new unlabeled samples are queried and added to the labeled set.

3.2 Self-training techniques for sequence tagging tasks

Many techniques were proposed for utilizing unlabeled data to enhance the performance of neural models trained for the most varied tasks. This section presents some of them that were designed for sequence tagging tasks. These approaches have been separated into 2 classes, namely: (1) Pseudo-labels, and (2) consistency regularization.

3.2.1 Pseudo-label

Lin et al[35] and Tran et al[62] proposed the use of an active-self learning algorithm applied to chinese and twitter text domains, respectively. For this algorithm, highly reliable unlabeled samples are automatically labeled by the model, with highly reliable samples being identified as those for which the model's confidence is higher than a predefined threshold. The results from both works indicate that this self-learning technique may reduce the annotation costs in active learning scenarios.

3.2.2 Consistency regularization

Consistency regularization is done by injecting perturbations to the unlabeled input data, and reinforcing the model's output distribution to match that obtained from the unaltered unlabeled data.

Cross-view training (CVT)

The cross-view training, proposed by Clark et al [14], is a self-training procedure inspired in the multi-view learning[7]. The self-training process intends to enhance the model's representation learning and regularization by forcing it to output the same predictions across different views of the input. The proposed algorithm uses auxiliary classification heads to the neural model, where each head learns to predict tokens given a limited view of the input sentence (e.g. left-context only, right-context only). Each auxiliary classification head is composed of two layers, where the first is a fully-connected layer with ReLU[43] activation, followed by a softmax function in order to generate a distribution over predicted classes. The authors proposed four auxiliary classification heads, with different restricted views, for the CVT training for NER. They are:

1. Forward head: Generates the distribution over classes for the current token given only the left-context.
2. Backward head: Generates the distribution over classes for the current token given only the right-context.

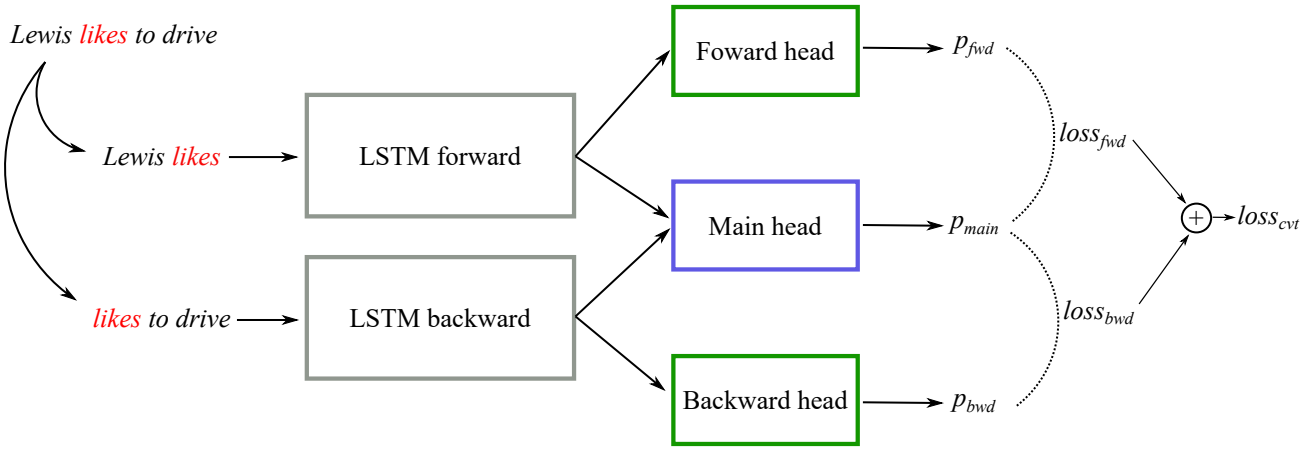

Figure 3.1: Diagram of the cross-view training technique, with two auxiliary classification heads. The word to be classified, *likes*, is shown in red. The forward and backward LSTM encoders encode the context information of the sentence in each direction. The main classification head uses both context sides to produce the class distribution for the word *likes* (i.e. p_{main}). The two auxiliary modules also generate a distribution over classes for the word *likes* (i.e. p_{fwd}, p_{bwd}), but with restricted input. The CVT loss in this case is the sum of the Kullback-Leibler divergence between the distribution given by the main classification head and the distributions given by the auxiliary modules.

3. Future head: Generates the distribution over classes of the unseen next token (i.e. to the right of the current one) given the left-context.

4. Past head: Generates the distribution over classes of the unseen previous token (i.e. to the left of the current one) given the right-context.

The CVT loss is the sum of a divergence metric between the distributions generated by the main classification head, and all the auxiliary classification modules. The original work uses the Kullback-Leibler divergence as the divergence metric. Figure 3.1 demonstrates how to compute the CVT loss in a setting where only the forward and backward heads are used.

Word dropout

The word dropout[14, 57, 32] can also be used as the base for a self-training algorithm. The word dropout consists of randomly selecting some words of a sentence, and replacing them with a special $< removed >$ or $< UNK >$ token. We can then acquire the soft-targets of the trained model for an unlabeled sample that sees all the original tokens, replace some of them with the special token, and train the model so that predictions are consistent with partial information. For sequence tagging tasks, the CVT has the advantage of training multiple-views at once, while the word dropout is restrained to a

single partial view. The CVT was, however, proposed for a neural model that uses LSTM encoders and may be complex to adapt to other architectures, while the word dropout is model-free, being easily adapted to most neural architectures for sequence tagging.

Virtual adversarial training (VAT)

Another self-training method is the virtual adversarial training (VAT), proposed by Miyato et al[40] for text classification and applied to sequence tagging by Clark et al[14] and Chen et al[11]. The VAT is an extension of the adversarial training (AT) proposed by Goodfellow et al[22], where noise is added to the input data, which is designed to incur the most change in the model's output distribution. The main difference between VAT and AT is that the former extends the latter to the case of self-training on unlabeled samples. VAT estimates the local gradient of a model's predictions using multiple forward-backward passes on the neural model, identifying the perturbation for which the model is most sensitive to. By adding this adversarial perturbation to the input, and minimizing the KL-divergence between the model's output distribution for the original and the modified inputs, the model's decision boundaries become locally smoother. This approach has been used successfully in text[40] and image classification[41]. One disadvantage of the VAT is that it requires the fine-tuning of its hyperparameters, namely the norm of the vector perturbation ϵ and the loss factor $\lambda_{VAT} \in (0, 1)$ that assigns a weight to the unsupervised VAT loss.

Paraphrasing

Chen et al[10] proposes to use paraphrasing of unlabeled samples, translating the sentence to an intermediate language and then translating it back, in order to generate perturbations on the unlabeled sentences. The authors hypothesize that even though words might change order and some words might disappear in the paraphrasing process, named entities are usually unchanged. Therefore, the proposed self-training method consists in computing the number of tokens pertaining to a named entity in both the original and the paraphrased sentence, and using the difference as a consistency loss. The supervised loss and the consistency loss are added and minimized together.

Synonym replacement

Lakshmi et al[32] compared the impact of different perturbations applied to self-training of NER models. The strategies compared were the addition of Gaussian noise on word embeddings, word dropout and token replacement by synonyms. From the experiments, it was noted that the approaches of word drop and token replacement by synonyms

performed at least as good as the Gaussian noise technique, while requiring less parameters to be tuned.

Virtual adversarial discrete perturbation

Lakshmi et al[32] presented results showing that discrete perturbations are competitive against continuous perturbations, while being simpler to implement and requiring less parameters to be tuned. Going in a similar direction, Park et al[46] proposed an extension to the VAT originally presented by Miyato et al[40] by replacing the continuous adversarial noise for a discrete token level adversarial perturbation. The discrete token replacement is designed to maintain the semantics of the sentence, but maximize the consistency loss of the model's predictions between the original and the modified sentences. In order to maintain sentence semantics, for each token that is selected to be replaced, a masked language model[15] identifies the tokens from the vocabulary that are the most similar to it. From the top-K sampled tokens with high similarity to the one being replaced, the ones that incur the greatest consistency loss are chosen. This approach, however, requires a pretrained masked language model, and identifying the best adversarial tokens can be time-consuming for bigger datasets, resulting in 250% to 350% increase in execution time when compared to the standard supervised training.

seqVAT

Chen et al[12] proposed the seqVAT technique which extends the original virtual adversarial training to a model that has a CRF classification layer. The authors propose to use the probability distribution over the entire sequence to compute the divergence of the VAT technique, instead of doing so for each token as done in other works[14]. But computing the probability distribution over all possible label sequences is intractable. The authors then propose to use a k-best viterbi decoding algorithm[30] to identify the k label sequences with highest probability and use these probabilities as the distribution, they also include an additional dimension that includes all the other possible label sequences. Therefore, the output distribution to be used for the computation of the divergence metric will be a vector of dimension $(k + 1)$. The computation of the VAT perturbation is unchanged from the original work [41], but now the authors propose to alternate mini-batches of unlabeled and labeled samples for training, contrary to the original work[41] that optimized both the supervised and the VAT loss together.

Table 3.2: Comparison between different self-training techniques applied to sequence tagging.

Technique	Parameter tuning	Advantage	Drawback
Cross-view training	No	+ More efficient form of word dropout	- Requires biLSTM encoder
Word Dropout	No	+ Aplicable to wide range of models + Improves model regularization	- Little improvement on prediction
Virtual adversarial training	Yes	+ Improves model regularization	Small improvement for sequence tagging
Paraphrasing	No	+ Applicable to wide range of models	- Requires translation model
Synonym replacement	No	+ Applicable to wide range of models	- Hard to use in specific domains (e.g. medic, legal)
Virtual adversarial discrete perturbation	No	+ Intuitive adversarial perturbation + No hyperparameters for fine-tuning	- Requires MLM for synonym search - Hard to use in specific domains (e.g. medic, legal)
seqVAT	Yes	+ Increases VAT performance for CRF model	Restricted to sequence models (e.g. CRF)

3.3 Concluding remarks

Section 3.1.1 presented the works from the literature that proposed active learning algorithms applied to the task of NER. In this dissertation we will focus on those that work with neural models, which are the current SOTA for this particular task. It was shown that current works on deep active learning from the literature still heavily rely on validation sets for early stopping, which may be undesirable in many active learning scenarios. Also, promising results have appeared recently on the use of partial annotation for deep active learning[50], which effectively reduces annotation costs. They require, however, multiple subsequences to be evaluated in the same sentence, including those that overlap one another, which adds a considerable computational overhead to the querying process.

