COMSM0037 Text Analytics Coursework: Text analytic system for entity and relation extraction from scientific articles

Abstract

This report shows the methods that we implemented and the evaluation for information extraction. The research was divided into separate tasks, which in the end would combine and show the result to the end-user. The tasks consist of extracting key entities from abstract scientific papers, extracting the relationships that entities can have and exploring the datasets to find valuable information through visualization. Two basic implementations will be used to solve these tasks. The first one is an NLP pipeline model using CRF (Conditional Random Fields) with some additional features like pos-tags and unigram. The second model is a Bidirectional LSTM neural network. We will evaluate the performance of each model and compare both results together. Alongside, we made a visual analysis to present top relations and the word cloud of a particular query word given by the user. The last task is an open assignment where we present a critical thinking analysis over the Glucose dataset.

1 Introduction

Many papers are uploaded nowadays to solve different scientific problems. There is a variety of models used, different techniques and algorithms about different topics. Thus, there is a need to better process and summarise these papers and identify critical information, which is the purpose of this report. We will be using natural language processing and machine learning models to solve these tasks. The data used to train the model are provided by ScienceIE[1] and are divided into three groups train, dev, test. The model is constructed to solve three different tasks by using these datasets. The first task has to do with the extraction of the relevant types of entities. We would explore all the entity types and try to make accurate predictions in test data after we train our model. We will evaluate each model used and describe each model limitations alongside some suggested solutions. On the second task, an analysis is done in the entities, and relations between them are extracted for finding synonyms or hyponyms. We used a CRF dependency parser that would construct step by step the parsing tree of entities, and from that, we could analyse all the dependencies between entities. Furthermore, we would show some standard error that appears and propose some required techniques to solve them. After completing the first and second task, a third task is implemented to explore the datasets. This is done for better visualisation to the end-user. Given a query from the client, we constructed a relation frequency that contained that query, and we created a word cloud that visualises similar words that are part of the same subject. The last task consists of an open part of the assignment to critically review a given scientific paper. We will try to answer some questions about the GLUCOSE dataset and detect any limitations. Given our current knowledge, we will present some ideas that we believe could improve the methodology that was used in that research.

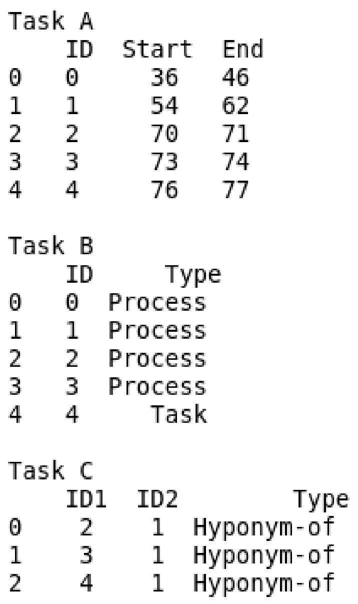
2 Task description

The tasks that were solved in this project were divided into three main subtasks:

Task 1: Construction of the model to extract the entities from the text in the datasets. Three main entities: Tasks, Process and Materials.

Task 2: Extracting relations from the entities of interest in order to obtain information on how they relate to one another. There were two relations provided in our datasets, Synonym-of and Hyponym-of.

Task 3: Exploring information in data for a better understanding of the relation for the client.
Below we represent an example of the output of the model for all the tasks. We see from the output



```
Task A
    ID  Start  End
0   0     36   46
1   1     54   62
2   2     70   71
3   3     73   74
4   4     76   77

Task B
    ID     Type
0   0  Process
1   1  Process
2   2  Process
3   3  Process
4   4     Task

Task C
    ID1  ID2        Type
0    2    1  Hyponym-of
1    3    1  Hyponym-of
2    4    1  Hyponym-of
```

Figure 1: Output from tasks

presented to the user for each task that we described in the Report. First, the ID identifies the entity in that particular document and the order in the document. Second, we can notice the span from the starting point to the endpoint of that entity in task A. In the second task, we display again the ID of the entity that we found alongside its type. Last, we display the relations between entities that we extracted from the dependency parser. It will display the entities IDs and decide whether they are synonyms or hyponyms.

2.1 Task 1 : Entities Extraction

Entity extraction is one of the central cores in natural language processing. For our problem, we formulated three entities type, which are Tasks, Process and Material. For each type, we had three different prefixes B (Begin), I(Inside) and O(Outside), that showed the span of that entity and we used them to calculate the size of the sequence. Below we represent all the unique set of entities that we analyzed.

