
# Business Relations extraction

**Spring 2022 project - Final Report**

**Author**
Romain Bourgeois
**Supervisors**
Thibault Ehrhart
Raphaël Troncy

Spring semester 2022

# Abstract

This project aims to build a general purpose relation extractor in the business field. Unfortunately, it failed to implement the extractor developed in the "Matching the Blanks" paper. This report describes the work done for this project. From data acquisition tasks with unsupervised methods and manual labeling, to named-entity recognition modeling, this paper abstracts the intended extractor by fine-tuning Bert, it also undertakes the relation classification task using the K-nearest neighbour algorithm and draws a method that could be used for discovering new relations. Even though the project yielded poor accuracy metrics, this project finally assembled the whole pipeline for extracting relation tuples from raw text inputs.

# Contents

# Chapter 1

# Introduction

Relation extraction is an information extraction domain that aims at identifying semantic relations between entities in a sentence. This task is key in extracting information from unstructured text for building knowledge graphs. Relation extraction can be used to derive RDF triples where relations serve as predicates and entities refer as subjects and objects. Entities are assigned with unique identifiers and ontologies may be leveraged to build complex knowledge graphs. Named-entity linking is therefore one of the challenges in relation extraction tasks. One could use existing databases in the semantic web or build its own entity identifiers from Named-entity recognition tasks, thereby increasing the scale and scope of the intended knowledge graphs. Furthermore, relations or predicates may or may not define asymmetrical relationships between entities, hence implying the need for specifying directions and rules over these directions. These knowledge graphs can add lots of value for business and finance professionals and relation extraction in these domains has received lots of attention the past few years, leveraging diverse methods imported from other domains such as the medical industry. These methods differ in the way they are precise and scalable. Precision is key because mislabelling relationships can potentially lead to poor decision making by practitioners and scalability is important if we want knowledge graphs to be informative and useful.
Recent advancements in language modeling are being intensively used to replace highly feature engineered solutions. Even though most of the focus has been leveraging labeled data for classifying relations, general purpose relation extractors have been recently used for making this task more scalable. Aiming to generate one such extractor, this project focuses on data acquisition and labeling, name-entity-linking, relation classification and exploration. First, this paper will provide a review of the literature of the topic. It will then detail the undertaken tasks for data acquisition and labeling. Furthermore, this paper will go through the name-entity-recognition step and discuss its limitations. It will then explain what went wrong with the implementation of the "Matching-the-blank" paper and describe the alternative undertaken and discuss its flaws. Finally, the paper will go through the relation classification task

with the nearest-neighbor algorithms. With an implementation of the k-means algorithms, a method aiming to discover new predicates will be explained in order to fully leverage general purpose extractors' advantages.

# Chapter 2

# State of the Art

Deep learning based relation extraction methods can be divided into two subclasses: pipeline and joint learning methods.
The pipeline method brings down the task in two steps. The first one is named-entity linking which can be done with either an entity database or a named-entity recognition task. The latter consists of classifying groups of two identities to a relation. In order to be able to take asymmetrical relations and presence of multiple relations within one sentence into account, these models all use a technique to embed target entities. CNN and RNN based models are mainly used. CNNs are great at extracting features and multi-level attention models have performed well on benchmarks. Chang et al (2016) achieved third rank on the SemEval 2010 Task 8 benchmark. Zhang and Wang (2016) added recurrent layers to build dependency-path aware attention neural networks and ranked fourth on the same benchmark. Wu and Ye (2019) proposed to leverage language models with entity information and yielded second in the ranking. Their method consists of enclosing entities with special tokens and feeding Bert's hidden layers to a fully connected neural network.
The state-of-the-art was achieved by Soares et al (2020). Their method called "Matching the Blank" introduced the first general purpose relation extractor. Building on Wu and Ye's entity-aware fine-tuned Bert model, their work draws on Harris' distributional hypothesis to relations to make entity tokens' output embeddings more informative. The hypothesis states that if a relation has been established between two entities, the same relation is expected to reappear if the entities are to be seen in another sentence. More specifically, they implement a second loss so that the entity marker embeddings get close if the underlying relation is similar. Each sequences in the training dataset will compute the cross entropy with the set of labeled data D and add this loss term to the cross entropy term of the labels of the masked tokens. The added loss term is specified in figure 2.1. Furthermore, a "blank" token is replaced with the entity marked with a high probability to prevent the model from simply learning the actual entities. The state of the art was achieved with the help of convolutional layers stacked to the embeddings output. Relations can also be inferred from

the embedding feature space alone and classification yields good results with a few labeled relations, providing that the model was trained with some abundant types of relations.