In order to leverage the unlabeled samples to improve our model's performance in an active learning scenario, which could potentially reduce annotation efforts by allowing the model to achieve peak performance with less human labeled data, section 3.2 presented a brief review of self-training techniques proposed in the semi-supervised training literature. It focused on techniques proposed for NER. Some techniques will be evaluated on experiment 3 presented in section 4.4.

Next section 4 presents the experiments to be performed in order to validate our proposed hypotheses, presented in section 1.3.

Chapter 4

Methodology

This chapter presents the methodology for the three experiments that are to be performed in order to validate the hypotheses proposed in section 1.3. Section 4.1 introduces the datasets and neural models that will be used on all three experiments, along with the performance metrics and hardware setup used. Section 4.2 presents the methodology of the first experiment, which is a comparison between different early stopping techniques, including our original proposed technique, when applied to deep active learning algorithms. Section 4.3 presents the methodology for the second experiment, where we demonstrate the sensitivity of the ASL algorithm proposed in the literature[62] to the initial set of labeled samples, and how our proposed deep active-self learning (DASL) algorithm performs under the same circumstances. Section 4.4 presents the methodology of the third experiment, where we evaluate the performance of the proposed DASL algorithm with different self-training techniques.

4.1 Datasets, Neural models and general setup

This section introduces the datasets, neural models, performance metrics and hardware setup to be used in all the proposed experiments.

4.1.1 Datasets

Many NER datasets exist in the most varied languages. For the experiments in this dissertation, we use two consolidated English NER datasets and one legal domain Portuguese NER dataset.

One legal domain NER dataset in Portuguese was also chosen to be experimented with. It is a new legal domain NER dataset in Portuguese named *Aposentadoria*[1]. It contains

[1]https://avio11.github.io/resources/aposentadoria/aposentadoria

Table 4.1: Datasets to be used for experiments on NER.

Dataset		CoNLL03	OntoNotes5.0	Aposentadoria
Domain		Reuters News	Variety	Brazilian legal texts
Language		English	English	Portuguese
Train set	Sent.	14,987	59,924	3,860
	Token	203,621	1,088,503	311,231
Valid set	Sent.	3,466	8,528	828
	Token	51,362	147,724	68,740
Test set	Sent.	3,684	8,262	827
	Token	46,435	152,728	65,912

named entities from 10 classes associated with retirement acts of public employees from the *Diário Oficial do Distrito Federal* (Brazilian Federal District official gazette, in direct translation).

Table 4.1 presents the datasets selected to be used, and presents relevant information about each one of them such as language and domain area.

All datasets were preprocessed by converting their original IOB notation to the IOBES notation, which is frequently adopted by works on NER in the literature[14][33][36] for being able to improve model performance in NER tasks[51]. Additionaly, all tokens with numbers with more than two digits in the English datasets were replaced by # characters. For the Portuguese dataset, all number digits were replaced by the 0 digit. Both replacements were done so that the datasets are compatible with the pretrained embeddings used in the NER models presented in the next section 4.1.2.

4.1.2 Neural Models

The models to be used are the same that appear in the experiments performed by Siddhant and Lipton[58], namely the CNN-CNN-LSTM introduced by Shen et al[57] and the CNN-biLSTM-CRF designed by Ma and Hovy[36]. An in-depth description of both models is presented next.

CNN-CNN-LSTM model

The CNN-CNN-LSTM model consists of a character-level convolutional encoder, followed by a word-level convolutional encoder and an LSTM tag decoder.

The character-level CNN is used to generate character level vector representations of a word. This CNN works by first transforming each character of a word into a vector representation, with embeddings being initialized with uniform samples from $[-\sqrt{\frac{3}{dim}}, +\sqrt{\frac{3}{dim}}$ as proposed by Ma and Hovy[36]. Dropout[60] is applied to the generated embeddings as presented by Ma and Hovy[36]. Then, a one dimensional convolutional layer is used

to extract information of neighboring characters, followed by ReLU activation[43] and max-pooling to generate fixed-size character-level word embeddings.

In order to generate a word representation, the character-level vector is then concatenated with a vector representation from a lookup table, which is initialized with pretrained word embeddings that are updated throughout training. For this work, GloVe embeddings of 100 dimensions pretrained on English newswire corpus[47] were used for the English NER datasets, while GloVe embeddings of 300 dimensions pretrained on multi-genre Portuguese corpus[24] were used for the Portuguese NER dataset.

The word-level CNN receives as input the full-embedding of each word and produces an encoded representation considering the immediate neighborhood of each word in a sentence. The word-level CNN is made up of N consecutive blocks of neural layers, where N is a hyperparameter of the model and each of these blocks is formed by a one dimensional convolution, followed by ReLU nonlinearities and dropout. The output of the word-level CNN is the concatenation of the full-embedding it received as input with the encoded representation it computed, forming a residual connection between input and output.

The LSTM tag decoder is responsible for the tagging of each word. It decodes a sequence of tags in a greedy manner, using the previous decoded tag and the encoded representation of the current word to be classified as inputs in order to make predictions. The LSTM tag decoder uses a one-hot encoded vector of the previous tag, concatenates it to the encoded vector representation of the current word and uses it as input. An LSTM layer generates a latent representation for this input, that a fully connected layer followed by a softmax function uses to generate a distribution of probabilities over all possible tags.

The model's hyperparameters used for the English NER datasets were similar to those presented in the experiments of Siddhant and Lipton[58], where the character-level CNN uses 25 dimensional character embeddings, the 1-d convolution layer has 50 filters, kernel size of 3, stride and padding of 1. The word-level CNN has two blocks, where each block has a 1-d convolutional layer with 800 filters, kernel size of 5, stride of 1 and padding of 2. The LSTM tag decoder consists of an LSTM layer of size 256. All dropout layers for the CNN-CNN-LSTM model have probability 0.5.

For the Portuguese NER dataset, a grid-search was performed using the validation set. The parameters used in the grid search are presented in Table 4.2. The selected hyperparameters were: The character-level CNN had the same parameters as that used for the English datasets. The word-level CNN has one block, consisting of a 1-d convolution layer with 400 filters, kernel size of 5, stride of 1 and padding of 2. The LSTM tag decoder has size 128. All dropout probabilities are still of probability 0.5.

Table 4.2: Hypeparameters used for the grid-search of the CNN-CNN-LSTM model. For the character-level CNN, we used the parameters from the original work, from Shen et al[57]

Module	Word CNN		LSTM decoder
Parameter	Blocks	Filters	Cell size
Values	[1, 2]	[200, 400, 600, 800]	[128, 256]

For training, the CNN-CNN-LSTM uses a Cross Entropy loss function, and the optimization of parameters is performed by stochastic gradient descent. Training hyperparameters were: momentum of 0.9, gradient clipping of 5.0, and learning rate of 0.015 for English datasets, and 0.010 for the Portuguese dataset.

CNN-biLSTM-CRF model

Similarly to the CNN-CNN-LSTM model, the CNN-biLSTM-CRF model uses a CNN to generate a character-level representation for each word. This character-level embedding is then concatenated with a word-level embedding from a lookup table, which is initialized with pretrained weights.

The full embedding, formed by the concatenation of character-level and word-level embeddings, is then fed to a biLSTM layer, that generates vectors that are encoded representations for each word in a sentence. A fully-connected layer is then used to reduce the dimension of the encoded vector's dimension to the number of possible tags. The reduced dimension vector is then fed to a CRF layer. Dropout layers are used before and after the biLSTM layer. All weight matrices for the biLSTM and fully-connected layers are randomly initialized using a uniform distribution to select samples from $[-\sqrt{\frac{6}{r+c}}, +\sqrt{\frac{6}{r+c}}$ as proposed by Ma and Hovy[36], where r and c are the number of rows and columns in the weight matrix. All the bias parameters from the biLSTM layer are initialized to be 0.0, except for the forget gate bias which is initialized to 1.0. All dropout probabilities are set to 0.5.

The model's hyperparameters for the English NER datasets are similar to those presented in the experiments of Siddhant and Lipton[58], where the character-level CNN uses 30 dimensional character embeddings, the 1-d convolution has 30 filters, kernel size of 3, padding of 1 and stride of 1. The biLSTM layer has size 300. All dropout probabilities are set to 0.5.

For the Portuguese dataset, a grid-search was employed to define the model's hyperparameters. The list of parameters used in the grid-search is presented in Table 4.3. The chosen model had the same parameters for the character-level CNN as that used for English datasets, while the biLSTM layer has size 256.

Table 4.3: Hyperparameters used for the grid-search of the CNN-biLSTM-CRF model. For the character-level CNN, we used the parameters from the original work, from Ma and Hovy[36].

Module	biLSTM encoder
Parameter	Cell size
Values	[128, 256]

For training, the CNN-biLSTM-CRF model uses the negative log-likelihood, as is the usual with CRF models, and the optimization is performed by stochastic gradient descent. Training hyperparameters were: momentum of 0.9, gradient clipping at 5.0, and learning rates of 0.015 for English datasets, and 0.0025 for the Portuguese dataset.

4.1.3 Performance metrics

In order to measure the performance of all active learning based algorithms, the micro-averaged exact-match (also known as span-based) f1-score will be used.

4.1.4 Hardware setup

All experiments were implemented and executed on the Google Colab platform[2] with a pro subscription. GPUs available were the Tesla P100, the Tesla V100, and the Tesla T4 all with 16GB.

4.2 Experiment 1: early stopping comparison

In this section, we present the methodology for the execution of the first experiment that compares the effects of different early stopping techniques on the performance of a neural model trained by a deep active learning algorithm. Section 4.2.1 describes the deep active learning algorithm to train the neural models, and how the early stopping techniques are implemented in it. In section 4.2.2 we introduce our proposed *dynamic update of training epochs* strategy for early stopping. Section 4.2.3 presents other early stopping techniques to be used as baselines.

4.2.1 Deep active learning algorithm

The deep active learning algorithm implemented is similar to previous works by Shen et al[57] and Siddhant and Lipton[58]. The algorithm for the DAL process is presented in Algorithm 1, where U represents the set of unlabeled data, L is the set of labeled data,

[2]https://colab.research.google.com

m is the machine learning model, Q is the query budget that represents the number of words that can be labeled by the oracle in one iteration of the DAL algorithm, S is the percentage of samples to be annotated from the whole dataset for the DAL algorithm to stop. $Query(\cdot)$ identifies the most informative unlabeled samples from the unlabeled set U by using a sampling function (e.g. least confidence), and returns a subset of unlabeled samples u with at most Q tokens to be annotated. $Oracle(\cdot)$ represents the human annotator that receives a subset of unlabeled samples u, labels it and adds it to the labeled set L. $Train(\cdot)$ represents the supervised training of the model using the annotated set L, it returns the trained model. The number of training epochs depends on the early stopping technique that is employed.

