# FINAL YEAR B.TECH. SEMESTER INTERNSHIP REPORT

A report submitted in partial fulfilment of the requirements for the Award of Degree of

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

By

**(Abhibha Gupta)**

**Enrolment Number: BT17CSE020**

**Under Supervision of**

(Dr. Sagar Sunkle), (Senior Scientist), (TRDDC, Pune)

भारतीय सूचना प्रौद्योगिकी संस्थान, नागपूर

## INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, NAGPUR

**(An Institution of National Importance by Act of Parliament)**

**BSNL RTTC, Near TV Tower, Besides Balaji Temple, Seminary Hills, Nagpur-440006**

# ACKNOWLEDGEMENT

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them. I am highly indebted to Dr Sagar Sunkle and Miss Krati Saxena for their guidance and constant supervision as well as for providing necessary information regarding the project and also for their support in completing the project. They made sure that it was a wholesome learning experience for me, even in remote mode. Finally, I would like to express my gratitude towards my parents for their invaluable guidance and ongoing encouragement which has sustained my efforts at all the stages of project development.

# EXTRACTION OF INFORMATION USING STATE OF THE ART TECHNIQUES IN THE CONTEXT OF VARIOUS I2I POCSs

Abhibha Gupta

Dr. Sagar Sunkle

October 31, 2020

# Introduction

One of the main challenges of **Natural language processing** (**NLP**) is converting unstructured data into a structured format. Structured data can then in turn be used to create knowledge graphs, train other machine learning models, etc. A widely used method for this is **Named Entity Recognition (NER)**[1]. It involves the identification and extraction of some particular entities of interest in the text. Many trained models available are able to extract names of people, places and organization with great accuracy.

In our case we had a corpus of **600 research texts** regarding the different types of coatings applied on steel. Our task was to populate a **domain model** consisting of predefined entities like ingredients used, there quantities, the conditions under which the steel coating process took place, coating type, substrate, etc. We started by creating our own corpus to train a NER model that could identify some of the basic entities like occurrences of molecules, processes, conditions, actions and quantities. Using the trained model we implemented an **Annotation tool cum Search tool** where one could perform queries, that would perform a search on the database and return focused results. This enabled us to populate our domain model without training an NER for all entities. The workflow has been described in the subsequent sections.

[1] https://en.wikipedia.org/wiki/Named-entity_recognition

# Index

# Part Of Speech (POS) Tagging

According to Wikipedia, 'In corpus linguistics, **part-of-speech tagging** (**POS tagging** or **PoS tagging** or **POST**), also called **grammatical tagging** is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition and its context.' There are 9 main categories namely,
noun, verb, article, adjective, preposition, pronoun, adverb, conjunction, and interjection.

For Example, in the sentence '**she sell sea shells on the seashore**' the POS tags for each word are,

| Word | POS Tag | Meaning |
|---|---|---|
| She | PRP | Personal pronoun |
| Sells | VBZ | Verb |
| seashells | NNS | Plural noun |
| On | IN | Preposition |
| the | DT | Determiner |
| Seashore | NN | Common noun |

Popular part of speech taggers for the English language include the **NLTK POS tagger**[2] and the **Spacy POS tagger**[3]. Popular datasets include the **POS Penn dataset**[4] and the **NLTK Treebank dataset**[5]**.**

For exploration purposes we used the **Conll**, **Penn** and the **Kaggle** datasets. The data entries were in the following format

[2] http://www.nltk.org/book/ch05.html

[3] https://spacy.io/usage/linguistic-features#pos-tagging

[4] https://www.sketchengine.eu/penn-treebank-tagset/

[5] https://www.nltk.org/howto/corpus.html#tagged-corpora

| Word | POS Tag |
|---|---|
| UN | NNP |
| Official | NN |
| Ekeus | NNP |
| heads | VBZ |
| for | IN |
| Baghdad | NNP |
| . | . |

## Online Machine Learning

According to Wikiwand[6], 'In computer science, **online machine learning** is a method of machine learning in which data becomes available in a sequential order and is used to update the best predictor for future data at each step, as opposed to batch learning techniques which generate the best predictor by learning on the entire training data set at once.' Online learning is a common technique used in areas of machine learning where it is computationally infeasible to train over the entire dataset, requiring the need of out-of-core algorithms. It is also used in situations where it is necessary for the algorithm to dynamically adapt to new patterns in the data, or when the data itself is generated as a function of time, e.g., stock price prediction. **Creme**[7] is an open source library for online machine learning. **Scikit learn**[8] also provides some implementations to support out-of-core learning.