Number of possible entities: 7
Possible entities: 'B-Task', 'I-Material', 'I-Task', 'O', 'I-Process', 'B-Process', 'B-Material'

We used two different methods to implement the requirements of this task. The first model we choose was a CRF (Conditional Random Fields), which is a sequence model algorithm or tagger that nltk.tag library[2] provide. The second model we choose was the Neural network model called BiLSTM (Bidirectional LSTM) using torch[3] library.

2.1.1 CRF sequence modelling tagger

We decided to construct our first model as a CRF tagger which is a discriminative model. We went with this implementation because they are more accurate models, and they can find the optimal global sequence. Moreover, this implementation has many advantages over the Naive Bayes or other methods that use bag-of-words representations, which assumes that every word is independent of the others and ignore the order of the words in the sentence. Thus, much information about the meaning of the sentence lose, so this is a drawback for our task. On the other hand, the CRF sequential model is an undirected graphical model, keeping states and considering both left and right word neighbours. This helps the system know about the sequence of entities by processing one at a time, which will compose all the meaning of the sentence and give us a better result for the prediction. To construct the model, we first retrieve each text from each paper and get every word and entity tag for the training. Then we wanted to add some syntactic information to make a better prediction, so we added a new feature to the tagger, the POST-tag for labelling each word with the corresponding tag, and we used 1-gram as a one-word sequence. Below we represent the implementation of the model. We pass the document as 1gram word,

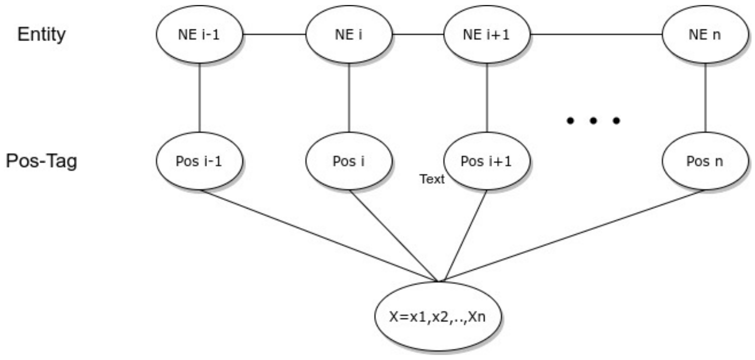


Figure 2: CRF with POS-tag

and we compute the Postag of that unigram, and then we connect each entity with its neighbours. By doing this, we make the prediction of the label taken into account the context information.

2.1.2 BiLSTM neural network

The second method that we implemented was a Bidirectional LSTM neural network. This model is part of an End-to-End deep neural network which is an approach that does not need any complex feature engineering, but we can predict the output by just using the raw text. We used the LSTM model because they can store long-distance information over the network, and they eliminate any vanishing gradient problem that other RNN models have because of their unique structure using gates. We added a second layer because they are optimal for sequence labelling tasks to transfer information forward and backward. After all, we wanted for each node to depend on both previous and later nodes. In order to train our model, we firstly needed to convert our data into word embeddings. We created a vocabulary for all the word and converted each document as a vector representation with word IDs. We did the same thing for entities, and thus, we constructed the labels that should be predicted. For each vector, we used a fixed sequence length = 318 (maximum document length), and for those smaller, we used some padding with zero value. We used 100 hidden units and three hidden layers to create the model as we found to be optimal values when we run our experiments. Below a figure is showed for the implemented model. As shown in the figure above, we used embedding as inputs with dimensions (vocabulary_size,100).

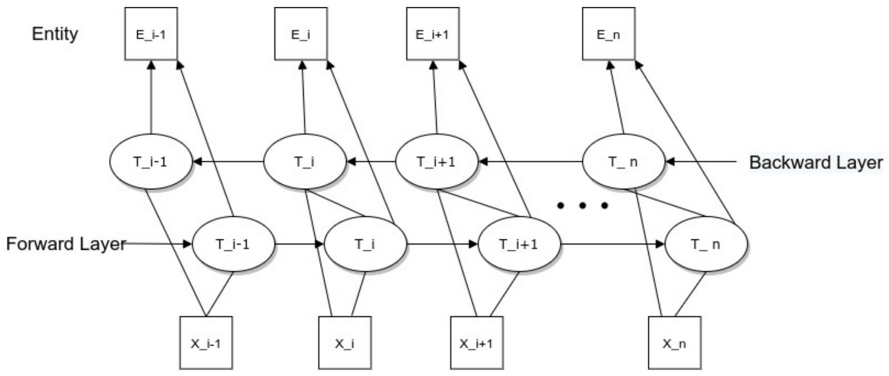


Figure 3: BiLSTM with word embedding

Moreover, we constructed 3 BiLSTM hidden layers. To prevent over-fitting and generalize the model, we used a dropout technique that drops and ignores some units in the hidden layers, and we set the frequency to be 0.2 (20%). The output consisted of 8 classes that represent different entities types that we had.