Algorithm 1 Active learning algorithm

```
procedure AL(L, U, m, Q, S)
    m ← Train(m, L, epoch)
    while |L| / (|L|+|U|) < S do
        u ← Query(U, m, Q)
        L ← Oracle(u)
        m ← Train(m, L)
```

The general parameters for the implementation of the DAL algorithm were similar to the previous works from the literature[57][58]. The labeled set was initialized with 1% of the whole training set randomly. The maximum number of training epochs was set to 50. The batch sizes were respectively 80, 16, 16 for the datasets OntoNotes5.0, CoNLL2003 and *Aposentadoria*, respectively. The active learning algorithm was stopped when 50% of the training set was labeled. The query budget was set to be approximately 2% of the number of tokens in the training set, with values of $20,000$, $4,000$ and $6,000$ for the datasets OntoNotes5.0, CoNLL2003, and *Aposentadoria*, respectively. Validation sets were not used for training, except for early stopping when necessary. The sampling function to be used is the *Maximum Normalized Log-Probability* (MNLP), which is introduced next.

Sampling function

The function to be used for querying samples to be hand-annotated is the *Maximum Normalized Log-Probability* (MNLP) proposed by Shen et al[57]. The MNLP was chosen as its computational cost is far lower than the Bayesian sampling functions proposed by Siddhant and Lipton[58], the current state-of-the art for deep active learning on sequential tagging tasks, while still being competitive with them.

The MNLP comes as an extension of the well-known least confidence (LC) sampling function. The LC has a tendency of selecting longer sequences, as those tend to have lower probabilities. The MNLP tries to reduce the impact of this bias by transforming the probability of a sequence into log-space and dividing it by the sequence's length. It has shown promising results on DAL for sequence tagging tasks in experiments performed by Shen et al[57], and Siddhant and Lipton[58].

The MNLP can be computed, given a sequence of elements X of length N, as

$$MNLP(X) = \max_{y_1,...,y_{N-1}} \frac{1}{N} \sum_{i=0}^{N} log\ P(y_i|x_i, y_0, y_1, ..., y_{i-1}). \tag{4.1}$$

Where x_i is the i-th element of sequence X, and y_i is the class attributed to x_i. The sequences to be queried are those with the lowest MNLP values.

4.2.2 Dynamic update of training epochs (DUTE)

Inspired by the *overall uncertainty* stopping criterion for active learning algorithms proposed by Zhu et al[67], we propose to use the model's overall confidence on its predictions for the unlabeled samples as a measure to reduce training epochs in later iterations of the active learning process.

If we assume that the model's confidence on its predictions of unlabeled samples indicates how well the labeled set represents the whole dataset, then fewer unlabeled samples are likely to be truly informative as the model's confidence increases. Meaning that the process of training the model becomes closer to a fine-tuning process in later rounds of the AL run, which would require fewer training epochs to be used.

To measure the confidence of the model's predictions on the unlabeled samples, we propose to use the harmonic mean of the model's normalized predictions for each unlabeled sample. The normalized prediction uses the MNLP presented in section 4.2.1 but transforming the resulting log-probability back into a probability value. The normalized confidence of a sequence of elements X can be computed as

$$NC(X) = e^{MNLP(X)}, \tag{4.2}$$

then, the mean confidence can be computed as

$$MC(U) = \frac{|U|}{\sum_{s \in U} NC^{-1}(s)}, \tag{4.3}$$

where U denotes the unlabeled set of samples, $MC(\cdot)$ is the harmonic mean confidence and s is an unlabeled sample in U. Finally, to update the number of training epochs at the k-th active-self learning iteration, we propose the use of the equation

$$\begin{aligned} epoch(k) =&(1.0 - momentum) \times (1.0 - MC(U)) \times epoch(k-1) + \\ &(momentum) \times epoch(k-1) \end{aligned} \tag{4.4}$$

where momentum is introduced to avoid rapid decay of training epochs in early rounds of the active learning process. The hyperparameters of the DUTE strategy are: The momentum, and the number of training epochs for the first round of the active learning algorithm (i.e. $epoch(0)$).

4.2.3 Baselines for comparison

As baselines for comparison, we selected three early stopping criteria. Namely, (1) traditional early stopping based on the f1-score on the validation set, (2) traditional early stopping based on the loss function on the validation set, and (3) batch gradient disparity (BGD) proposed by Forouzesh and Thiran[17]. The BGD was experimented with as it is an early stopping criterion that doesn't rely on a validation set, a similar motivation to our DUTE strategy. We also experimented with the evidence-based early stopping criterion proposed by Mahsereci et al [37], but in initial experiments we observed that the model was continually trained for the full number of training epochs without early stopping throughout the active learning process. Due to the high computational cost, we stopped performing experiments with this criterion. Also, it is important to note that we use early stopping techniques that rely on validation sets as hypothetical baselines, as they are not viable in real world low-resource scenarios where active learning applications thrive.

For all the mentioned early stopping techniques we will use the same patience value of 5, the same value used in the original BGD work[17], meaning that the training process is interrupted after 5 epochs of no improvement of the early stopping metric (e.g. f1-score, loss, BGD). We will also evaluate the consecutive BGD, where the BGD metric needs to increase over 5 consecutive epochs for the training to be interrupted.

The comparison between the different early stopping techniques will be done by evaluating both the trained model's performance on the test set and the number of training epochs necessary for it in each active learning iteration. The best early stopping strategy is the one that trains the best model with the least number training epochs.

4.3 Experiment 2: Sensitivity of the active-self learning algorithms

In this second experiment, we wish to demonstrate the sensitivity of the ASL proposed in the literature[62] to the initial set of labeled samples. We hypothesize that a poorly representative initial labeled set, the one used to train the first model in the ASL algorithm, can negatively impact the final model's performance. We also propose a new DASL algorithm designed to minimize this sensitivity, which we present in section 4.3.2. The traditional active-self learning algorithm as implemented in the work of Tran et al[62] is described in the section 4.3.1. From this point onwards, we will refer to the traditional active-self learning strategy by ASL, and we will refer to our proposed algorithm as DASL.

4.3.1 Traditional active-self learning

The ASL algorithm is an extension of the generic active learning algorithm presented in Algorithm 1. The main difference is that one additional step is added where the trained model is allowed to automatically annotate the unlabeled samples for which it has a confidence greater than a predefined threshold. Algorithm 2 describes the active-self learning procedure.

Algorithm 2 Traditional active-self learning algorithm

1: **procedure** ASL(L, U, m, Q, S, $min\_confidence$)
2: $m \leftarrow$ Train(m, L, $epoch$)
3: **while** $\frac{|L|}{|L|+|U|} < S$ **do**
4: $L \leftarrow$ Self_label(U, m, $min\_confidence$)
5: $u \leftarrow$ Query(U, m, Q)
6: $L \leftarrow$ Oracle(u)
7: $m \leftarrow$ Train(m, L)

In order to demonstrate the sensitivity of this ASL algorithm, we will create an additional hyperparameter that indicates how big the labeled set must be for the trained model to start the self-labeling process. This means that we will effectively only allow the oracle to annotate unlabeled samples, thus effectively returning to a purely active learning setting, until a predefined percentage of the whole training set has been annotated. Only after this percentage of the training set has been annotated by the oracle we will allow for the model to start labeling samples automatically. In order to investigate the sensitivity, we will evaluate the algorithm with the self-learning procedure starting at: 1%, 5%, 10% and 15% of the whole training set being annotated by the oracle.

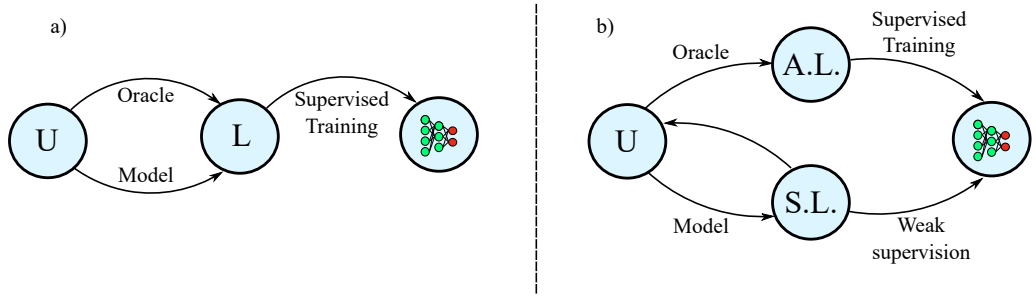

Figure 4.1: Diagrams of the proposed Active-Self learning methods. Note that U stands for unlabeled set, L represents the labeled set; $A.L.$ stands for active labeled, which is the set with samples labeled manually by a human annotator; $S.L.$ stands for self labeled, which is the set with samples automatically annotated by the trained model. Figure a) represents the ASL algorithm proposed in the literature, while Figure b) describes our proposed algorithm.

The minimum confidence for the model to self-label an unlabeled sample was selected to be 0.99, with the maximum confidence of the model being 1.0. And the remaining parameters were the same used in the experiment 1, which were presented in section 4.2.1. we will use the f1-score on the validation set with a patience of 5 epochs as the early stopping criterion for the experiments with the traditional ASL algorithm. we will compare the evolution of the performance of the model trained in each iteration of the ASL process, as well as the number of wrongly labeled tokens.

4.3.2 Proposed deep active-self learning

Due to this sensitivity of the ASL algorithm from the literature to the initial set of labeled samples, we propose a few changes to it in hopes of both reducing this sensitivity, and increasing the model's performance. Figure 4.1 shows the differences between the traditional ASL algorithm and our proposed version after a few changes.

Our proposed algorithm comes as an extension to the traditional ASL process described in section 4.3.1. The main changes are that we now separate the samples labeled by the oracle from those labeled automatically by the trained model, and at the end of the ASL iteration, the self-labeled samples are returned to the unlabeled set. The motivation behind these changes comes from semi-supervised learning, in particular the work from Clark et al[14], where self-training is employed concurrently to supervised training. Semi-supervised methods that employ self-training use unlabeled samples to increase the performance of a model that is also trained on a sufficiently big labeled set. This might not be the case for active-self learning algorithms, especially in earlier rounds where little labeled data is available and the manually labeled set is most definitely not a good representation of the whole dataset. Because of that, the proposed algorithm selects un-

labeled samples with high-confidence and that can be used, more reliably, for self-training but returns them to the unlabeled set after training the model. This potentially avoids introducing permanent bias to the ASL process. we also use a smaller learning rate when training using the self-labeled samples (denoted as *weak supervision* in Figure 4.1). By assigning a smaller learning rate for self-labeled samples, and by returning them back to the unlabeled set after each ASL iteration, the machine learning model is expected to be less influenced from wrongly labeled samples in earlier rounds of the active-self learning process.

