Relation Extraction

Multilingual Natural Language Processing HW3

Lorenzo Ciarpaglini

August 8, 2025

Abstract

This report explores the task of Relation Extraction (RE) in Natural Language Processing, a crucial step in understanding the semantic relationships between entities within a text. I adopted two distinct approaches to tackle this task: a two-phase pipeline and a table filling methodology.

1 Introduction

Relation Extraction (RE) is a fundamental task in Natural Language Processing that involves identifying semantic relationships between entities within a text. Given a sentence that includes two or more entities, the goal of RE is to detect the spans of these entities (i.e., the specific words that constitute an entity) and to determine the nature of the relationships between them. In the subsequent sections of this report, I will explore various approaches to tackle the RE task and I will analyze the effectiveness of these methods.

2 Methods

In this section, I outline the various approaches adopted for RE.

2.1 Table Filling

Given a sentence, construct a matrix A, with rows and columns corresponding to sentence tokens. For distinct tokens i and j, $A[i, j]$ specifies the relationship, where i is the subject and j the object, possibly indicating no-relation. An Example is reported in figure 1.

Subject \ Object	h_1: Joe	...	h_4: Apple	h_5: in	h_6: Cupertino
h_1		...	**employed-at**	no_relation	no_relation
...	...		...	...	...
h_4	**employer**	...		no_relation	**located-in**
h_5	no_relation	...	no_relation		no_relation
h_6	no_relation	...	no_relation	no_relation	

Figure 1: A matrix explaining the relationships between entities in the sentence.

The simpler method for deriving this matrix involves using a transformer model to generate embeddings for each token. The A matrix is then constructed by calculating the element-wise product of the embeddings for tokens i and j, thus defining $A[i, j]$. This approach leverages the nuanced semantic representations captured by transformer-based embeddings to infer the relationships between tokens. From now on I will refer to this first approach as the 'Baseline model'. A graphic explanation of this process is reported in figures 2, 3.

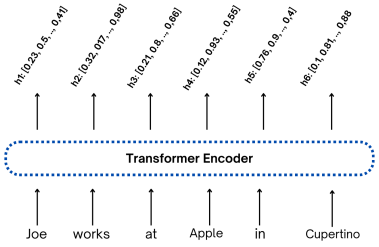


Figure 2: Embedding vector generated by the transformer encoder for each token.

	h1	h2	h3	h4	h5	h6
h1	h1 · h1	h1 · h2	...	...	...	h1 · h6
h2	h2 · h1	...	...	...	...	...
h3	...	...	...	...	...	...
h4	...	...	...	...	...	...
h5	...	...	...	...	...	...
h6	h6 · h1	...	...	...	...	h6 · h6

Figure 3: Matrix built by multiplying each token embedding vector with all the other embedding vectors.

An alternative approach involves concatenating the matrix generated through element-wise product with the attention matrix produced in the last layer of the transformer. Initially, I take the attention matrix, which matches the number of heads in the last layer, and perform a projection using a linear layer to match the dimensionality of the embeddings obtained from the transformer for each sentence token. This allows for the concatenation of the previously obtained matrix with the current one. Subsequently, in both methods, I apply two linear layers to reduce the size of the latent space and finally employ a classifier to determine the relationships. From now on I will refer to this second approach as the 'Mixed model'.

2.2 Two-phase pipeline

In the two-phase pipeline for Relation Extraction, the first step involves identifying token spans for each entity within a sentence using the IOB format, where "B" denotes the beginning of an entity, "I" represents internal tokens of the entity, and "O" indicates tokens not part of an entity. After identifying N entities, sentences are generated for each entity pair, specifying the subject and object with special annotation tokens, respectively, [SUB][/SUB] and [OBJ][/OBJ]. These modified sentences are then processed by a second model that produces a sentence embedding, which is subsequently classified by a classifier to determine the relation between the entity pair, if any. The first phase employs a fine-tuned BertForTokenClassification model, while the second phase uses a Bert model to obtain the sentence embedding from the [CLS] token embedding, followed by a linear layer acting as a classifier. From now on I will refer to this third approach as the 'Pipeline model'. The complete architecture is shown in figures 4, 5.

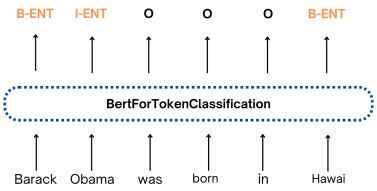


Figure 4: Phase 1: Named Entity Recognition

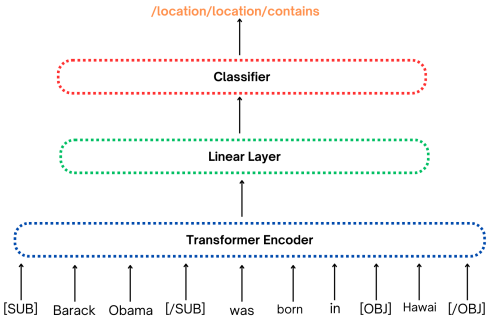


Figure 5: Phase 2: Relation Extraction

3 Experiments

To determine which of the three models was best suited for addressing the task of Relation Extraction, I conducted three parallel experiments. In each experiment, I trained the models on the training set and evaluated their performance on the validation set, monitoring the F1-score and loss for both the training and validation sets for all models.
Regarding the models based on the table-filling approach, the Baseline model achieves an F1-score of 0.49 on the training set and 0.46 on the validation set. In contrast, the Mixed model reaches an F1-score of 0.55 on the training set and 0.50 on the validation set, making it the better model of the two. On the other hand, the Pipeline approach achieves an F1-score of 0.91 on the training set and 0.90 on the validation set, establishing it as the best-performing model overall.