$$p(l = 1|\mathbf{r}, \mathbf{r}') = \frac{1}{1 + \exp f_\theta(\mathbf{r})^\top f_\theta(\mathbf{r}')}$$

$$\mathcal{L}(\mathcal{D}) = -\frac{1}{|\mathcal{D}|^2} \sum_{(\mathbf{r},e_1,e_2)\in\mathcal{D}} \sum_{(\mathbf{r}',e_1',e_2')\in\mathcal{D}} \tag{1}$$
$$\delta_{e_1,e_1'}\delta_{e_2,e_2'} \cdot \log p(l = 1|\mathbf{r}, \mathbf{r}')+$$
$$(1 - \delta_{e_1,e_1'}\delta_{e_2,e_2'}) \cdot \log(1 - p(l = 1|\mathbf{r}, \mathbf{r}'))$$

Figure 2.1: Additional loss element from "Matching the Blanks" paper

# Chapter 3

# Implementation

## 3.1 Data acquisition and unsupervised methods

The literature yields a consensus over the need of coming up with at least a minimum level of labeled data. Unfortunately, we could not find a public dataset on the business domain. The initial plan was to build high-precision and low-recall methods to retrieve some relationships. Following the method from Zuo et al (2017), this may enable us to retrieve entities and crawl into vast unstructured text and fetch sentences mentioning those entities, with the hope of finding more complex syntactic structures. The methods were built using the dependency parsing tool from the 'spacy' library. This tools infers relationships between words in a sentence and describe their syntactic roles. It looked for a list of simple hand-crafted forms of sentences and relations were attributed depending on some input conditions.

More specifically, three syntactic structures were targeted. The first one was called subject-verb-complement (SVO). The basic idea is to establish a list of verbs and extract the sentences that satisfy some conditions on the predicted entities from spacy. The steps of the method are the following:

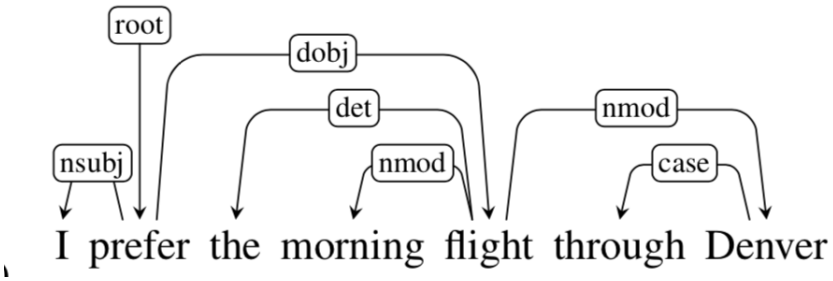


Figure 3.1: Dependency parsing

1. select relation name, a list of verbs corresponding to the target relation, select a list entity types
2. while iterating through documents:
3. save all the entities with a document using spacy's NER model. Types of entities can be specified by a list. The types depend on the target relation
4. while iterating through sentences:
5. save the token indexes of the entities that were previously found
6. if the number of entities found in the sentence is greater or equal to two, continue:
7. check if the lemmatized root corresponds to one of the item with the target verbs. If it is true, continue
8. go one step up in the dependency tree, if the item is a subject and an entity: continue. If not, try going one more step up in the tree and check the conditions
9. if conditions were met in 8, execute the same step for direct and indirect objects, with max number of steps down in the dependency tree of two
10. if all conditions were met, the relation is saved and exported

The second one was called attribute (ex: "Facebook is a competitor of Tiktok") and segregated the predicate based on a noun instead of a root verb. The method works similarly as SVO. Finally, an example of the last one would be "Instagram was the company acquired by Facebook" and was called "acl", selection was made on the verb.

The rationale behind this methods was that relationships identified by this process were likely to reoccur in text. The probability that these occurrences would describe the same relationship being hence high, we could label more complex sentence structures to feed a classifier. Moreover, it shall be noted that those inferences still had to be checked to avoid mislabelling. Unfortunately, this could have only worked if we could crawl a very large amount of text which we could not. The methods are coded under the `unsupervised_methods.py` file. The `unsupervised_relationExploration` notebook displays an implementation of these methods. Recall scores will be displayed in the next section.