[6] https://www.wikiwand.com/en/Online_machine_learning#/Implementations

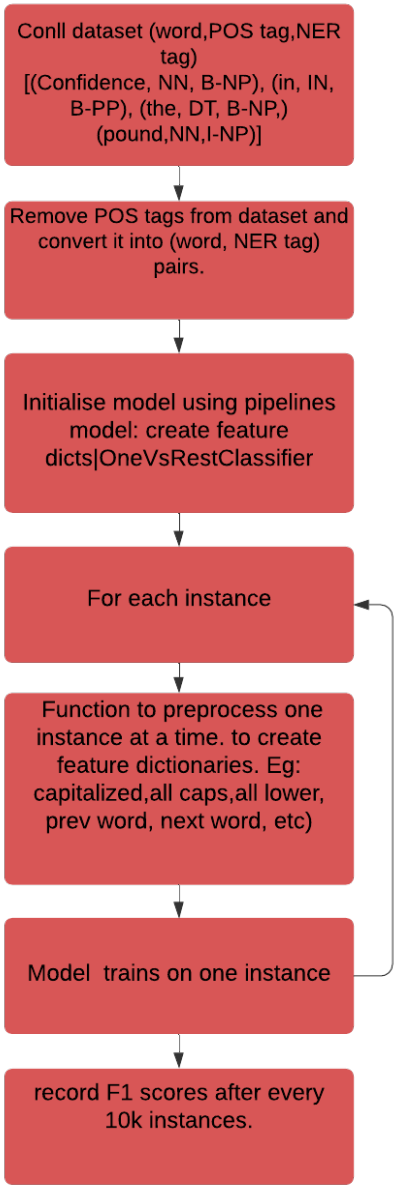
[7] https://github.com/creme-ml/creme

[8] https://scikit-learn.org/0.15/modules/scaling_strategies.html

# Exploration Work

## POS tagging

Our main aim was to perform NER on the steel coatings text to extract entities to populate our domain model. To perform NER on our coatings text we were required to train our own custom model that catered to our need. Since the task of POS tagging is close to NER, we started out by using different POS tagging datasets like the **Conll dataset**[9], the **Penn dataset**[10] and the **POS tagging dataset from Kaggle**[11]. We started out by exploring an open source library called **Creme**.




**Creme:** [12] is an open source library for online machine learning. Since it trains from an instance at a time, it is used to learn realtime from streaming data. We used different algorithms like AdaBoost[13], Naive Bayes[14], Random Forest[15], Decision Tree[16], The One Vs Rest Classifier[17], Ensemble Bagging Classifier[18] to experiment with the data. Out of these the One vs Rest Classifier (Classifier: Decision Tree classifier) gave the best result of a F1 score of 0.66 on the Conll dataset. We used

[9] https://www.clips.uantwerpen.be/conll2003/ner/

[11] https://www.kaggle.com/tarunpaparaju/jigsaw-competition-part-of-speech-tagging

[12] https://github.com/creme-ml/creme

[13] https://creme-ml.github.io/api-reference/ensemble/AdaBoostClassifier/

[14] https://creme-ml.github.io/api-reference/naive-bayes/GaussianNB/

[15] https://creme-ml.github.io/api-reference/tree/RandomForestClassifier/

[16] https://creme-ml.github.io/api-reference/tree/DecisionTreeClassifier/

[17] https://creme-ml.github.io/api-reference/multiclass/OneVsRestClassifier/

[18] https://creme-ml.github.io/api-reference/ensemble/BaggingClassifier/

the pipeline approach to train our classifiers.

**Pipeline Approach[19]:** The creators of Creme highly recommend to implement pipelines while training one's model. In this method every instance before being used by the model to train passes through some steps of preprocessing (in our case creation of feature arrays) before being fed into the machine learning algorithm.

Since we weren't getting satisfactory F1 scores we switched to other methods.

[19] https://creme-ml.github.io/user-guide/pipelines/

# Named Entity Recognition (NER)

According to Wikipedia, '**Named-entity recognition** (**NER**) (also known as **(named) entity identification**, **entity chunking**, and **entity extraction**) is a subtask of information extraction that seeks to locate and classify named entities mentioned in unstructured text into predefined categories such as person names, organizations, locations, medical codes, time expressions, quantities, monetary values, percentages, etc.' Popular trained taggers include the **Stanford NER Tagger**[20] and the **Spacy NER Tagger**[21]. Popular datasets include the **Conll NER 2003**[22] dataset**.**

For example, in the sentence, **Apple is owned by Steve Jobs and is based in California**. The named entities recognized by the Stanford NER tagger are as follows,

| Word | Entity Name |
|---|---|
| Steve Jobs | Person |
| Apple | Organisation |
| California | Location |

We used the Conll 2003 dataset for our exploration task. Note that words that don't fall in the above mentioned categories are marked as 'O' or other.

| Word | Entity Name |
|---|---|
| CRICKET | O |
| - | O |
| LEICESTERSHIRE | I-ORG |
| TAKE | O |
| OVER | O |
| AT | O |

[20] https://nlp.stanford.edu/software/CRF-NER.shtml

[21] https://spacy.io/usage/linguistic-features#named-entities

[22] https://www.clips.uantwerpen.be/conll2003/ner/

| Word | Entity Name |
|---|---|
| TOP | O |
| AFTER | O |
| INNINGS | O |
| VICTORY | O |
| . | O |