2.1.3 Evaluation

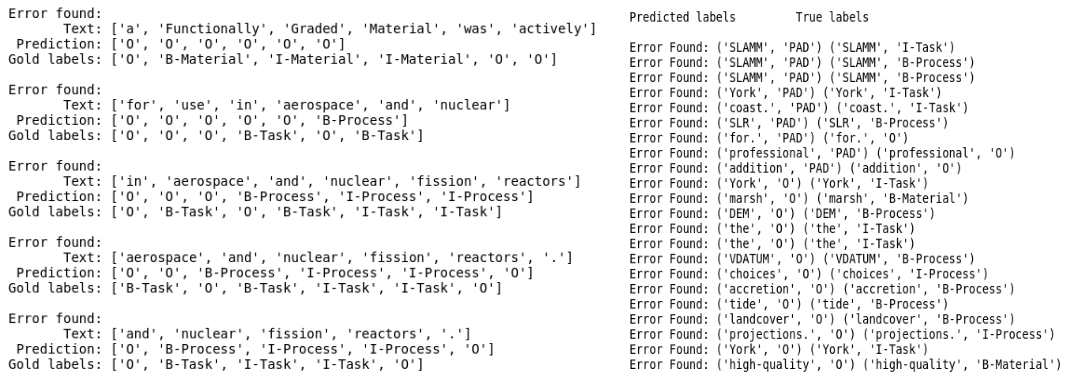
In order to compare the two implemented models and evaluate each performance, we decided to run some experiments. First, we compared the two model by measuring the accuracy in test data. Furthermore, for the CRF models, we measured the F1-score for each entity type and compare individual performance. On the other hand, for the LSTM model, we run some experiment to measure the model error using the loss function. We tried to tune the NN model parameters by training the model in train and validation data for ten epochs, and then we tested the model on test data. This could not be done in the CRF model because no parameter could be tuned during the training ,and this is a limitation in the implementation. Below we represent a table with the performances for both models and all the experiments that we tried for the evaluation.

Sequence models	
Simple CRF	
F1 score Process	0.192
F1 score Material	0.150
F1 score Task	0.0389
Macro-average f1 score	0.127
Test Accuracy score	**70.38%**
CRF with post-tag	
F1 score Process	0.211
F1 score Material	0.212
F1 score Task	0.0436
Macro-average f1 score	0.156
Test Accuracy score	**73.816%**

BiLSTM model performance				
Epoch	Train Loss	Validate Loss	Train Accuracy	Validate Accuracy
1	1.3908	0.7584	60.29%	81.65%
2	0.6636	0.7095	82.98%	81.97%
3	0.6342	0.7049	83.19%	82.09%
4	0.6237	0.687	83.25%	82.25%
5	0.6138	0.6823	83.33%	82.25%
6	0.593	0.6567	83.33%	82.25%
7	0.5552	0.6308	83.33%	82.25%
8	0.5151	0.6074	83.46%	82.26%
9	0.4828	0.5991	84.10%	81.87%
10	**0.389**	**0.591**	**86.01%**	**81.62%**
Test Loss: 0.5209 / Test Accuracy: 84.5000%				

Seeing the tables above, we can notice that the BiLSTM model performs better than the CRF (POSTtag+unigram) as we notice an accuracy of around 85% compared with the Sequence model 74%. This happens because the deep neural networks have hidden layers that can generate complex functions that can be learned automatically and fine-tuned our model by training with test and development datasets. We can say that accuracy has some limitation in datasets that are not equally distributed, and we can say that in our data, there are more O than other types of entities $Task, Material, Process$[4]. We measured the F1-score for every type of entity in the simple CRF model, and we can notice that Process and Material had around 15% f1-score while the Task entity was performing poorly to around 3%. We improved the performance of the CRF model by adding the new features (POStag+1gram), and we saw a slight 5% improvement in both accuracy and F1-score.
We can say that there are some computational limitations at both the models for the training stage as they consume much time. There is also a need for parallelisation of the process in order to make it scalable. This can be a limitation in the LSTM model because the processing of each entity is done step by step. We investigated the error made by the model, and we noticed that most of them had to do with the misclassification of Material to O(Outside) and Tasks to Process. This may happen because, in our datasets, we have more data as O and Processes than as Materials and Tasks, so the model can not learn them effectively. On the other hand, we can notice different kind of errors in the LSTM model. We notice that the model predicted the 'PAD' type, which is the padding that we made to create the embedding for the inputs. The other error notice is the wrong prediction of TASKS to Outside type, and