The learning rate to be used during self-training was selected to be one tenth of the original learning rate. This choice is motivated by the fact that the self-labeled samples are assumed to be noisy, and we don't wish for them to be able to make as big of an impact on the update of the model's weights when compared to human annotated samples. we will use the DUTE strategy as the early stopping criterion during the experiments with our proposed DASL algorithm. Also, during training we alternate training on human labeled and self-labeled mini-batches, in a setup similar to that used by the semi-supervised algorithm proposed by Clark et al[14]. we will compare the performance of the trained model, along with the number of wrongly labeled tokens at each iteration of the proposed DASL process.

4.4 Experiment 3: Comparison of self-learning techniques

The third experiment comes to further evaluate our DASL algorithm introduced in section 4.3.2. The Experiment 3 compares how different self-training techniques impact the performance of the trained models. we will evaluate three self-training techniques, along with the weak supervision presented in Experiment 2. All self-training techniques to be evaluated rely on soft-targets for each token generated by the model. The CNN-biLSTM-CRF model is unable to generate token-level soft-targets as the CRF layer generates a distribution over label sequences instead. Because of that, we will use the output of the biLSTM layer of this model, along with a softmax layer to generate the distribution over tokens.

The first self-training technique to be evaluated is the word dropout, where we mask random tokens from the input sentences with special $< UNK >$ tokens and try to predict their classes based solely on their surrounding context. We use a word dropout probability of 50%, meaning approximately half of the tokens are substituted.

The second self-training technique to be tested is the cross-view training technique. This technique was developed to be used in neural models that are able to hide part of the

context from the input sentences, such as models with biLSTM encoders. Thus, it is not possible to apply this technique to the CNN-CNN-LSTM model, and we will report the results of this particular self-training technique only for the CNN-biLSTM-CRF model. Similarly to the original work by Clark et al[14], we will alternate training on mini-batches of labeled and unlabeled samples. The learning rate will be the same for both labeled and unlabeled mini-batches.

The third technique will be the virtual adversarial training[40]. Similarly to the original work, we will not alternate on labeled and unlabeled samples, but instead we will minimize the sum of both loss functions (i.e. supervised loss and VAT loss). Through experiments using the validation set, we selected the ϵ (perturbation norm) to be 0.01 and λ_{VAT} to be 1.

The experiments on the DASL algorithm with these three self-training techniques will use the f1-score on the validation set as an early stopping criterion, in order to identify the true upper bound of the trained model's performance. The baseline for these algorithms will be the deep active learning algorithm proposed by Shen et al[57], and introduced in section 4.2.1 using the f1-score for early stopping. we will use the same patience of 5 for the experiments with the Word dropout technique, but we will increase the patience to 15 for the experiments on the VAT and CVT techniques, as initial experiments have shown that these techniques have a more difficult time to generalize, and with a small patience they tend to stop training very early which results in a significant decrease in the performance of the trained model.

4.5 Experiment 4: Token level self-labeling

The experiments 2 and 3 presented previously relied on identifying whole unlabeled sentences that could be reliably used for self-training. This implied that the model's confidence on its predictions for all tokens in the sentences selected for self-training was high, which may not be the case. We now change the previous setup which identified whole unlabeled sentences to be self-labeled, and turn to identifying tokens that can be self-labeled reliably. Because of this change, we don't use our proposed DASL algorithm. Instead, we will use a modified version of the DAL algorithm proposed by Shen et al[57] and described in more details in section 4.2.1. The single difference between the original and our modified DAL algorithm is in its labeling process. The original algorithm queried the most uncertain samples and asked the oracle to annotate them. Our modified algorithm, however, identifies the tokens with low confidence and asks the oracle to annotate them. The remaining unlabeled tokens from the query are self-labeled by the model, as the model's confidence for them is high.

Identifying high-confidence tokens

Suppose our neural model outputs a prediction distribution $p \in \mathcal{R}^C$, with C being the number of possible classes, where each dimension indicates the likelihood of a particular class for the input token. For this work, a high confidence of the model for a token implies that the most likely predicted class has a probability higher than a predefined threshold. We will use a threshold confidence of 99%, implying that tokens that have less than 99% of confidence in their predicted class will be labeled by the oracle, while the remaining tokens will be self-labeled by the model.

In order to identify the high-confidence tokens, the model needs to output a distribution over classes for each token. For models with CRF classification layers, such as the CNN-biLSTM-CRF, this is not possible because the CRF produces a distribution over entire sequences. Therefore, for this fourth experiment only the CNN-CNN-LSTM model will be used.

Self-labeling with refinement of predictions

After the low confidence tokens have been labeled by the oracle, we need to self-label the remaining unlabeled tokens from the queried samples. A naive implementation of this self-labeling process is to predict the classes for all tokens and use the predictions to annotate high-confidence tokens. however, we now have the labels of the non high-confidence tokens, that were annotated by the oracle, and we should leverage them to refine the predictions. For instance, the CNN-CNN-LSTM model classifies tokens using a greedy decoding approach, where it uses the predicted label of the previous token in order to predict the class of the current token. We modify the greedy decoding of this model in the following manner: If the previous token had lower-confidence, we feed the model with the the human annotated label, instead of the previous predicted label.

This refinement step was inspired by the iterative refinement algorithm, proposed by Park et al [46]. The original work use the iterative refinement to identify synonyms to substitute specific tokens from a sentence, with low impact on the sentence's coherence. The idea is to identify potential synonyms for specific tokens, replace them and verify if a masked language model (MLM) predicts the synonyms with high confidence. The synonyms with the lowest MLM scores are replaced by other synonyms. This process is repeated a predefined number of times, hence iterative, with a decrease in the number of tokens that are replaced in each iteration. For our refinement step, however, we use the reliable human annotated tokens to enhance the model predictions for the unlabeled tokens.

Baseline and comparisons

The baseline for the fourth experiment will be the original DAL algorithm, proposed by shen et al[57] and introduced in Section 4.2.1. We'll use the early stopping technique that rely on the f1-score on the validation set. We will compare our modified DAL algorithm against the baseline. It is expected that our modified algorithm requires less human resources.

We will also evaluate our modified algorithm with our proposed DUTE strategy. We'll compare it to modified algorithm that uses the f1-score early stopping technique.

All algorithms will be simulated until 50% of the whole training set has been annotated, either by the oracle or by self-labeling. All hyperparameters will be the same as those used in experiment 1 presented in Section 4.2.

Chapter 5

Results

This chapter presents the results for the four experiments proposed in Chapter 4. Section 5.1 presents the results for the experiment 1, a study on the impact of different early stopping techniques on a deep active learning algorithm. Section 5.2 presents the results for the second experiment, an analysis of the sensitivity of active-self learning algorithms from both the literature and our original proposal. Section 5.3 presents a study on how our proposed DASL algorithm behaves when implemented with different self-training techniques. Section 5.4 presents a comparison of the token level self-labeling process, indicating the reduction in human annotation and the impact on the trained model's performance. All performed experiments were simulated four times, with the same initial labeled sets, and the results reported in the following sections are the mean of these four runs.

5.1 Experiment 1

The experiment 1 compares the impact of different early stopping techniques on the performance of a model trained through active learning. Figure 5.1 presents a comparison between the 5 early stopping techniques, including our DUTE strategy. From the results, we note that the model trained with our proposed strategy has performance similar to those trained using traditional techniques that rely on validation sets. We also note that the BGD and the consecutive BGD perform consistently worse than the other techniques in earlier rounds. This behaviour may be explained by the fact that the BGD technique was proposed and tested for feedforward neural models, while we employ recurrent neural models. The gradient of a recurrent neural model is heavily influenced by the length of the input sequence, and similarly in Shen et al[57], we arrange the sentences so that sentences that fall in the same batch have similar length. This causes the gradients computed on

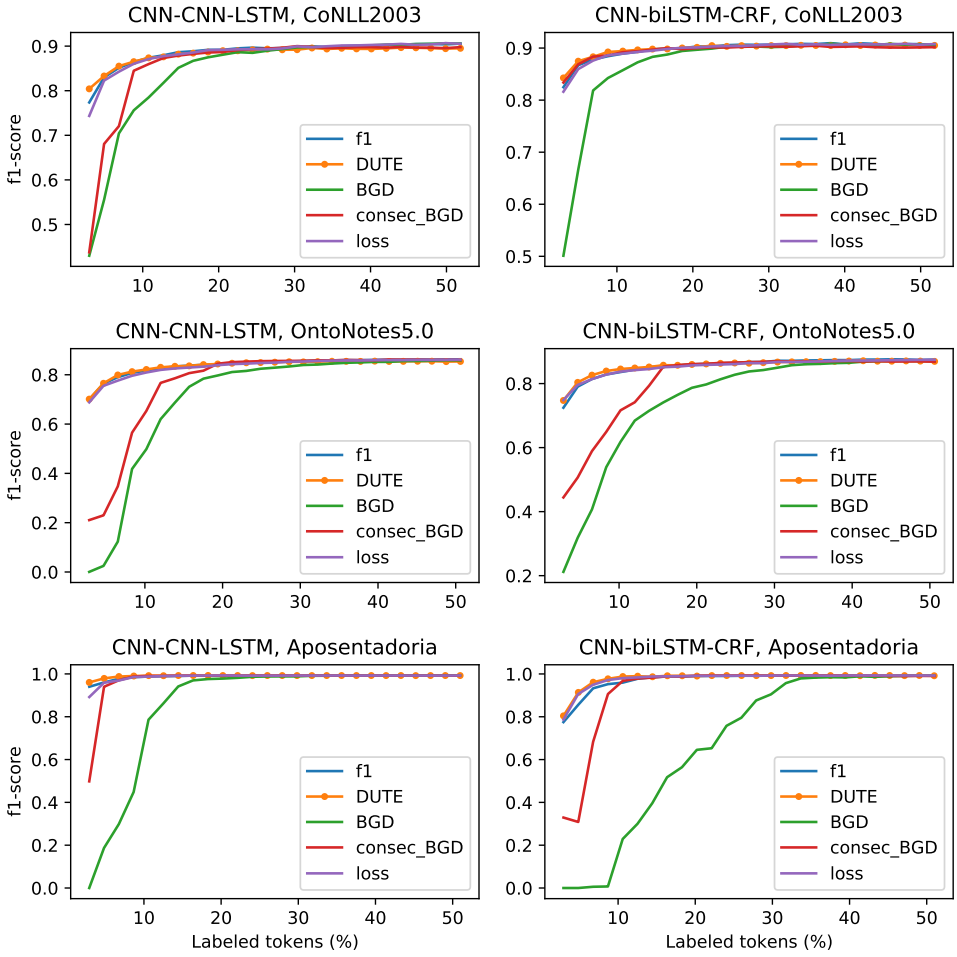

Figure 5.1: Performance of the models trained on an active learning setting with different early stopping criteria. In all graphs, *f1* indicates the early stopping based on the f1-score computed on the validation set, *loss* indicates the ES criterion based on the validation set loss, *BGD* is the batch gradient descent technique, *consec_BGD* is the BGD technique that must increase over consecutive epochs, and *DUTE* is our proposed ES strategy

different mini-batches to not necessarily converge at the same speed, thus resulting in the BGD metric not accurately assessing the model's generalization for the early stopping.