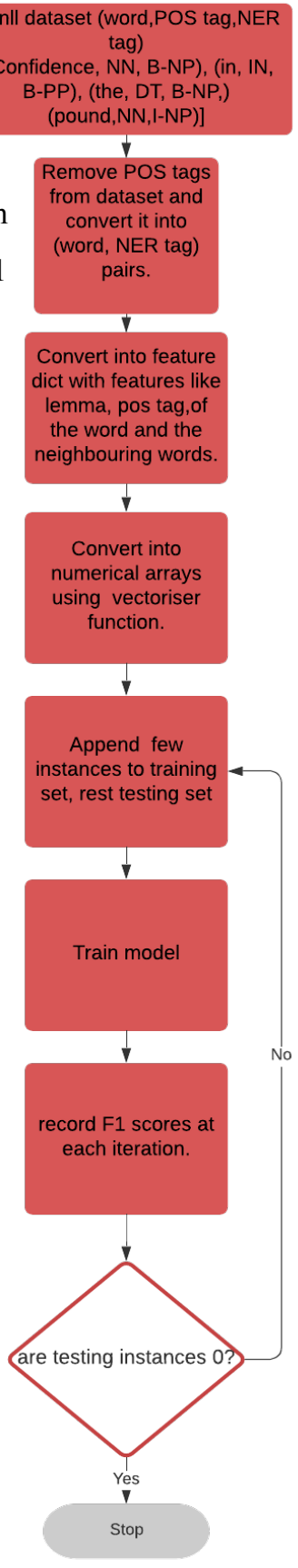
# Exploration work

## NER tagging

We now started with the task of NER on the **Conll dataset** [23] by exploring **Scikit learn.**

**Scikit learn:** It provides limited support for incremental machine learning. We used the partial fit method[24] on the Perceptron[25] algorithm to train the model in an incremental manner and got an aggregate F1 score of **0.96.**

**Incremental machine learning:** In this method after preprocessing and splitting the dataset into train-test sets we perform the training process by first taking a few instances in the training set and the rest in the testing set. For every iteration while training, we append a few instances to the training set and reduce the testing instances. We observe the F1 scores after every iteration. This is especially useful in cases where we have unannotated data and we want to find out how much data we need to annotate to get an optimal score. Usually after some iterations the F1 score reaches a threshold value and it does not increase by a considerable amount. At that point we can conclude that the model isn't learning anything new. If the trend of data for the rest of the dataset is the same then we don't need to annotate any more data.




[23] https://www.clips.uantwerpen.be/conll2003/ner/

[24] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html#sklearn.linear_model.Perceptron.partial_fit

[25] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html#sklearn-linear-model-perceptron

# Preparation of Dataset

## MW and AMCPE Structures

To train our own custom NER model we started by preparing a dataset. For now our basic target entities were molecules, quantities, actions, conditions and processes. We identified two particular relations of interest,

**MW structures:** Molecule-Weight/Quantity Structures and

**AMCPE structures:** Action-Molecule-Condition-Process-Extra Structures

The following example shows the output generated from each of the libraries **NLTK POS tagger**[26] and **Chemical Tagger**[27]

**Sentence:** To generate a Ni–P/Ni–B duplex coating, the surface of the HVOF-sprayed 420 martensitic stainless steel coating was prepared for plating by mechanical grinding, acetone degreasing and etching in a 30 vol.%HCl solution for 1 min.

**MW - Structures:**

**Chemical Tagger**

**Output:**<MOLECULE><OSCARCM><OSCAR-CM>**Ni-P**</OSCAR-CM><DASH>/</DASH><OSCAR-CM>**Ni-B**</OSCAR-CM></OSCARCM></MOLECULE>
<MOLECULE><OSCARCM><OSCAR-CM>**acetone**</OSCAR-CM></OSCARCM></MOLECULE>
<MOLECULE><QUANTITY><PERCENT><CD>**30**</CD><NN>**vol.**</NN><NN-PERCENT>**%**</NN-PERCENT></PERCENT></QUANTITY><OSCARCM><OSCAR-CM>**HCl**</OSCAR-CM></OSCARCM></MOLECULE>

[26] http://www.nltk.org/book/ch05.html

[27] http://chemicaltagger.ch.cam.ac.uk/

<TimePhrase><IN-FOR>**for**</IN-FOR><CD>**1**</CD><NN-TIME>**min**</NN-TIME></TimePhrase>