this happens because we had trained the model with a lot of O Type. This can be shown also below in the figure. What we can do for the CRF model is to add more data for training to equal the distribution



```
Error found:
       Text: ['a', 'Functionally', 'Graded', 'Material', 'was', 'actively']
 Prediction: ['O', 'O', 'O', 'O', 'O', 'O']
Gold labels: ['O', 'B-Material', 'I-Material', 'I-Material', 'O', 'O']

Error found:
       Text: ['for', 'use', 'in', 'aerospace', 'and', 'nuclear']
 Prediction: ['O', 'O', 'O', 'O', 'O', 'B-Process']
Gold labels: ['O', 'O', 'O', 'B-Task', 'O', 'B-Task']

Error found:
       Text: ['in', 'aerospace', 'and', 'nuclear', 'fission', 'reactors']
 Prediction: ['O', 'O', 'O', 'B-Process', 'I-Process', 'I-Process']
Gold labels: ['O', 'B-Task', 'O', 'B-Task', 'I-Task', 'I-Task']

Error found:
       Text: ['aerospace', 'and', 'nuclear', 'fission', 'reactors', '.']
 Prediction: ['O', 'O', 'B-Process', 'I-Process', 'I-Process', 'O']
Gold labels: ['B-Task', 'O', 'B-Task', 'I-Task', 'I-Task', 'O']

Error found:
       Text: ['and', 'nuclear', 'fission', 'reactors', '.']
 Prediction: ['O', 'B-Process', 'I-Process', 'I-Process', 'O']
Gold labels: ['O', 'B-Task', 'I-Task', 'I-Task', 'O']
```

```
Predicted labels        True labels

Error Found: ('SLAMM', 'PAD') ('SLAMM', 'I-Task')
Error Found: ('SLAMM', 'PAD') ('SLAMM', 'B-Process')
Error Found: ('SLAMM', 'PAD') ('SLAMM', 'B-Process')
Error Found: ('York', 'PAD') ('York', 'I-Task')
Error Found: ('coast.', 'PAD') ('coast.', 'I-Task')
Error Found: ('SLR', 'PAD') ('SLR', 'B-Process')
Error Found: ('for.', 'PAD') ('for.', 'O')
Error Found: ('professional', 'PAD') ('professional', 'O')
Error Found: ('addition', 'PAD') ('addition', 'O')
Error Found: ('York', 'O') ('York', 'I-Task')
Error Found: ('marsh', 'O') ('marsh', 'B-Material')
Error Found: ('DEM', 'O') ('DEM', 'B-Process')
Error Found: ('the', 'O') ('the', 'I-Task')
Error Found: ('the', 'O') ('the', 'I-Task')
Error Found: ('VDATUM', 'O') ('VDATUM', 'B-Process')
Error Found: ('choices', 'O') ('choices', 'I-Process')
Error Found: ('accretion', 'O') ('accretion', 'B-Process')
Error Found: ('tide', 'O') ('tide', 'B-Process')
Error Found: ('landcover', 'O') ('landcover', 'B-Process')
Error Found: ('projections.', 'O') ('projections.', 'I-Process')
Error Found: ('York', 'O') ('York', 'I-Task')
Error Found: ('high-quality', 'O') ('high-quality', 'B-Material')
```

Figure 4: Prediction errors vs golden labels in two models

of the different type entities, and with some more data preprocessing, we can reduce the occurrence of outside entities with some word normalization techniques like StopWords/Stemming or TF-IDF. On the other hand, on the BiLSTM model, we can try to minimize the size of the sequence to mean length by removing some unnecessary padding. We can also try some stop words removal that will minimize the number of outside types. The Neural Network model could be improved if we used an pretrained model (transformer) like BERT and use it for this specific task.

2.2 Task 2 : Semantic Relations Extraction

2.2.1 CRF tagger with dependency parser

The third task required implementing a model that would be trained in the dataset provided and would predict the two semantic relationships between the entities. These relationships were two, the first one Synonym-of and the Second one hyponym-of. To create the model, we used the CRF tagger with (POST-tag +1gram features) that we described before, and we added a Dependency parsing as a new feature provided by CoreNLPDependencyParser[5]. We decided to use the annotations from the gold standard entities in the test set because we wanted to get the best possible outcome of predictions. Adding this feature would help to identify the relations between entities by generating the parse trees. After the tree was completed, we then extracted all the relations and created the feature described below.

