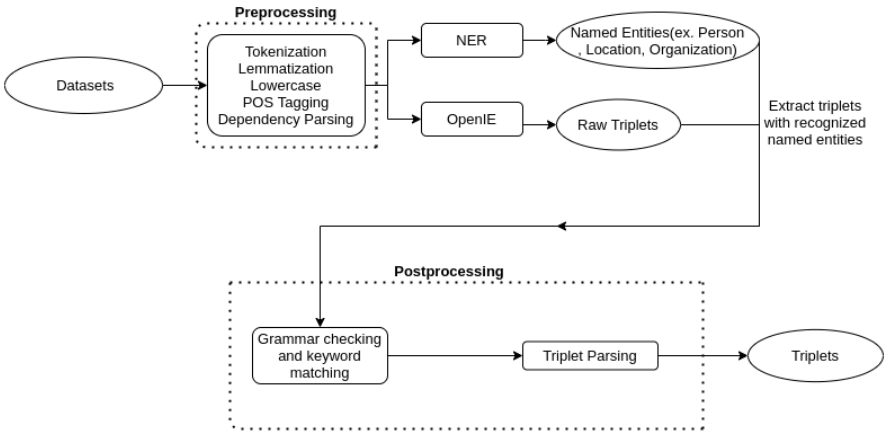
1 Overview



2 Named Entity Recognition

1. **Input**: Unstructured text **Output**: Text tagged with named entities (Organization, Date, Person, Location)

2. **Preprocessing Steps**:
- Tokenization
- Lowercase
- Lemmatization
- POS Tagging
- Dependency Parsing

3. Currently **Stanford NER** is performed to extract the general named entities. In the future, it would be more efficient to train an own NER tagger for domains such as engineering and technology, so that we can get domain specific entities and skip the step of keyword matching later.

3 OpenIE

1. **Input**: Unstructured text **Output**: triplets built based on the "Subject-Verb-Object" rule

2. Currently the **Stanford relation extractor** is used for this task. Some triplets generated do

not make sense or are not grammatically correct. For instance, we may get "us-are not-in Munich" or other bad triplets.

```
nature , in , scientific reports
us , scale quickly , our computer
us , scale , our computer
our computer , cheaply leveraging , existing microchip industry
it , allowing , us
our computer , leveraging , microchip industry
our computer , leveraging , existing microchip industry
our computer , cheaply leveraging , microchip industry
```

3. From the named entities extracted in the second step, we could extract the triplets that have recognized named entities. However, many of these triplets are duplicated or still are not grammatically correct. Therefore, further postprocessing steps need to be performed.

4 Postprocessing

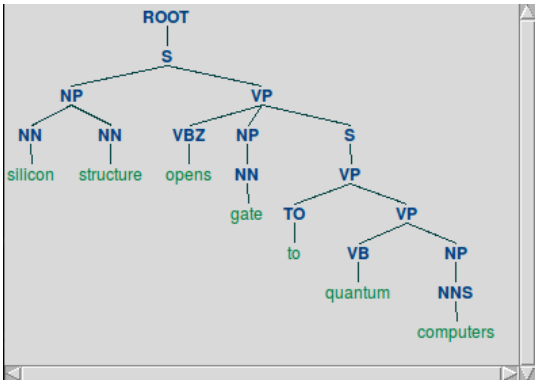
4.1 Grammar Checking and Keyword Matching

1. Currently the triplets with incorrect grammar are removed using rule-based methods. To get the domain specific triplets, we use keyword matching to get the relevant triplets.

4.2 Triplet Parsing

1. There are a lot of triplets with similar structure created by the relation extractor (For example, "quantum computing-is-in trend" and "quantum computing-is-currently in trend). We only need one for graph visualization. To solve the problem of triplet duplication, statistical parsing is used to get the triplets that have the best parsed results.

2. Currently, expectation maximization training is used for triplet parsing. First, **Stanford Parser** is used to parse each triplet to get its rules. These rules could be visualized in form of a parse tree:



For each triplet, we get such a parse tree. For similar triplets, we will get similar parse analysis (with extra rules such as rules about adjective or adverb). Based on how often the triplets appear in the data, the probability of the rules will be calculated, something like:

Regel	f	p
S $\rightarrow$ NP VP	3	1
NP $\rightarrow$ D N	1	0.14
NP $\rightarrow$ D N N	1	0.14
NP $\rightarrow$ N	4	0.57
NP $\rightarrow$ NP PP	1	0.14
VP $\rightarrow$ V	1	0.33
VP $\rightarrow$ V NP	2	0.67

the total probability of the tree bank will then be calculated. Afterwards, probability of tree banks will be weighted and reweighted, and the triplets with highest probabilities at the end will be chosen

(see stat_parser.py) . In the future, **inside outside algorithm** could be used to make it more efficient.

5 Current Issues

Some issues could still be found after these steps:

1. **Semantically correct triplets, but verb forms incorrect**: For example, resulted triplets will be "silicon-be crafted into-array of complex structures", "researcher-build-cryptologically quantum computer", or "Google-creating-cloud access".

2. **Semantically and syntactically correct triplets that do not deliver any key information**: For example, "research-is- driving-great deal of investment" does not provide any key information (i.e. about what?). Training sentence level embeddings for example with Quick Thought can help resolve this problem. We could use seed triplets as query to extract other similar triplets.

3. **Visualization with Gephi**: Edge labels could not be shown in the pdf and png formats. Other tools such as **ReGraph**, **NetworkX**, etc. could be tested in the future.