Figure 5.2 shows the number of training epochs for each early stopping technique. We note that, while the DUTE strategy requires more epochs than the traditional techniques, they have have similar behaviour. More specifically, in earlier rounds of the active learning process, the model needs more epochs as the new data being labeled is highly informative. However, on later rounds newly labeled data are not as informative and the training becomes a fine-tuning process, requiring less epochs. Our proposed strategy captures this behaviour, without relying on a validation set.

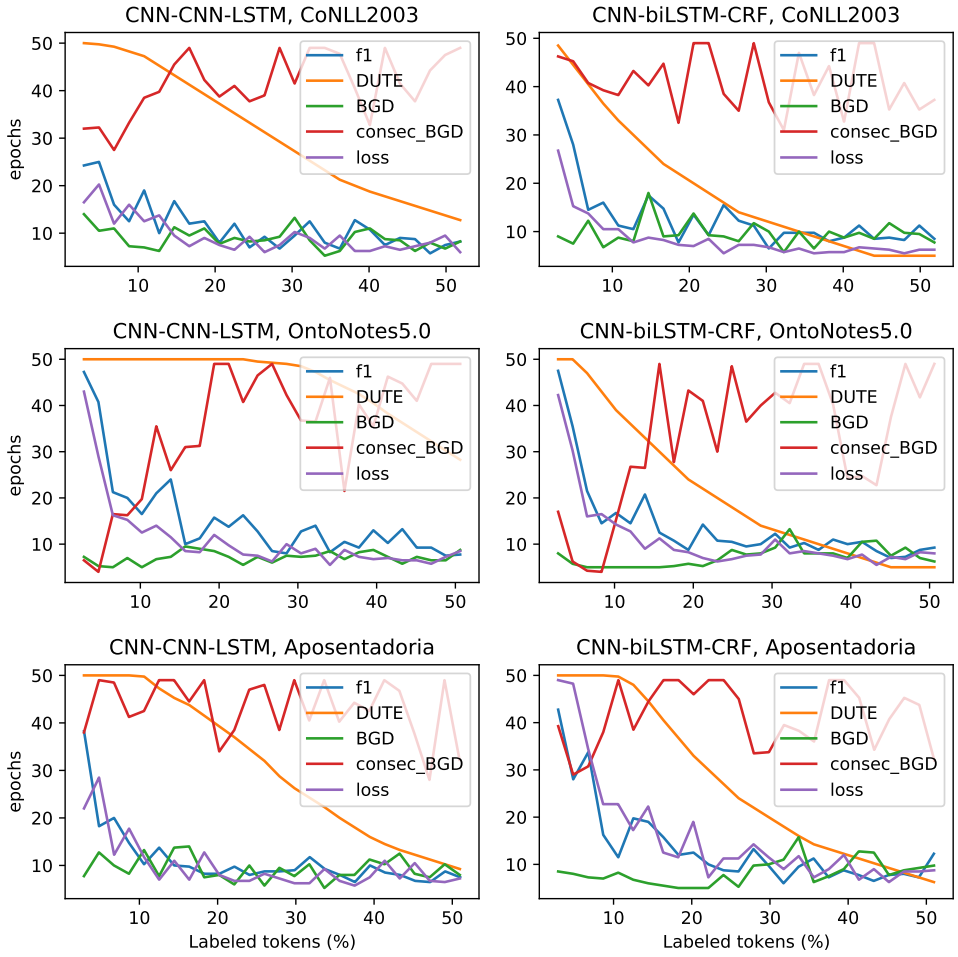

Figure 5.2: Number of training epochs (*y-axis*) by the labeled set size (*x-axis*), indicating the number of epochs needed to train the model as more labeled samples are annotated by the oracle. Similarly to Figure 5.1, *f1* indicates the early stopping based on the f1-score computed on the validation set, *loss* indicates the ES criterion based on the validation set loss, *BGD* is the batch gradient descent technique, *consec_BGD* is the BGD technique that must increase over consecutive epochs, and *DUTE* is our proposed ES strategy

5.2 Experiment 2

The sensitivity of the traditional active-self learning algorithm is demonstrated in this section. Figure 5.3 presents a comparison on the number of wrongly labeled tokens by both traditional and our proposed active-self learning algorithms. For the traditional algorithm, we evaluate the self-training sensitivity by waiting 1%, 5%, 10%, and 15% of the training set to be hand annotated by the oracle before the self-training process starts. As expected from the traditional ASL, we note that the amount of mislabeled tokens decrease

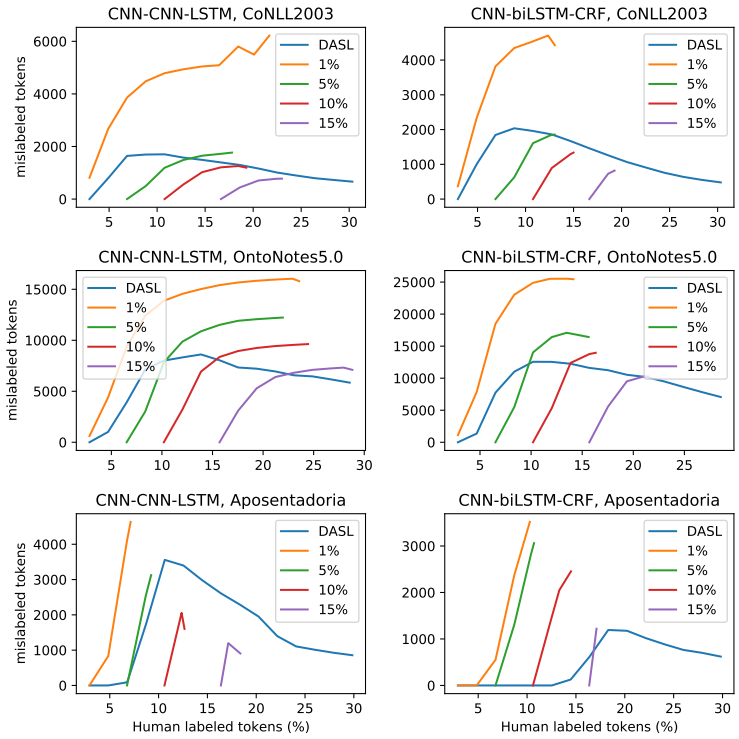

Figure 5.3: Number of wrongly self-labeled tokens by the traditional and proposed active-self learning algorithms. The percentages on the legend represent the traditional ASL algorithm with the self-learning process starting when this percentage of the training set has been labeled by the oracle. DASL represents our proposed algorithm.

the longer the self-learning process waits to start. We also observe that our proposed algorithm is more robust, as it consistently mislabels less tokens than the traditional algorithm, while not requiring to wait for a significant percentage of the dataset to be annotated by the oracle. Figure 5.4 presents the impact of the traditional *ASL* algorithm on the performance of the trained models, when the self-training process waits for 1%, 5%, 10%, and 15% of the training set to be annotated by the oracle. The performance of our proposed algorithm is also reported as *DASL*. We note that our algorithm outperforms the traditional algorithm for most of the experiments. The exceptions happen on the *Aposentadoria* dataset, which due to its simplicity, can be reliably solved by the traditional ASL.

5.3 Experiment 3

The third experiment compares the performance of a model trained by our proposed DASL algorithm, but employing different self-training techniques, against the DAL algorithm proposed by Shen et al[57]. The Figure 5.5 presents the curves from the simulations.

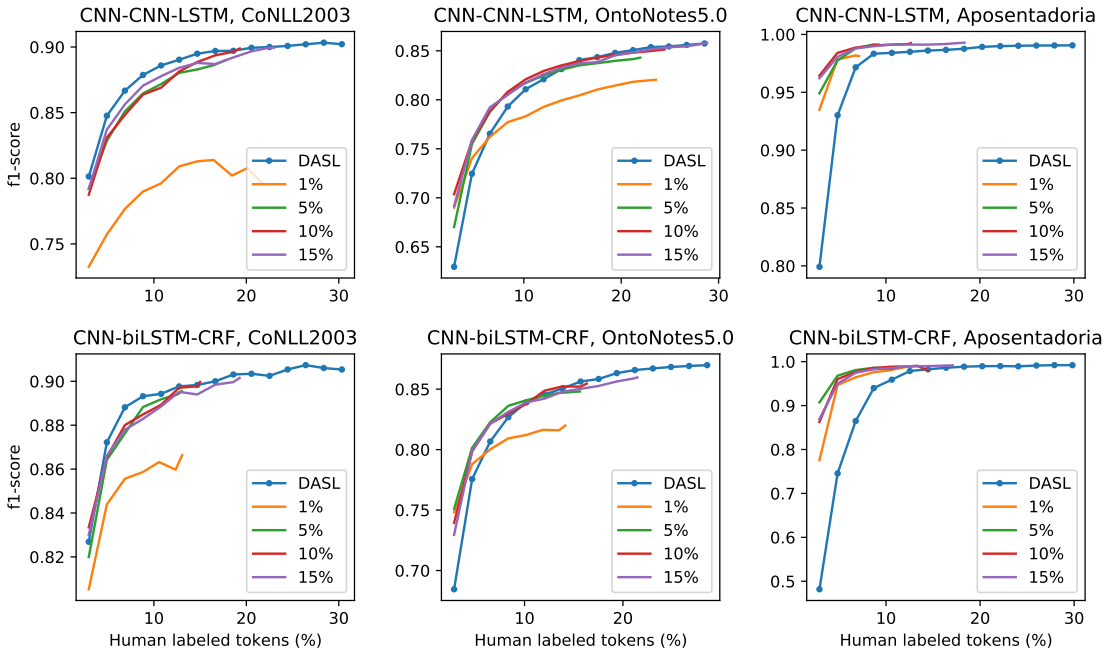

Figure 5.4: Performance of the models trained by the traditional and proposed active-self learning algorithms. The percentages on the legend represent the traditional ASL algorithm with the self-learning process starting when this percentage of the training set has been labeled by the oracle. DASL represents our proposed algorithm.