The graphs relative to the metrics the Mixed model, are reported in Figures 6, while the graphs relative to the metrics of the pipeline model phase 1 and phase 2 are reported in Figures 7 and 8.

From the Figure 8.a, it's evident that after two epochs of training, the model begins to deteriorate, with its F1 dropping from 0.90 to 0.55 on the training set with a similar pattern observed on the validation set. This led me to halt the training process and select the best parameters obtained after the initial two epochs.

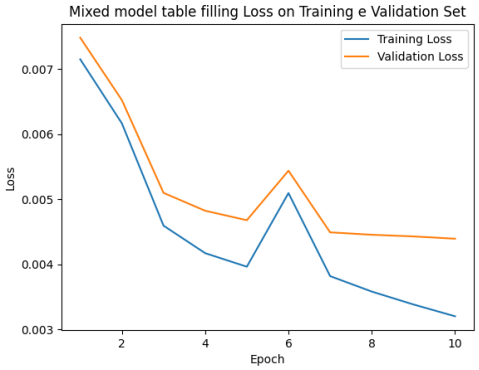
4 Conclusion

The results of the previous experiments indicate that the Mixed model achieves slightly better results compared to the Baseline model. However, the best-performing model is the Pipeline one, which achieves an F1-score of approximately 0.85 on the test set.

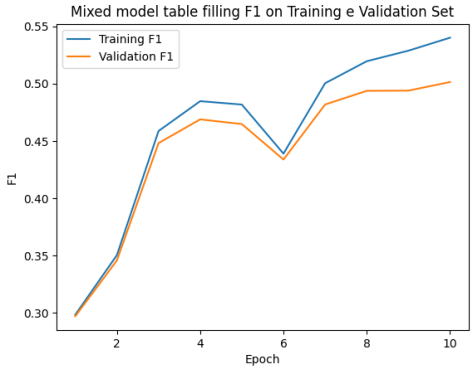
Finally, all the results obtained on the test sets for each model are documented in Table 1.

Model	F1-score
Baseline	0.42
Mixed model	0.49
Pipeline model	**0.85**

Table 1: F1-score for each models on test set

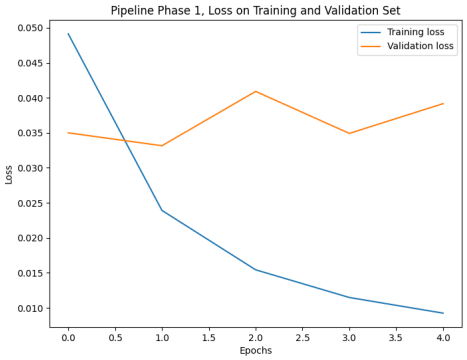


(a) Mixed model Loss on Training and Validation set

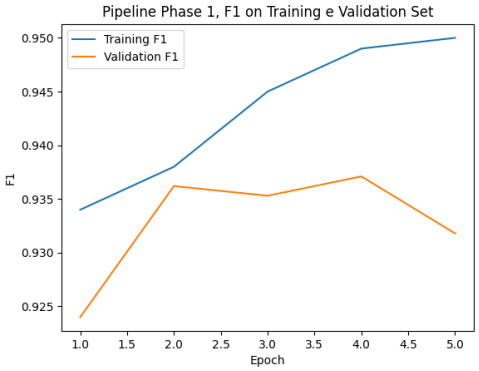


(b) Mixed model F1 on Training and Validation set

Figure 6

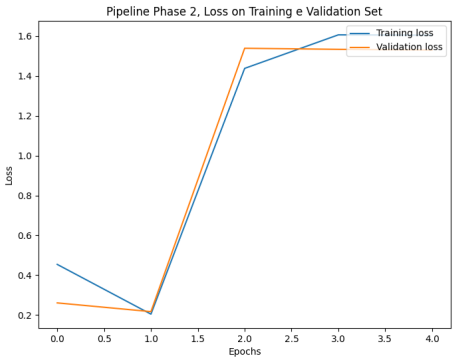


(a) Pipeline model phase 1 Loss on Training and Validation set

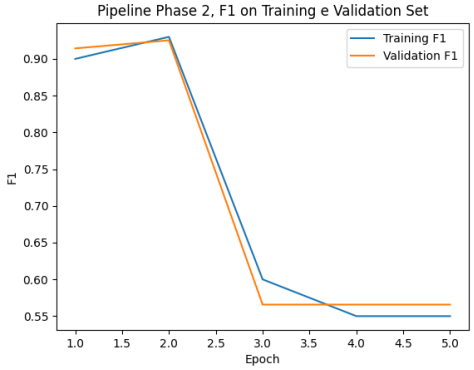


(b) Pipeline model phase 1 F1 on Training and Validation set

Figure 7



(a) Pipeline model phase 2 Loss on Training and Validation set



(b) Pipeline model phase 2 F1 on Training and Validation set

Figure 8