**Obtained entities: Molecules:** Ni-P/Ni-B, acetone, HCl

**Quantity: - 30 vol.%**

## AMCPE Structures:

**Chemical Tagger:** Ni-P/Ni-B, acetone, HCl, for 1 min

**NLTK POS Tagger:** plating

**Obtained entities:**

**Molecules:** Ni-P/Ni-B, acetone, HCl

**Actions:** plating

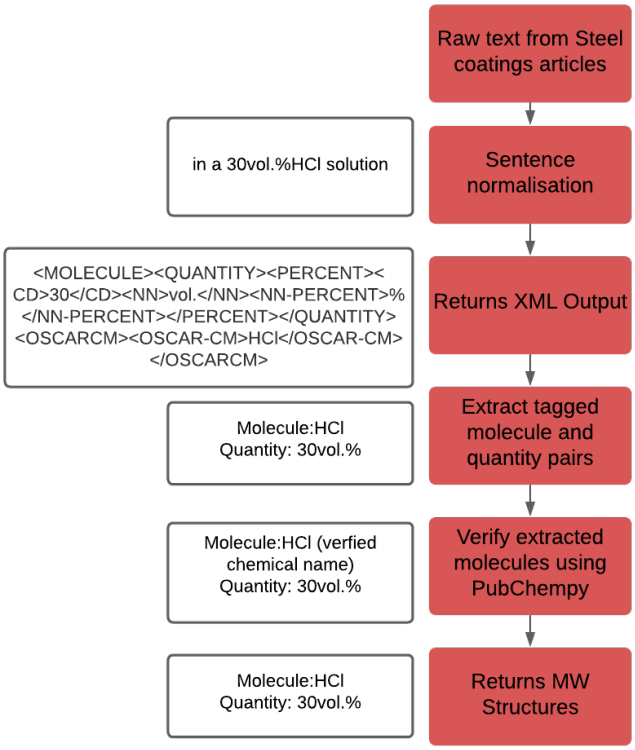
**Conditions:** for 1 min

**Quantity: 30 vol.%**

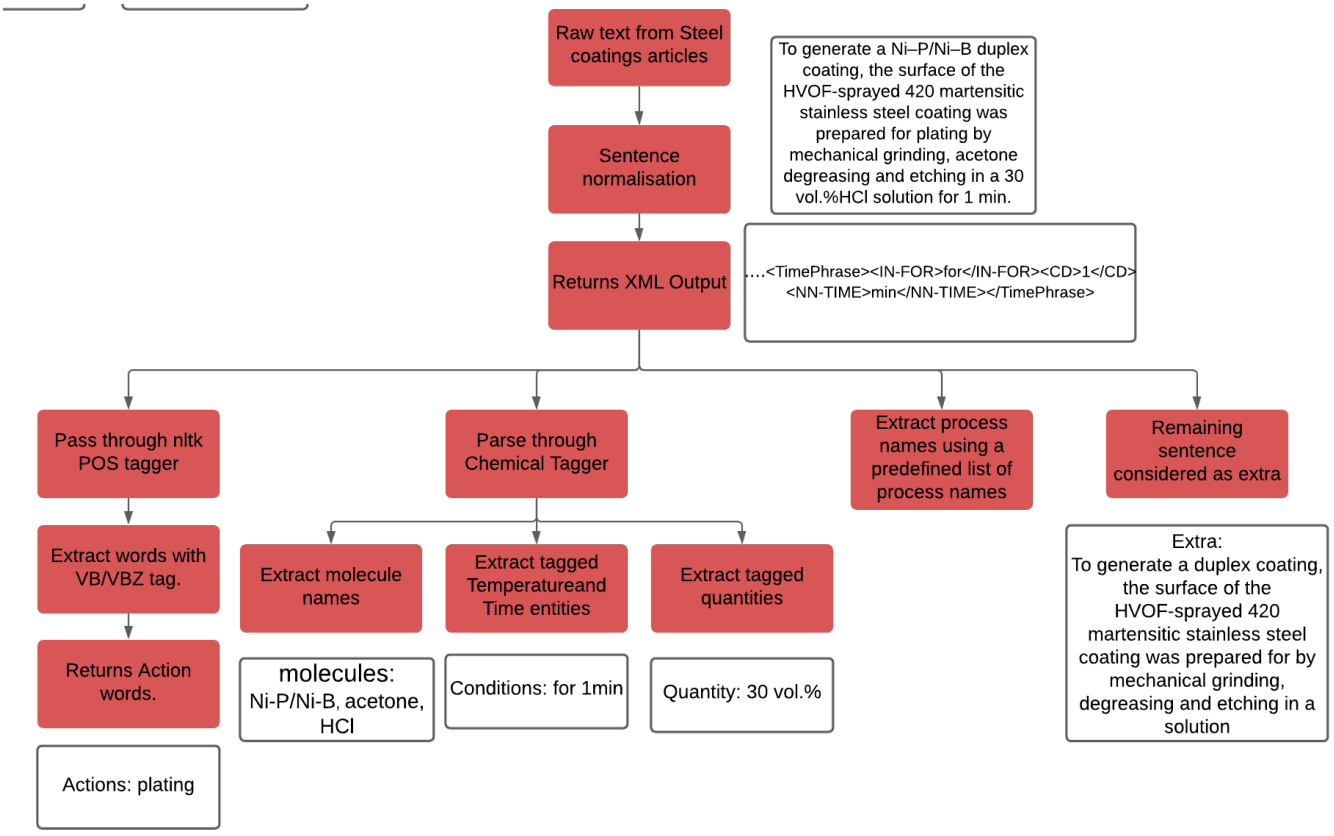
**Process: -**

**Extra:** To generate a duplex coating, the surface of the HVOF-sprayed 420 martensitic stainless steel coating was prepared for by mechanical grinding, degreasing and etching in a solution

## MW structures



# AMCPE Structures



# Preparation of Dataset

## Consolidated dataset

Since Chemical Tagger wasn't giving optimum results we incorporated the help of more open-source libraries like **ChemData extractor**[28], **Quantulum**[29] and **AllenNLP**[30] to extract the desired entities. We wrote heuristics  and used the libraries like to extract the required entities from raw coatings text.

The following example shows the output generated from each of the libraries mentioned above.