## 3.2 Dataset

A dataset of one hundred news articles was provided. Among those, fifty were afp articles and the rest came from various economic journals. The text files can be found under the 'rawdata' folder. The dataset was manually labeled for both entities and relations with the help of the 'UBIAI' platform.

The list of entities and relations were selected somehow arbitrarily as we read through the articles. The named-entity recognition consists of classifying the following entities: organizations, products and services, amount of money, person, market and location. On the other hand, eleven relations were investigated.

| Relations | Count | Unsupervised method recall count |
|---|---|---|
| Partnership | 49 | 1 |
| Research Project | 9 | 0 |
| Subsidiary | 29 | 1 |
| Purchase | 1 | 0 |
| Financing | 9 | 0 |
| Recruitment | 0 | 0 |
| Launched product or service | 9 | 2 |
| Has product or service | 112 | 0 |
| Firm is based in | 38 | 2 |
| Person works in | 85 | 0 |
| Operates in market | 163 | 0 |

The first thing that comes to mind is that the dataset is very unbalanced, hence making it difficult for a standard classifier to predict. One could use the most abundant relations only or build a general purpose extractor. Another point to make is the fact that all relations are asymmetrical except "partnership" which means that only an entity-aware classifier could be used to complete the task. Finally, it shall be noted that the unsupervised methods were implemented the named-entity recognition model from spacy, hence making it impossible to target some types of entities such as 'market' or 'amount of money'.

## 3.3 Named-entity recognition

The entity detection task could not be done with spacy alone. In fact, labeling raised the fact that most occurring relations in my dataset were not or poorly modeled with spacy's. Moreover, products, services and markets often required the presence of multi-word classification which yielded poor results with spacy's 'product' label. As a result, a named-entity recognition (NER) model was implemented to match this project's needs.

A NER classifier was implemented inside the 'ner' folder to classify sequences of tokens from the classes described above. Two pre-trained models were used: 'distilbert' and 'finbert'. The first one is a standard Bert classifier and the latter was pre-trained on financial texts. The two models are open-source in the huggingface library. The models were trained on 'Google Collab' and a grid search was implemented for hyperparameter tuning and the one with the highest F1 score was saved.

## 3.4 Matching the Blanks implementation and Bert fine-tuning

The "matching the blank" method was chosen for this project. The decision was made because of the high imbalance that was present in the labeled dataset. It also satisfied the ability to detect asymmetrical relations and was robust to the presence of multiple predicates in sentences. This paper finally aimed to develop a method for exploring new types of relations within business corpuses. Providing a well-trained general purpose relation extractor, one could explore a corpus in its feature space and identify unlabelled clusters. Those could hence be labeled and be fed to retrain the relation extractor for better accuracy and higher scope of predicates. This will be further explained in section f.

Unfortunately, the project failed in implementing the paper. Lacking skills in pytorch and tensorflow, the goal was to implement the "matching the blank" task at the preprocessing phase instead of doing it during the training iteration in batches. It sounded easier to implement because we were only familiar with the trainer method from the transformers library. The project failed in modifying the loss function which seemed to not be tunable with the trainer method. As a result, the general purpose relation extractor was trained without the special loss and embeddings were pulled from the entity special token embeddings. The result is that the extractor was built without learning to represent entity embeddings in a similar fashion for the relations. Some contextual information is still to be incorporated through the embeddings but its efficiency is highly compromised. The pretrained Bert models used were the same as the one used for named-entity recognition and the masked-token method was used to fine-tune the language models. Special tokens were places around the entities. These were extracted for every sentences and all combinations of entity embeddings were computed. The preprocessing methods can be found in the `rel_preprocessing` notebook and the training file under the name `fine_tune_maskedModel.ipynb`.

## 3.5 Classification using the K-nearest neighbour algorithm

Images d,e and f from figure 3.2 display all 3 types of features that were exported for the Bert model. The first one takes the embedding from the start token only, the second one max pools the entities' token embeddings and the last one takes the first entity token only.