('source_entity': 'particle', 'target_entity': 'direct', 'source_pos': 'NN', 'target_pos': 'JJ', 'dep_path_source': '', 'dep_path_target': 'obl_advmod', 'dep_path_length': 2, 'hyponym-of')

As we see from the figure, we added to the feature the source entity, which is the first entity in the relations, the second entity which is the target entity. We have also considered the POST-tag for each word, the path and length in the tree. Last we have the golden standard label, indicating if the relation is a Synonym or a hyponym.

We collected all the features from the relations that were in the training dataset and trained a Naive Bayes Classifier to predict the labels from the nltk-library[2].

2.2.2 Evaluation

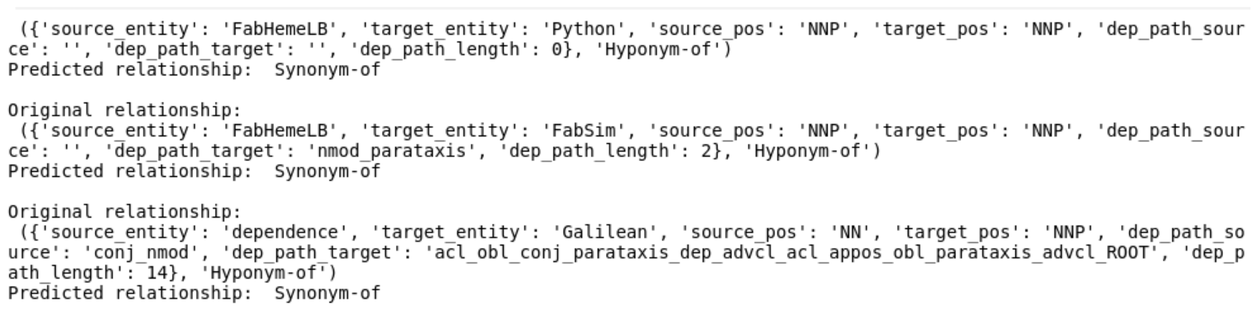
For the evaluation of the model, we used the Naive Bayes classifier in order to make predictions of the relationships between entities. The metrics that we analyzed were again the F1-score and the Accuracy. We first trained our model with the dependencies extracted for the training datasets. Then we measured the performance on development datasets, and after that, we additionally trained the model with the dev dataset again. In the end, we tested for the classification in the test data. Below we represent the confusion matrix for the predicted label vs golden label and calculate the type 1 and 2 error.

Confusion matrix		
	Synonyms	hyponyms
Synonyms	69	31
hyponyms	23	73

CRF with Dependency parsing	
F1 score Dev data	0.62
Accuracy score Dev data	66%
F1 score Test data	0.72
Accuracy score Test data	**74.4%**

Seeing the confusion matrix, we can notice that the false positives = 31 (Type 1 error) were the most mispredicted error. This is because the model predicted the Synonyms while the actual value was hyponyms. This is showed also in the errors below. We notice that the number of false-negative = 23 which is a Type 2 error while it said hyponym while it was synonym.

On the other hand, we noticed that Accuracy in test data was around 74.4%, and we noticed that the model had an F1-score=0.72 compared to 0.62 in dev data.



```
 ({'source_entity': 'FabHemeLB', 'target_entity': 'Python', 'source_pos': 'NNP', 'target_pos': 'NNP', 'dep_path_sour
ce': '', 'dep_path_target': '', 'dep_path_length': 0}, 'Hyponym-of')
Predicted relationship:  Synonym-of

Original relationship:
 ({'source_entity': 'FabHemeLB', 'target_entity': 'FabSim', 'source_pos': 'NNP', 'target_pos': 'NNP', 'dep_path_sour
ce': '', 'dep_path_target': 'nmod_parataxis', 'dep_path_length': 2}, 'Hyponym-of')
Predicted relationship:  Synonym-of

Original relationship:
 ({'source_entity': 'dependence', 'target_entity': 'Galilean', 'source_pos': 'NN', 'target_pos': 'NNP', 'dep_path_so
urce': 'conj_nmod', 'dep_path_target': 'acl_obl_conj_parataxis_dep_advcl_acl_appos_obl_parataxis_advcl_ROOT', 'dep_p
ath_length': 14}, 'Hyponym-of')
Predicted relationship:  Synonym-of
```

Figure 5: Relationship prediction errors vs actual labels

This misclassification problem can be improved by constructing a Deep neural network classifier instead of the Naive Bayes that could achieve better results with the addition of computational costs and use different feature engineering techniques like regular expression or word normalization that can then be used to construct word embeddings.