**Sentence:** To generate a Ni–P/Ni–B duplex coating, the surface of the HVOF-sprayed 420 martensitic stainless steel coating was prepared for plating by mechanical grinding, acetone degreasing and etching in a 30 vol.% HCl solution for 1 min.

**ChemData extractor:** Ni-P/Ni-B, acetone, HCl

**Allen NLP:**

**Verb:** generate, sprayed, prepared, plating, degreasing, etching

**Arg0:** by mechanical grinding

**Arg1:** a Ni – P / Ni – B duplex coating , the surface of the HVOF - sprayed 420 martensitic stainless steel coating was prepared for **plating** by **mechanical grinding** , acetone **degreasing** and etching in a 30 vol.%HCl solution for 1 min, the surface of the HVOF - sprayed 420 martensitic stainless steel coating, for 1 min|in a 30 vol.%HCl solution

**Arg2:** -

**Quantulum:** 420, 30 vol.%, 1 min

**Obtained Entities:**

**Molecules:** Ni-P/Ni-B, acetone, HCl

---

[28] http://chemdataextractor.org/

[29] https://github.com/marcolagi/quantulum

[30] https://github.com/allenai/allennlp

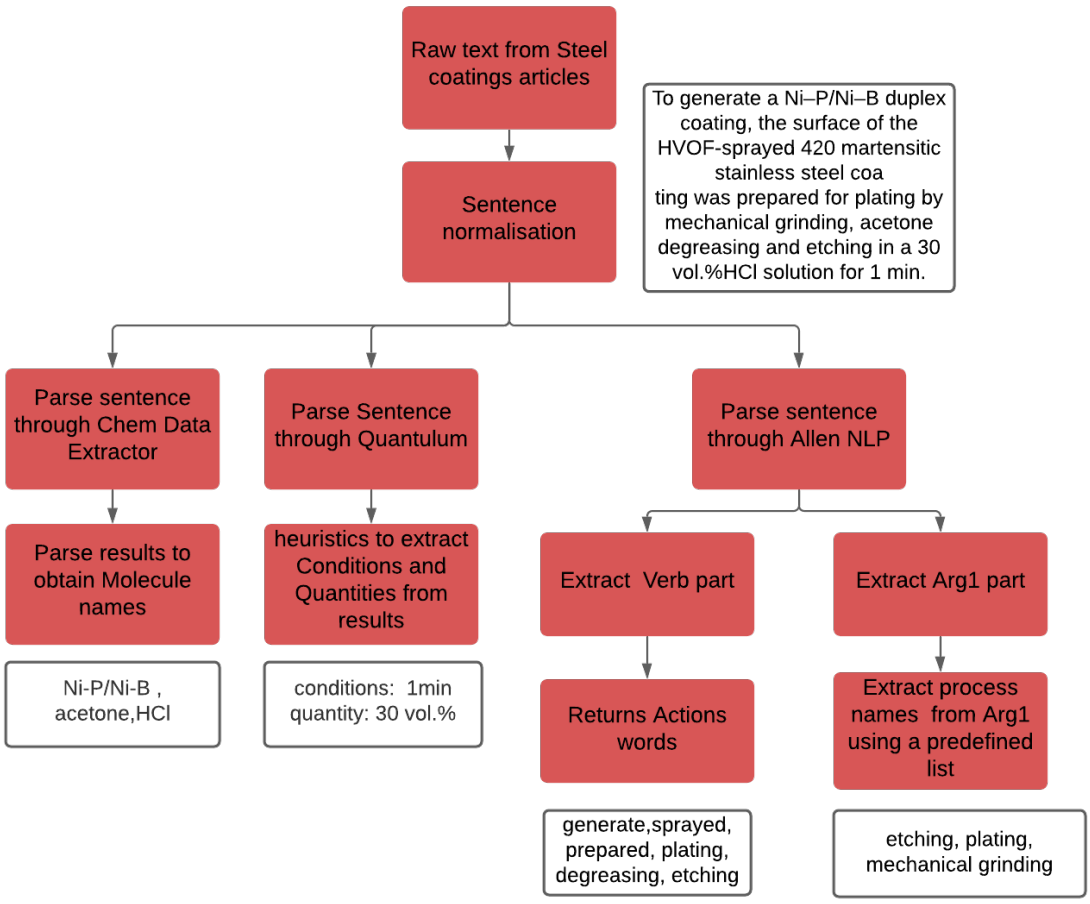
**Quantity:** 30 vol.%

**Actions:**generate, sprayed, prepared, plating, degreasing, etching

**Process:** etching, plating, mechanical grinding

**Conditions:** 1 min

# Consolidated dataset (AMCPW)




# Manual Annotation of Dataset

Since there were some errors in the extraction process, we manually annotated about 100 files using **Doccano**[31], an open source library for text annotation. The number of instances for each entity type was as follows:

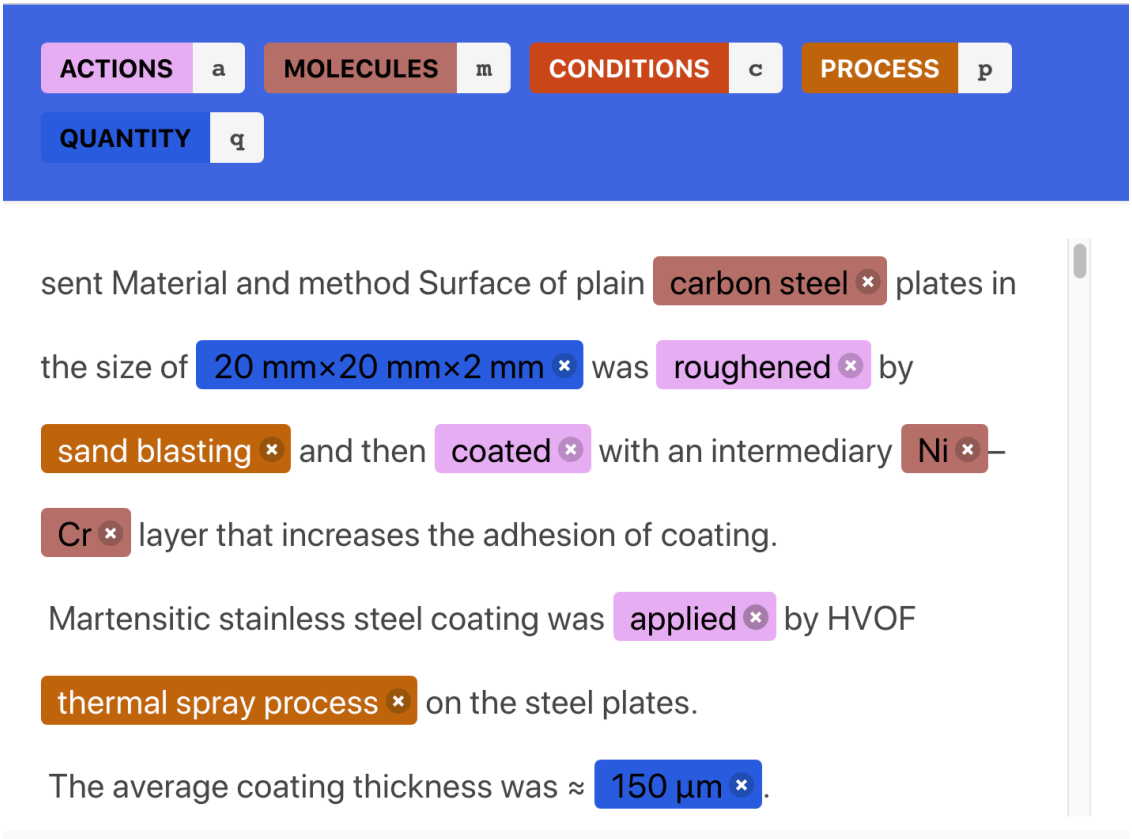
**Actions:** 2107

**Molecules:** 954

**Quantities:** 439

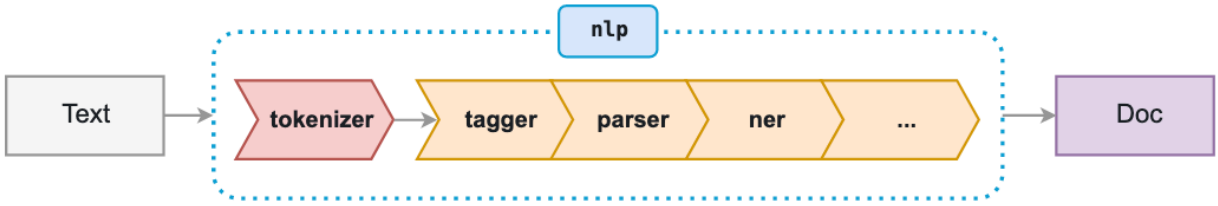
**Processes:** 242

**Conditions:** 198




[31] http://doccano.herokuapp.com/

**Spacy:** Spacy's default model comes with its own processing pipeline consisting of a tagger, a parser and an entity recognizer. A novel bloom embedding strategy with subword features is used to support huge vocabularies in tiny tables. Convolutional layers with residual connections, layer normalization and maxout non-linearity are used, giving much better efficiency than the standard BiLSTM solution.This makes Spacy's pretrained models **10× smaller, 20% more accurate, and even cheaper to run** than other bulkier deep learning models.

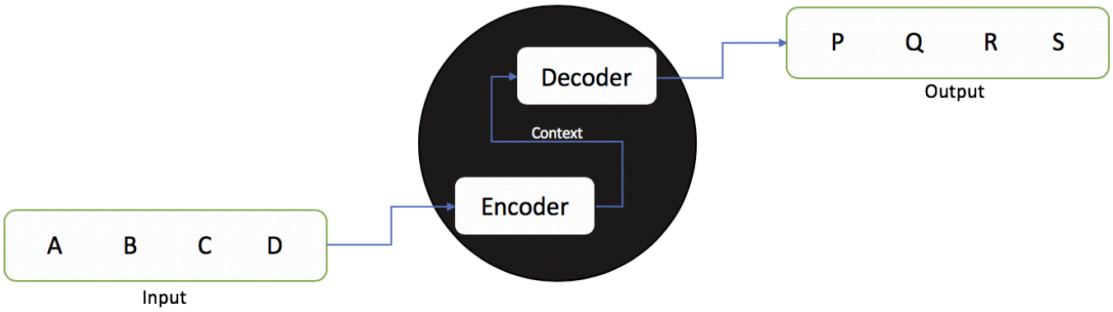



**CRF Suite:** CRFsuite is an implementation of Conditional Random Fields (CRFs) [Lafferty 01][Sha 03][Sutton] for **labeling sequential data**. It provides the following features:

- Fast training and tagging.
- Simple data format for training and tagging
- State-of-the-art training methods
- Forward/backward algorithm using the scaling method
- An efficient file format for storing/accessing CRF models

**Seq2Seq Model:** A Sequence2Sequence (Seq2Seq) model is a model that takes a sequence of items (words, letters, time series, etc) and outputs another sequence of items. The model is composed of an *encoder* and a *decoder*. The encoder captures the **context** of the input sequence in the form of a **hidden state vector** and sends it to the decoder, which then produces the output sequence. Since the task is sequence based, both the encoder and decoder

tend to use some form of RNNs, LSTMs, GRUs, etc.



# Beginning-Inside-Outside (BIO)Tagging

We have used BIO tagging for our dataset to tackle the problem of multiword entities. It is a common tagging format for tagging tokens in a chunking task in computational linguistics (ex. named-entity recognition. The **B-** prefix before a tag indicates that the tag is the beginning of a chunk, and an **I-** prefix before a tag indicates that the tag is inside a chunk. The **B-** tag is used only when a tag is followed by a tag of the same type without O tokens between them. An **O** tag indicates that a token belongs to no entity / chunk. For example, for the sentence **HVOF Thermal spray process was used for etching stainless steel using 1M HCl**, the tagging will look like this:

| Word | Entity Name |
|---|---|
| HVOF | B-Process |
| Thermal | I-Process |
| Spray | O-Process |
| Process | O |
| Was | O |
| Used | O |
| For | O |
| Etching | Process |
| 1M | Quantity |
| HCl | Molecule |
| Stainless | B-Molecule |
| Steel | O-Molecule |

# Exploration Work

## NER Tagging

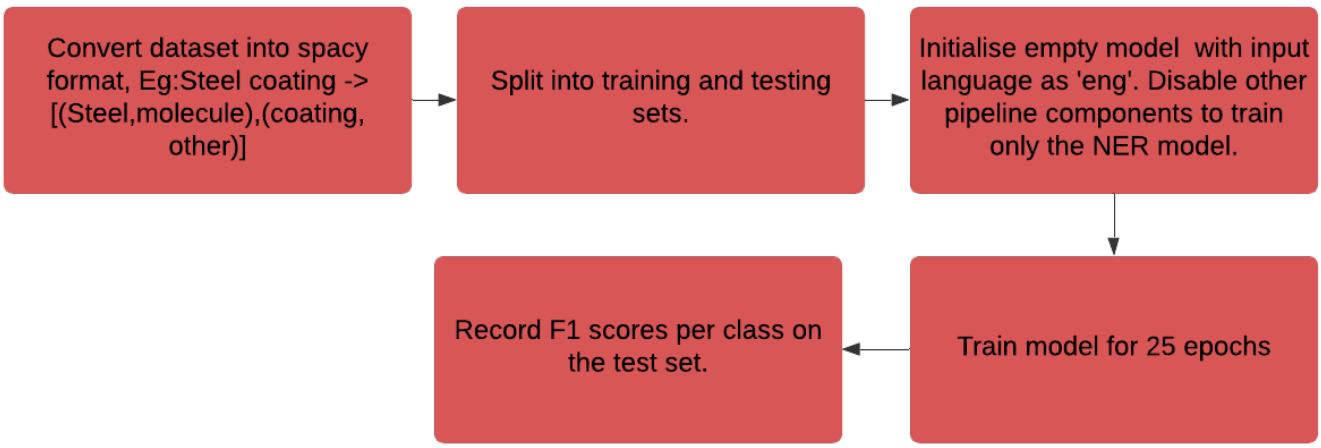
We now start with our task of training a custom NER model on the **Annotated Coatings Dataset**. We experiment with **Spacy NER**, **CRF Suite** and a **Seq2Seq model** using Keras.

**Spacy NER:** For our task of NER we disabled the other pipeline components so that the model trained faster. We fine tuned the pre-trained model on the Annotated coatings dataset. A consolidated **F1 score of 0.83** was achieved against a baseline of 0.69.