We observe that, in general, the more sophisticated self-training techniques are able to slightly outperform the DAL baseline on the earlier rounds, when less labeled data is available. This slight advantage, however, does not hold as more data is labeled by the oracle. In fact, we observe that when 20% of the data has been labeled by the oracle, different self-training techniques stagnate, or do not improve as much as expected. This has been observed for the VAT technique on the CoNLL2003 dataset when training the CNN-CNN-LSTM model. Similarly, the CVT technique stagnates on the CoNLL2003 and OntoNotes5.0 for the CNN-biLSTM-CRF model.

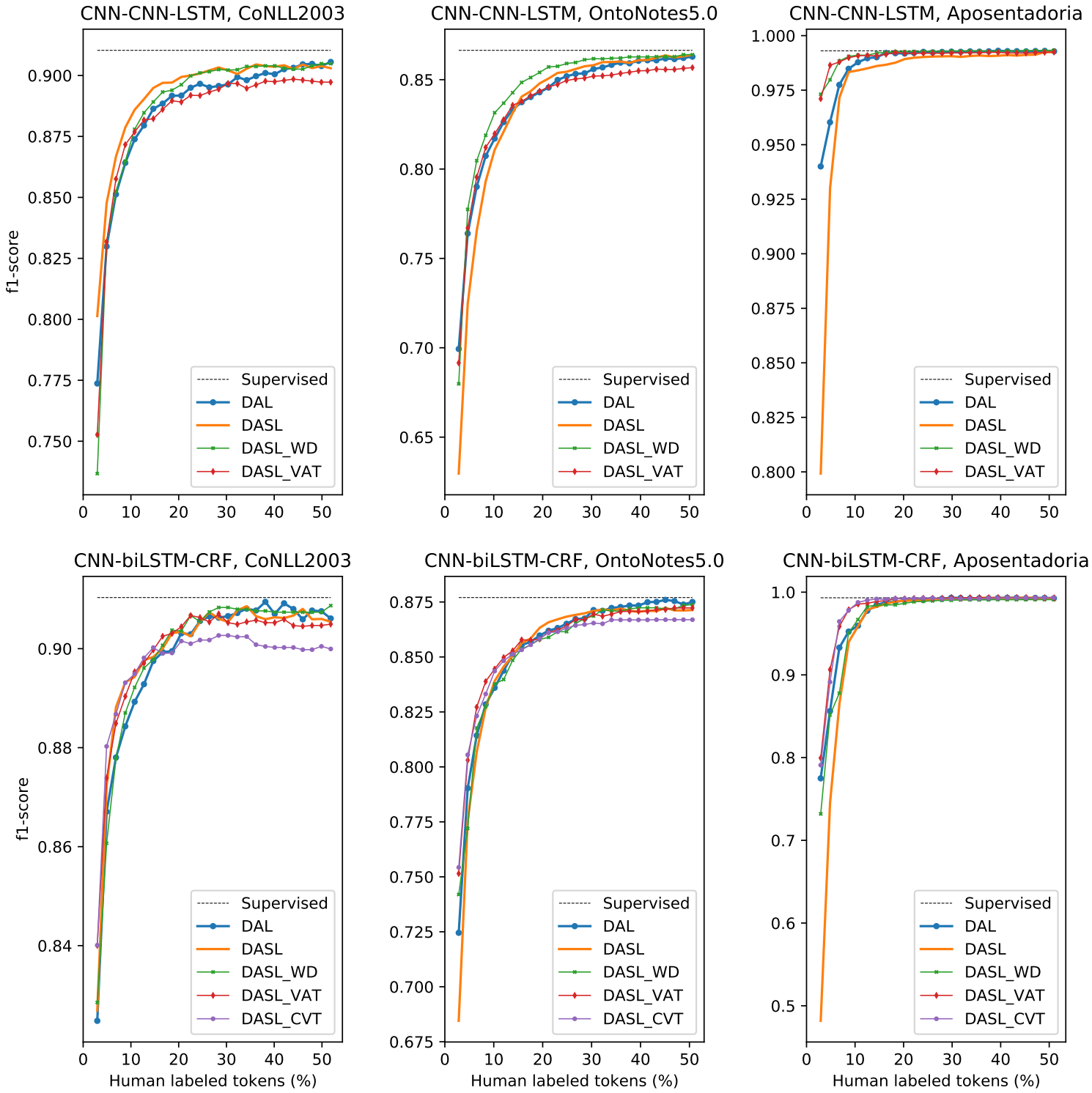

Figure 5.5: Performance of the proposed DASL algorithm with different self-training techniques (*y-axis*) by the percentage of hand annotated tokens (*x-axis*). In the graphs, WD stands for word dropout, VAT is the virtual adversarial training, and CVT is the cross-view training

Table 5.1: Percentage of the training set that was annotated by the oracle in order to train a model that reaches 99% of its peak performance. DAL indicates the algorithm from the literature, while MDAL is our modified algorithm with token level self-labeling. Reduction indicates the percentage of tokens from the training set that weren't required to be hand annotated by the oracle for the model to reach almost peak performance.

Dataset	CoNLL2003	OntoNotes5.0	Aposentadoria
DAL	36.20%	25.91%	8.68%
MDAL	6.96%	11.54%	4.73%
Reduction	29.24%	14.37%	3.95%

5.4 Experiment 4

The fourth experiment compares the modified DAL algorithm with token-level self-labeling, against the baseline that is the original DAL algorithm proposed by Shen et al[57]. The Figure 5.6 compares the original and the modified DAL algorithms, both using the validation f1-score for early stopping. The graphs to the left on Figure 5.6 show that both the original and the modified DAL algorithms achieve similar performance with the same amount of labeled data, but with our modified algorithm allowing for tokens to be self-labeled. From the graphs in the center, we observe that the models trained with the modified DAL algorithm reach peak performance with significantly less hand annotated tokens. This is justified by the fact that for the modified algorithm, most tokens are self-annotated by the trained model reliably, as shown in the graphs to the right in Figure 5.6.

The Figure 5.7 compares the modified algorithm using the DUTE strategy to the algorithm with early stopping based on the f1-score computed on the validation set. We observe that the performance of the trained model with the DUTE strategy has a faster increase, but may not reach the model's peak performance. We also observe, from the graphs to the right, that the model trained with the DUTE strategy mislabels tokens more often. The results show that the DUTE strategy is outperformed by the early stopping using the f1-score on the validation set, but taking into consideration the fact that it doesn't rely on a validation set and that it still achieves good performance, it can be reliably implemented in practice in low-resource active learning scenarios.

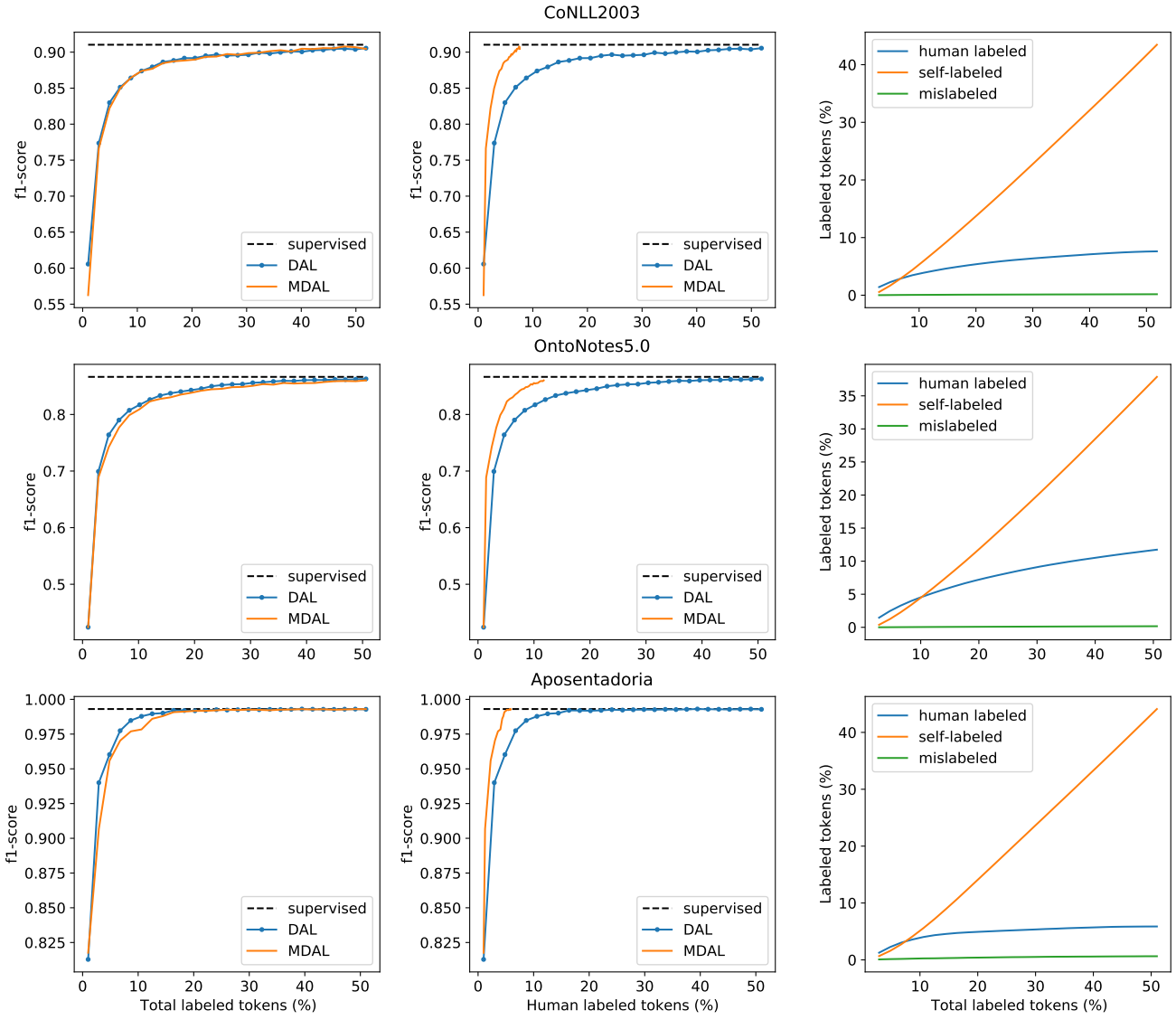

Figure 5.6: Comparison between the original and modified DAL algorithms. Graphs on the left column compare the performance of the trained model (*y-axis*), by the total amount of labeled data (*x-axis*), including tokens annotated by the oracle and self-annotated by the trained model in the case of our modified algorithm. Graphs on the center column compare the f1-score of the trained model (*y-axis*) by the amount of data annotated by the oracle (*x-axis*). Graphs on the right column compare the percentage of the training set that was annotated by a human, by the model through self-labeling, and the samples that were mislabeled. In all graphs, DAL represents the original deep active learning algorithm from the literature, while MDAL indicates our modified algorithm with token level self-labeling.