2.3 Task 3 : Data Exploration

After the implementation of the models for both task, we decided to explore the dataset. We decided to investigate two main things about the test data. Firstly we chose one example entity (Uranium and atomic) and counted the number of occurrences. Then, we created a set of unique relations and used a bar chart to show the frequency found in the datasets.

As we see from the chart, we notice that these relations have two main clusters of words. The first cluster contains information about atoms, substances, and a connection with human physical features where we see different relations between chemical compounds. On the other hand, the second cluster contains relations about a python module, its short term FaB and from the material, the transistors that it is constructed. Overall we can notice that every synonym of the word is just a short-form representation of whether the hyponym consists when that particular entity is part of the larger scale.

We wanted to explore more the datasets from the entity aspect. We decided to use the same word queries Uranium and Atom from the previous experiment from the test datasets, which had the largest occurrence. We created a list of words with all the other entities that co-occur in the same abstract paper as the queries entities. After collecting all the list of entities, we created a cloud of words using the Wordcloud library[6]. We used an image mask for better visualization and to get better cluster separations between words.

The word cloud figure shows us the most frequent word by increasing the font size. We can notice that the most frequent words are: uranium, polymer, surfactant, ATRP. We can also see that the clusters related to the same subject are grouped and closed together. This can be seen for all the chemical substances gathered near query word uranium like chloride, imidazolium etc. On the other hand, another cluster can be noticed, which describes surfactant, a liquid linked with superconductors and carbohydrates. To conclude, with the experiment, we showed that the user could use different entity queries, analyze the

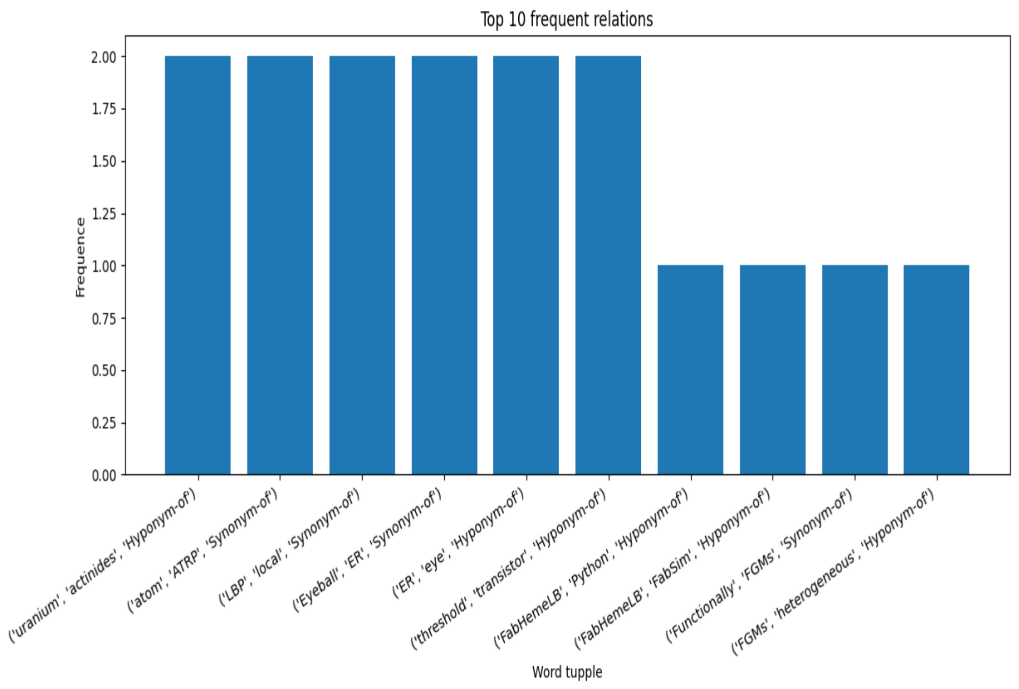


Figure 6: Top 10 entity relations

word clouds and take more information about the relations of words. Moreover, we can extract other entities in abstracts that have a similar meaning. This can be used later to search other articles with similar topics and find different solutions about that topic and models.

3 Conclusion

This project aimed to identify algorithms effectively using natural language processing techniques for information extraction. We showed that for the task such as entities extraction, Deep neural network such as BiLSTM performed better with 85% of accuracy than the Discriminative models such as Conditional random fields at around 74% with the use of extra features like Postag+1gram.

We explained that there are some limitations in both models when comparing the f1-score for each entity. This was due to the unbalanced distribution of entities types. We suggested improving the performance by making some additional pre-processing and word normalization tasks or combining the two models.