The k-nearest neighbor algorithm was chosen as a classifier because it follows the "Matching the Blank" purpose that embeddings learn to be similar for identical entities. Furthermore, such an algorithm was expected to mitigate the imbalance effect if we were to choose a low number of K. The best accuracy score was found for a k equal to 3 at a mean accuracy score of 17The poor accuracy can be explained by the fact that the model did not learn to represent the embeddings similarly for identical relation statements. Also, most of the

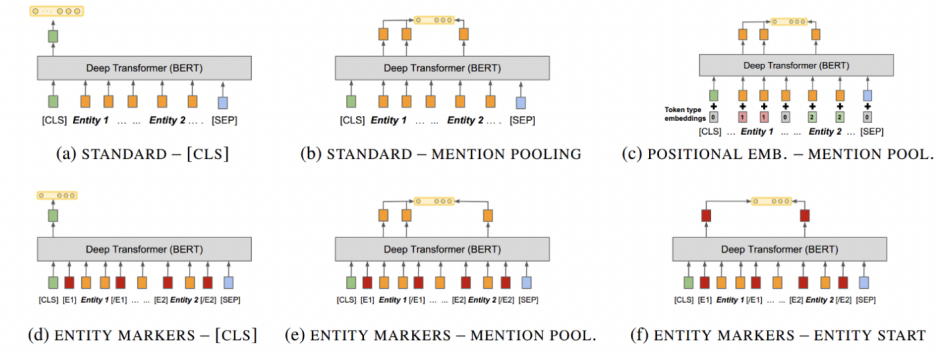


Figure 3.2: Extracted embeddings

data did not present any relations, these embeddings invading feature space are hence worsening the effect of this poor representation. On the other hand, the accuracy scores are suffering from relation imbalances which bring the average score down. The notebook is named `relation_classification.ipynb`'.

## 3.6 Optimal K-means and new relations discovery

This step of the project does not make a lot of sense given the effectiveness of the extractor but was briefly implemented in `relation_classification.ipynb`'. It consists of running the k-means algorithm for a range of clusters and compute the average highest presence score of relations within those clusters. One could then inspect the unlabelled clusters and identify other relations that were not labeled for.

## 3.7 Testing pipeline

A testing pipeline was implemented in the `predict_relations_pipeline`' notebook. I sequentially map all combinations of entities and predict their relations using the k-nearest neighbor classifier. An example of the output is given in figure 3.3.

```
predicting sentence : ['instagram was bought by facebook to compete with tiktok.']
predicting on  6  different combinations of entities
predicted ( instagram , PARTNERSHIP , ['tiktok']
predicted ( facebook , PARTNERSHIP , ['tiktok']
no relations predicted for head entity  ['instagram']  and child entity  ['tiktok']
predicted ( instagram , PARTNERSHIP , ['facebook']
predicted ( facebook , PARTNERSHIP , ['tiktok']
predicted ( instagram , PARTNERSHIP , ['facebook']
predicting sentence : ['eurecom is based in france and offers educational courses.']
predicting on  2  different combinations of entities
no relations predicted for head entity  ['eurecom']  and child entity  ['france']
predicted ( eurecom , BASED_IN , ['france']
predicting sentence : ['eurecom will partner with inria for various research projects.']
predicting on  2  different combinations of entities
predicted ( eurecom , PARTNERSHIP , ['inria']
predicted ( eurecom , PARTNERSHIP , ['inria']
```

Figure 3.3: Testing pipeline

# Chapter 4

# Conclusion and Future works

To conclude, this paper explored the various techniques needed for building knowledge graphs. Unfortunately, the results were far from satisfying. When opting to build a general purpose relation extractor was arguably the appropriate choice for such a project, the goal was too ambitious and a relation classifier based on Bert may have been a more appropriate choice. Further development could be investigated in a successful implementation of the "Matching the Blanks" paper. One could also focus on the named-entity linking task which is essential for knowledge graph construction.

# Bibliography

[1] Wang L, Cao Z, De Melo G, Liu Z. Relation classification via multi-level attention CNNs. vol. 3. Berlin, Germany; 2016. p. 1298–1307. Available from: 10.18653/v1/p16-1123.

[2] Cai R, Zhang X, Wang H. Bidirectional recurrent convolutional neural network for relation classification. vol. 2. Berlin, Germany; 2016. p. 756–765. Available from: 10.18653/v1/p16-1072.

[3] Wu S, He Y. Enriching pre-trained language model with entity information for relation classification. Beijing, China; 2019. p. 2361–2364. Available from: 10.1145/3357384.3358119.

[4] Soares LB, FitzGerald N, Ling J, Kwiatkowski T. Matching the blanks: Distributional similarity for relation learning. Florence, Italy; 2020. p. 2895–2905.

[5] Zhe Zuo, M. Loster, Felix Naumann. Uncovering Business Relationships: Context-sensitive Relationship Extraction for Difficult Relationship Types; 2017