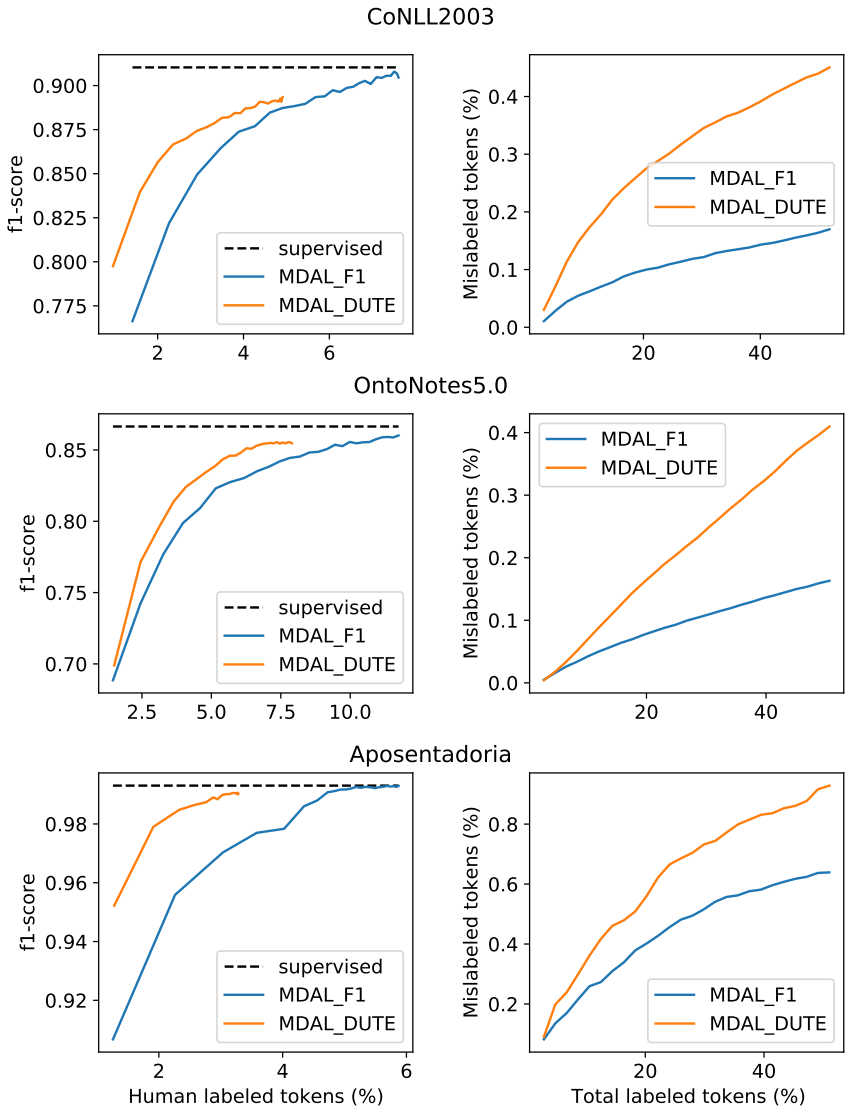

Figure 5.7: Comparison between the modified DAL algorithms with different early stopping techniques. The graphs to the left compare the performance of the trained models (*y-axis*), by the quantity of tokens labeled by the oracle (*x-axis*). MDAL_F1 represents the model trained with early stopping based on validation f1-score, while MDAL_DUTE represents the model trained with our DUTE strategy. The graphs to the right compare the amount of mislabeled tokens caused by each early stopping technique.

Chapter 6

Conclusion and future work

From the first experiment, we noticed that our proposed DUTE strategy for early stopping of active learning-based algorithms is competitive with traditional techniques that rely on validation sets, while being better than other early stopping criteria that don't rely on validation sets. One drawback of our method is that it cannot identify overfitting of the trained model.

The second experiment demonstrated that the proposed active-self learning algorithm is robust to the initial set of labeled samples. When compared to the traditional active-self learning algorithm, our achieves higher performance in more challenging datasets. It can also be used to help annotate entire datasets, albeit not with gold labels, but more reliably than previous ASL algorithms.

Contrary to expectations, the third experiment has shown that using more sophisticated techniques did not improve the model's performance in a significant manner. Instead, we have observed that the pseudo-label technique employed on our ASL algorithm proposed in section 4.3.2 often outperforms more sophisticated techniques, and sometimes even slightly outperforms the DAL baseline.

The fourth experiment shows that the change of self-labeling paradigm, from annotating whole sentences to a token level annotation, resulted in a considerable reduction of human annotation costs. Our modified algorithm significantly outperforms the algorithm from the literature proposed by Shen et al[57], by allowing the machine learning model to reach near peak performance with up to 29.24% less human annotated tokens on the CoNLL2003 dataset. Our proposed algorithm also outperforms the subsequence-based active learning algorithm proposed by Radmard et al[50], presented in Section 3.1.1, as our algorithm trains a model to near peak performance using 6.96% and 11.54% of the training set (hand annotated) against the 27% and 13% required for their algorithm, on the datasets CoNLL2003 and OntoNotes5.0 respectively. The results reported for the subsequence-based algorithm are those presented in the original work by Radmard et

al[50], as we did not replicate it.

From the experiments performed, we observed that sentence level self-labeling did not achieve the expected results, as it wasn't able to improve the model's performance or reduce annotation costs when compared to current works on DAL from the literature. Our modified deep active learning algorithm with token level self-labeling, however, did outperform previous algorithms from the literature, as it allowed us to train a high performing model with a significant decrease in hand annotated data. It was also shown that our modified algorithm coupled with the DUTE strategy is a viable option for training good NER models in very low resource settings, as it considerably reduces annotation costs from the active learning algorithm while not requiring a validation set for early stopping.

Future directions

Most works on active learning propose new sampling functions as shown in chapter 3. These works focus on improving the convergence rates of AL, allowing the model to reach peak performance with the least amount of labeled data. However, active learning algorithms suffer from a series of practical implementation issues, that are most of the time not addressed. One of the most serious problems is hyperparameter tuning. In real-world scenarios, active learning algorithms are one-shot, where we can label one subset of samples from the whole dataset expecting it to be representative of the whole set's distribution. If the subset is not representative of the whole, the effort to annotate has been spent and cannot be reused. Therefore, areas of research such as autoML and unsupervised hyperparameter tuning are closely related to the practical implementation of active learning algorithms. Neural architecture search is also closely related to this problem [19].

One other direction is to delve into the theory of model generalization to propose more efficient early stopping techniques that don't rely on validation sets. The identification of when or why a model generalizes has been a hot-topic of research[17], but this information hasn't been translated into reliable early stopping techniques.

The experimentation on token-level self-labeling may also be improved. One experiment that could not be performed due to the dissertation deadline is the iterative refinement of predictions. After identifying the low-confidence tokens, and having them labeled, we can reassess whether the confidence of the high-confidence tokens (that would be self-labeled) has changed. If a token that was previously classified with high-confidence now has a low-confidence, it can be hand annotated. This iterative approach of using newly labeled tokens to reassess the confidence of the other tokens may reduce the number of tokens being wrongly self-labeled.

References

[1] Akbik, A., Blythe, D., and Vollgraf, R. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics* (Santa Fe, New Mexico, USA, Aug. 2018), Association for Computational Linguistics, pp. 1638–1649. 11

[2] Arthur, D., and Vassilvitskii, S. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (USA, 2007), SODA, Society for Industrial and Applied Mathematics, p. 1027–1035. 20

[3] Babych, B., and Hartley, A. Improving machine translation quality with automatic named entity recognition. In *Proceedings of the 7th International EAMT workshop on MT and other language technology tools, Improving MT through other language technology tools, Resource and tools for building MT at EACL* (2003). 5

[4] Baevski, A., Edunov, S., Liu, Y., Zettlemoyer, L., and Auli, M. Cloze-driven pretraining of self-attention networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (Hong Kong, China, Nov. 2019), Association for Computational Linguistics, pp. 5360–5369. 1

[5] Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. *arxiv preprint* (2016). 15

[6] Barrow, J., Jain, R., Morariu, V., Manjunatha, V., Oard, D., and Resnik, P. A joint model for document segmentation and segment labeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (Online, July 2020), Association for Computational Linguistics, pp. 313–322. 1

[7] Blum, A., and Mitchell, T. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory* (New York, NY, USA, 1998), COLT, Association for Computing Machinery, p. 92–100. 29

[8] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics 5* (2017), 135–146. 11

[9] C. S. A. V. S. Neto, J. R., and de Paulo Faleiros, T. Deep active-self learning applied to named entity recognition. In *Intelligent systems* (Online, Nov. 2021), BRACIS, Springer International Publishing. 4

[10] Chen, J., Wang, Z., Tian, R., Yang, Z., and Yang, D. Local additivity based data augmentation for semi-supervised NER. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (Online, Nov. 2020), Association for Computational Linguistics, pp. 1241–1251. 31

[11] Chen, L., Garcia, F., Kumar, V., Xie, H., and Lu, J. Industry scale semi-supervised learning for natural language understanding. *arxiv preprint* (2021). 31

[12] Chen, L., Ruan, W., Liu, X., and Lu, J. SeqVAT: Virtual adversarial training for semi-supervised sequence labeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (Online, July 2020), Association for Computational Linguistics, pp. 8801–8811. 32