Furthermore, we showed that a dependency tree should be used for tasks such as entity relation extractions. We managed to achieve an accuracy of 75% and an f1-score=0.72 in test data. Last but not least, we showed how data could be explored with a simple query entity. First, we extracted the top entities relationships that showed data to be more focused on chemical substances. Then, we visualized the word cloud of that query and showed how the user could explore similar entities in abstracts from the same topic.

3.1 Task 4 : Open Assignment

The fourth task consists of an open assignment, and we included this task as a different section in our report. We were given three papers to do a critical review, and my selection was GLUCOSE: GeneraLized and COntextualized Story Explanations." by Mostafazadeh, Nasrin et al.[10]. The review is constructed as answering questions that would explain the main findings in the paper, the experimentations and in the end, we will give our opinion about the limitations and what technique we would try to solve them.

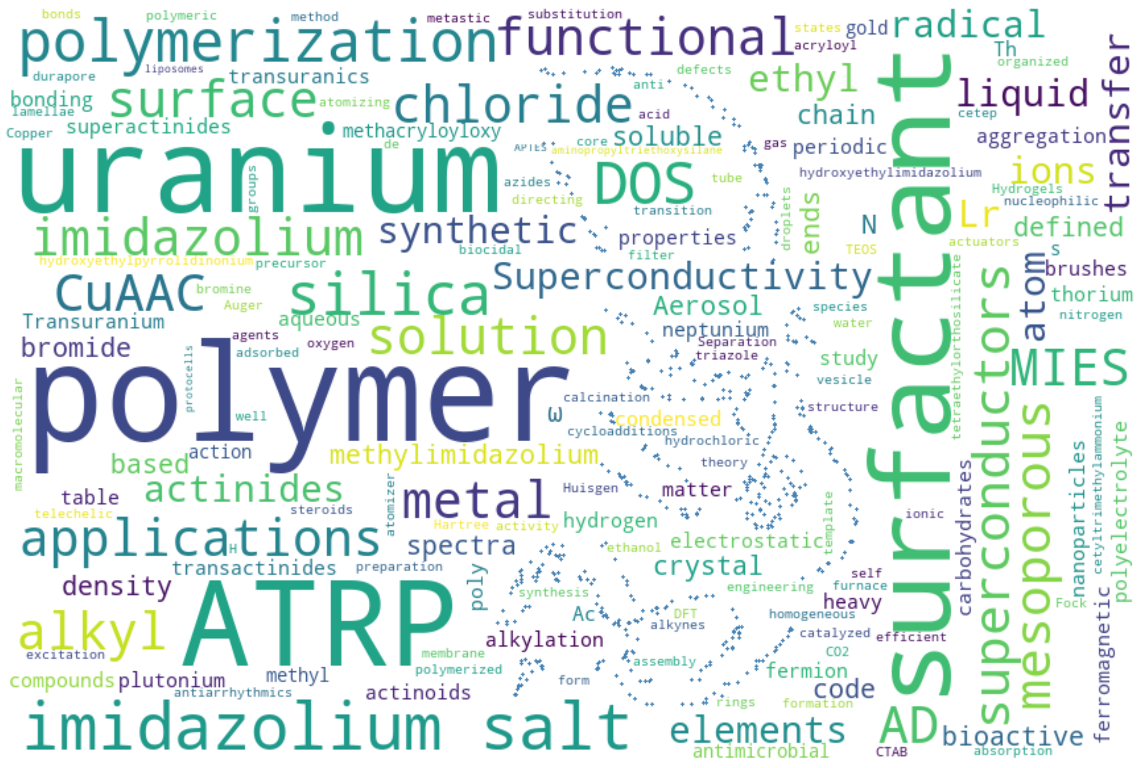


Figure 7: Wordcloud of entities

3.1.1 The research questions that the paper addresses

The paper that we will review is GLUCOSE, an Artificial intelligence mental model dataset that tries to understand common sense interference as humans do. The platform consists of large datasets of cause-effect knowledge that are grouped to create a general environment. The paper addresses two research questions. The first one is to measure and compare how effective is the Glucose dataset platform to extract cause and effect explanation. The focus is to learn as much as possible knowledge from daily circumstances. The second question that it addresses is why this implementation has advantages that separate it from other existing models and show that training on these type of data could lead models to match human capabilities.

3.1.2 What the paper proposes or what its main findings are

To answer the main finding firstly we have to mention what problems they found with current models and how they solved them. Overall, they found two main obstacles with existing models. The first one was that there was a lack of data about commonsense knowledge in the current industry, and they constructed a platform that could gather that type of data. The second obstacle was how to convert and use this knowledge to an artificial intelligence machine. They found that by constructing a model that used the proposed datasets, machines were able to extract contextual knowledge that helped capture information about entities, locations, causes, etc. All this information was presented in 10 dimensions and could be used as input to improve deep neural networks performance in everyday tasks. Using many different models showed that training in the GLUCOSE dataset, the models improved by 50%, could make more generalized predictions and learn specific cause and effect rules.

3.1.3 The experimentation used in the paper

To compare the coverage of the dataset knowledge, they compared it with two other datasets named ATOMIC[7] and ConceptNet[8]. They converted the different dimensions of the datasets to relations like previous datasets. Next, they made different queries for each resource and measured the coverage included from that query. They showed that their dataset could capture additional information than the other two. The dimensions were then evaluated by humans through the platform to compare system performance. The following experiment was done to evaluate the predictions of different neural network models by using Glucose data as inputs. Three Pretrained language model were used: the One-sided Generation model, the Full Rule Generation model and the Encoder-Decoder Model. They also compared the result with human evaluation by different judges. The result showed that for each dimension, the Encoder- Generation model performed nearly the same as humans, which was shown in the very similar scores. They also showed that fine-tuning the models on the datasets improved the overall score. Last, they experimented with baseline models and showed that the results were incapable of extracting this contextual knowledge.

3.1.4 Critique: the limitations of their proposed methods and experiments

To evaluate the dataset, they experimented by measuring the performance of the models and tried to predict each dimension separately in one sentence. By doing so, they recognised that each dimension should be able to be explored together or even in multiple sentences. This would open an opportunity to make better predictions for different natural language processing downstream tasks, which would achieve better overall accuracy. The second limitation that they agree with is that humans can be prone to biases because data are evaluated in a static judgment by humans. Therefore a new way of evaluation should be used in order to remove this unwanted bias.

3.1.5 Any related pieces of literature that address the same problem, that the paper mentions or that you have found yourself

As we described above, there were other datasets ATOMIC and CONCEPTNET, that use similar methods for gathering this context information. There were some differences in the way that they were constructed, but the idea was similar. They also mentioned in the journal that different neural language models could be trained on this data. An example of this is showed in the multitask learner by Radford et al. [9] which described a sizeable pre-trained transformer that can be used on these datasets for multiple downstream tasks and achieve excellent performance.

3.1.6 What you would do try next if you were working on this problem in future

The paper described only ten static semi-structured entries in the dataset, thus making it very task-specific. What I would try to do is, explore more of these relations and rules by using a dependency parser that could find some additional rules and add additional feature to the dataset. Another design that I would suggest would be to apply a pre-trained model like BERT for the downstream task. This introduces inductive transfer learning as a human does to solve everyday tasks. We could apply multi-task learning, which combines inductive bias across tasks, and we could join many tasks together and fine-tuned them. By doing this, we would combine all the dimensions in the GLUCOSE dataset. The model would learn general features from more data, and this could also reduce overfitting.

References

[1] Augenstein, I., Das, M., Riedel, S., Vikraman, L., & McCallum, A. (2017). Semeval 2017 task 10: Scienceie-extracting keyphrases and relations from scientific publications. arXiv preprint arXiv:1704.02853.

[2] Nltk.Tag.Crf — NLTK 3.6 Documentation. https://www.nltk.org/_modules/nltk/tag/crf.html. Accessed 14 Apr. 2021.

[3] LSTM — PyTorch 1.8.1 Documentation. https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html. Accessed 14 Apr. 2021.

[4] Brownlee, Jason. "Failure of Classification Accuracy for Imbalanced Class Distributions." Machine Learning Mastery, 31 Dec. 2019, https://machinelearningmastery.com/failure-of-accuracy-for-imbalanced-class-distributions/.

[5] "Dependency Parsing." CoreNLP, https://stanfordnlp.github.io/CoreNLP/depparse.html. Accessed 14 Apr. 2021.

[6] WordCloud for Python Documentation. Wordcloud 1.8.1 Documentation. https://amueller.github.io/word_cloud/. Accessed 14 Apr. 2021.

[7] Sap, M., Le Bras, R., Allaway, E., Bhagavatula, C., Lourie, N., Rashkin, h., ... & Choi, Y. (2019, July). Atomic: An atlas of machine commonsense for if-then reasoning. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, No. 01, pp. 3027-3035).

[8] Speer, R., Chin, J., & havasi, C. (2017, February). Conceptnet 5.5: An open multilingual graph of general knowledge. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 31, No. 1).

[9] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.

[10] Mostafazadeh, N., Kalyanpur, A., Moon, L., Buchanan, D., Berkowitz, L., Biran, O., & Chu-Carroll, J. (2020). Glucose: Generalized and contextualized story explanations. arXiv preprint arXiv:2009.07758.