[13] Chen, Y., Lasko, T. A., Mei, Q., Denny, J. C., and Xu, H. A study of active learning methods for named entity recognition in clinical text. *Journal of Biomedical Informatics 58* (2015), 11 – 18. 24

[14] Clark, K., Luong, M.-T., Manning, C. D., and Le, Q. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (Brussels, Belgium, Oct.-Nov. 2018), Association for Computational Linguistics, pp. 1914–1925. vii, 1, 22, 29, 30, 31, 32, 35, 43, 44, 45

[15] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (Minneapolis, Minnesota, June 2019), Association for Computational Linguistics, pp. 4171–4186. 11, 32

[16] Ertekin, S., Huang, J., Bottou, L., and Giles, L. Learning on the border: Active learning in imbalanced data classification. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management* (New York, NY, USA, 2007), CIKM, Association for Computing Machinery, p. 127–136. 28

[17] Forouzesh, M., and Thiran, P. Disparity between batches as a signal for early stopping. *arxiv preprint* (2021). 18, 41, 58

[18] Gao, N., Karampatziakis, N., Potharaju, R., and Cucerzan, S. Active entity recognition in low resource settings. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (New York, NY, USA, 2019), CIKM, Association for Computing Machinery, p. 2261–2264. 7, 20, 24

[19] Geifman, Y., and El-Yaniv, R. Deep active learning with a neural architecture search. In *Advances in Neural Information Processing Systems (NeurIPS)* (2019),

H. Wallach, H. Larochelle, A. Beygelzimer, E. Fox, and R. Garnett, Eds., vol. 32, Curran Associates, Inc. 58

[20] Ghayoomi, M. Using variance as a stopping criterion for active learning of frame assignment. In *Proceedings of the NAACL HLT shop on Active Learning for Natural Language Processing* (2010), Association for Computational Linguistics, pp. 1–9. 27

[21] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`. 17

[22] Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arxiv preprint* (2015). 31

[23] Grishman, R., and Sundheim, B. Message understanding conference- 6: A brief history. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics* (1996). 5

[24] Hartmann, N. S., Fonseca, E. R., Shulby, C. D., Treviso, M. V., Rodrigues, J. S., and Aluísio, S. M. Portuguese word embeddings: Evaluating on word analogies and natural language tasks. In *Anais do XI Simpósio Brasileiro de Tecnologia da Informação e da Linguagem Humana* (Porto Alegre, RS, Brasil, 2017), SBC, pp. 122–131. 36

[25] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 770–778. 14

[26] Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arxiv preprint* (2015). 22

[27] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural Comput. 9*, 8 (Nov. 1997), 1735–1780. 14

[28] Houlsby, N., Huszár, F., Ghahramani, Z., and Lengyel, M. Bayesian active learning for classification and preference learning. *arxiv preprint* (2011). 25

[29] Hsu, W.-N., and Lin, H.-T. Active learning by learning. *Proceedings of the AAAI Conference on Artificial Intelligence 29*, 1 (Feb. 2015). 19

[30] Huang, L., and Chiang, D. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology* (Vancouver, British Columbia, Oct. 2005), Association for Computational Linguistics, pp. 53–64. 32

[31] Kobayashi, K., and Wakabayashi, K. Named entity recognition using point prediction and active learning. In *Proceedings of the 21st International Conference on Information Integration and Web-Based Applications and Services* (New York, NY, USA, 2019), iiWAS, Association for Computing Machinery, p. 287–293. 7, 25

[32] Lakshmi Narayan, P., Nagesh, A., and Surdeanu, M. Exploration of noise strategies in semi-supervised named entity classification. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM)* (Minneapolis, Minnesota, June 2019), Association for Computational Linguistics, pp. 186–191. 30, 31, 32

[33] Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. Neural architectures for named entity recognition. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (San Diego, California, June 2016), Association for Computational Linguistics, pp. 260–270. 35

[34] Laws, F., and Schütze, H. Stopping criteria for active learning of named entity recognition. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1* (USA, 2008), COLING, Association for Computational Linguistics, p. 465–472. 27, 28

[35] Lin, Y., Sun, C., Xiaolong, W., and Xuan, W. Combining self learning and active learning for chinese named entity recognition. *Journal of Software 5* (05 2010). 29

[36] Ma, X., and Hovy, E. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (Berlin, Germany, Aug. 2016), Association for Computational Linguistics, pp. 1064–1074. xvi, 35, 37, 38

[37] Mahsereci, M., Balles, L., Lassner, C., and Hennig, P. Early stopping without a validation set. *arxiv preprint* (2017). 41

[38] McAllester, D. Simplified pac-bayesian margin bounds. In *Learning Theory and Kernel Machines* (Berlin, Heidelberg, 2003), Springer Berlin Heidelberg, pp. 203–215. 18

[39] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2* (Red Hook, NY, USA, 2013), NIPS, Curran Associates Inc., p. 3111–3119. 11

[40] Miyato, T., Dai, A. M., and Goodfellow, I. Adversarial training methods for semi-supervised text classification. In *International Conference on Learning Representations (ICLR)* (2017). vii, 31, 32, 45

[41] Miyato, T., ichi Maeda, S., Koyama, M., Nakae, K., and Ishii, S. Distributional smoothing with virtual adversarial training. *arxiv preprint* (2016). 31, 32

[42] Mollá, D., van Zaanen, M., and Smith, D. Named entity recognition for question answering. In *Proceedings of the Australasian Language Technology Workshop* (Sydney, Australia, Nov. 2006), pp. 51–58. 5

[43] Nair, V., and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning* (Madison, WI, USA, 2010), ICML, Omnipress, p. 807–814. 29, 36

[44] Neubig, G., Nakata, Y., and Mori, S. Pointwise prediction for robust, adaptable Japanese morphological analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (Portland, Oregon, USA, June 2011), Association for Computational Linguistics, pp. 529–533. 25

[45] Nobata, C., Sekine, S., Isahara, H., and Grishman, R. Summarization system integrated with named entity tagging and IE pattern discovery. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC)* (Las Palmas, Canary Islands - Spain, May 2002), European Language Resources Association (ELRA). 5

[46] Park, J., Kim, G., and Kang, J. Consistency training with virtual adversarial discrete perturbation. *arxiv preprint* (2021). 32, 46

[47] Pennington, J., Socher, R., and Manning, C. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (Doha, Qatar, Oct. 2014), Association for Computational Linguistics, pp. 1532–1543. 11, 36

[48] Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (New Orleans, Louisiana, June 2018), Association for Computational Linguistics, pp. 2227–2237. 11

[49] Pradhan, S., Moschitti, A., Xue, N., Ng, H. T., Björkelund, A., Uryupina, O., Zhang, Y., and Zhong, Z. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning* (Sofia, Bulgaria, Aug. 2013), Association for Computational Linguistics, pp. 143–152. 1, 10, 25

[50] Radmard, P., Fathullah, Y., and Lipani, A. Subsequence based deep active learning for named entity recognition. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (Online, Aug. 2021), Association for Computational Linguistics, pp. 4310–4321. 26, 27, 33, 57, 58

[51] Ratinov, L., and Roth, D. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL)* (Boulder, Colorado, June 2009), Association for Computational Linguistics, pp. 147–155. 35

[52] Sang, E. F. T. K. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *Proceedings of the 6th Conference on Natural Language Learning - Volume 20* (USA, 2002), COLING, Association for Computational Linguistics, p. 1–4. 7

[53] Sang, E. F. T. K., and Meulder, F. D. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL* (2003), pp. 142–147. 1, 10

[54] Schohn, G., and Cohn, D. Less is more: Active learning with support vector machines. In *Proceedings of the Seventeenth International Conference on Machine Learning* (San Francisco, CA, USA, 2000), ICML, Morgan Kaufmann Publishers Inc., p. 839–846. 28

[55] Settles, B., and Craven, M. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (USA, 2008), EMNLP, Association for Computational Linguistics, p. 1070–1079. 1, 24

[56] Shen, D., Zhang, J., Su, J., Zhou, G., and Tan, C.-L. Multi-criteria-based active learning for named entity recognition. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)* (Barcelona, Spain, July 2004), pp. 589–596. 7, 24

[57] Shen, Y., Yun, H., Lipton, Z., Kronrod, Y., and Anandkumar, A. Deep active learning for named entity recognition. In *Proceedings of the 2nd Workshop on Representation Learning for NLP* (Vancouver, Canada, Aug. 2017), Association for Computational Linguistics, pp. 252–256. xvi, 1, 13, 25, 26, 27, 30, 35, 37, 38, 39, 40, 45, 47, 48, 51, 54, 57

[58] Siddhant, A., and Lipton, Z. C. Deep Bayesian active learning for natural language processing: Results of a large-scale empirical study. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (Brussels, Belgium, Oct.-Nov. 2018), Association for Computational Linguistics, pp. 2904–2909. 25, 27, 35, 36, 37, 38, 39, 40

[59] Srihari, R., and Li, W. A question answering system supported by information extraction. In *Proceedings of the Sixth Conference on Applied Natural Language Processing* (USA, 2000), ANLC, Association for Computational Linguistics, p. 166–172. 5

[60] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res. 15*, 1 (Jan. 2014), 1929–1958. 35

[61] Sutton, C., and McCallum, A. An introduction to conditional random fields. *Found. Trends Mach. Learn. 4*, 4 (Apr. 2012), 267–373. 8, 10

[62] Tran, V. C., Nguyen, N. T., Fujita, H., Hoang, D. T., and Hwang, D. A combination of active learning and self-learning for named entity recognition on twitter using conditional random fields. *Knowledge-Based Systems 132* (2017), 179–187. 3, 24, 29, 34, 42

[63] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems* (2017), vol. 30, Curran Associates, Inc. xiii, 15, 16

[64] Vlachos, A. A stopping criterion for active learning. *Comput. Speech Lang. 22*, 3 (July 2008), 295–312. 28

[65] Wang, W., Cai, W., and Zhang, Y. Stability-based stopping criterion for active learning. In *2014 IEEE International Conference on Data Mining* (2014), pp. 1019–1024. 28

[66] Zheng, G., Mukherjee, S., Dong, X. L., and Li, F. Opentag: Open attribute value extraction from product profiles. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2018), KDD, Association for Computing Machinery, p. 1049–1058. 25, 27

[67] Zhu, J., Wang, H., Hovy, E., and Ma, M. Confidence-based stopping criteria for active learning for data annotation. *ACM Trans. Speech Lang. Process. 6*, 3 (Apr. 2010). 27, 28, 40