Table 1

| Class name | F1 score |
|---|---|
| **Actions** | 0.64 |
| **Molecules** | 0.57 |
| **Quantity** | 0.53 |
| **Process** | 0.50 |
| **Conditions** | 0.53 |
| **Overall** | 0.83 |

Figure 1

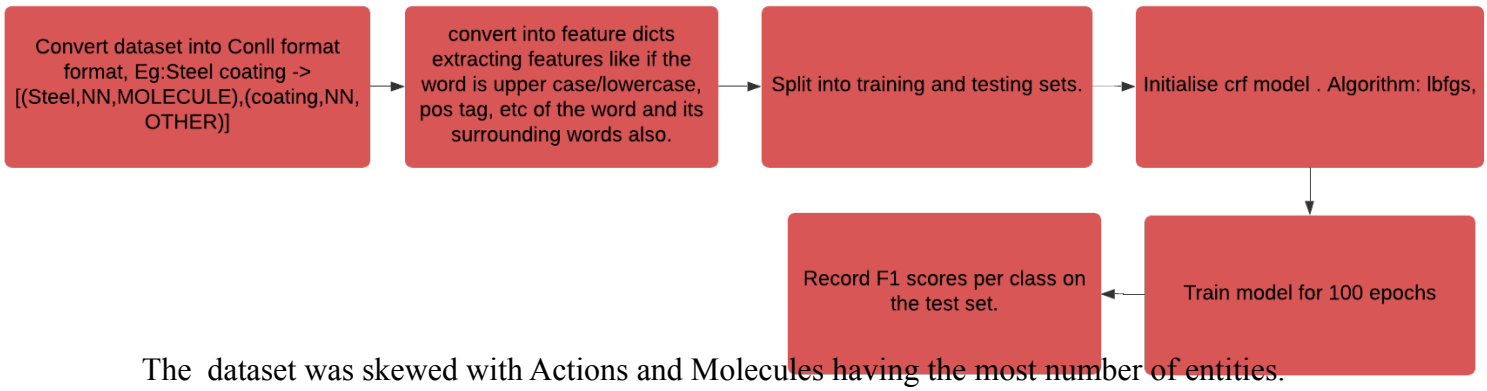


**CRF Suite:** Sklearn provides a wrapper for Crf Suite, an implementation of conditional random fields. Crf's are good in considering the context into account during the training process. The algorithm used is similar to gradient descent. For each word and its neighboring words in the training instance we created feature arrays by extracting features like POS tags, whether the word is in upper case or lower case, etc. We got a consolidated **F1 score of 0.88** against a baseline of 0.65.

Table 2

| Class name | F1 score |
|---|---|
| **Actions** | 0.95 |
| **Molecules** | 0.43 |
| **Quantity** | 0.13 |
| **Process** | 0.167 |
| **Conditions** | 0.00 |
| **Overall** | 0.88 |

Figure 2



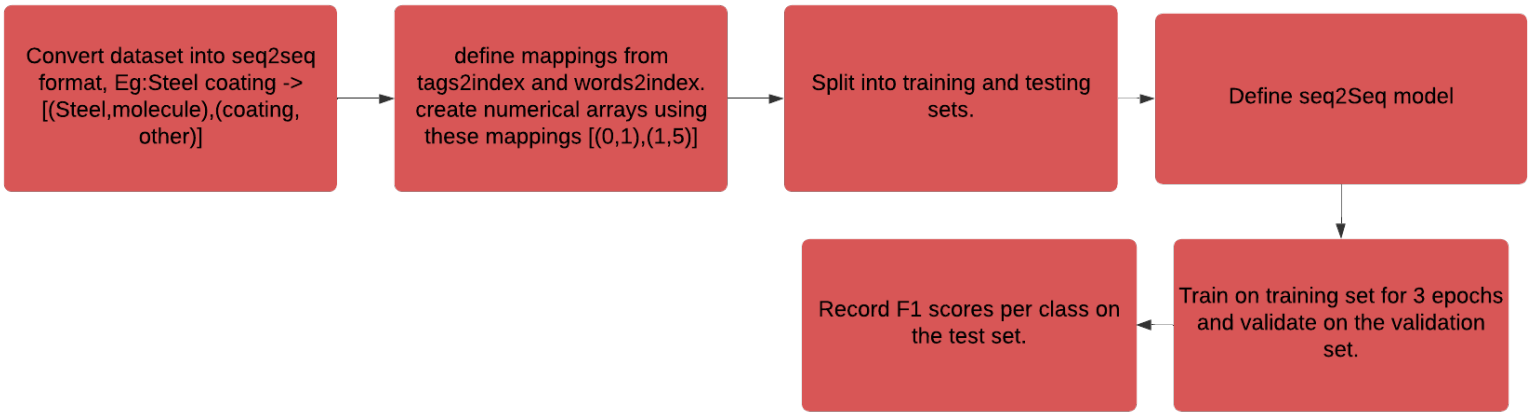

The dataset was skewed with Actions and Molecules having the most number of entities. The dataset thus formed will be referred to as the **'Annotated Coatings dataset'** from here on.

**Seq2Seq using Keras on Tensorflow**: We also experimented using deep learning techniques by implementing a Seq2Seq Model using Keras. The network was a Bi-directional LSTM with an added Elmo embedding layer as the input. We set a non zero value for dropout to ensure that the model does not overfit on our dataset. Since we require a lot of data for deep learning techniques, the model couldn't perform well and gave a consolidated **F1 score of 0.68**.

Table 3

| Class name | F1 score |
|---|---|
| **Actions** | 0.93 |
| **Molecules** | 0.64 |
| **Quantity** | 0.0 |
| **Process** | 0.0 |
| **Conditions** | 0.0 |
| **Overall** | 0.68 |

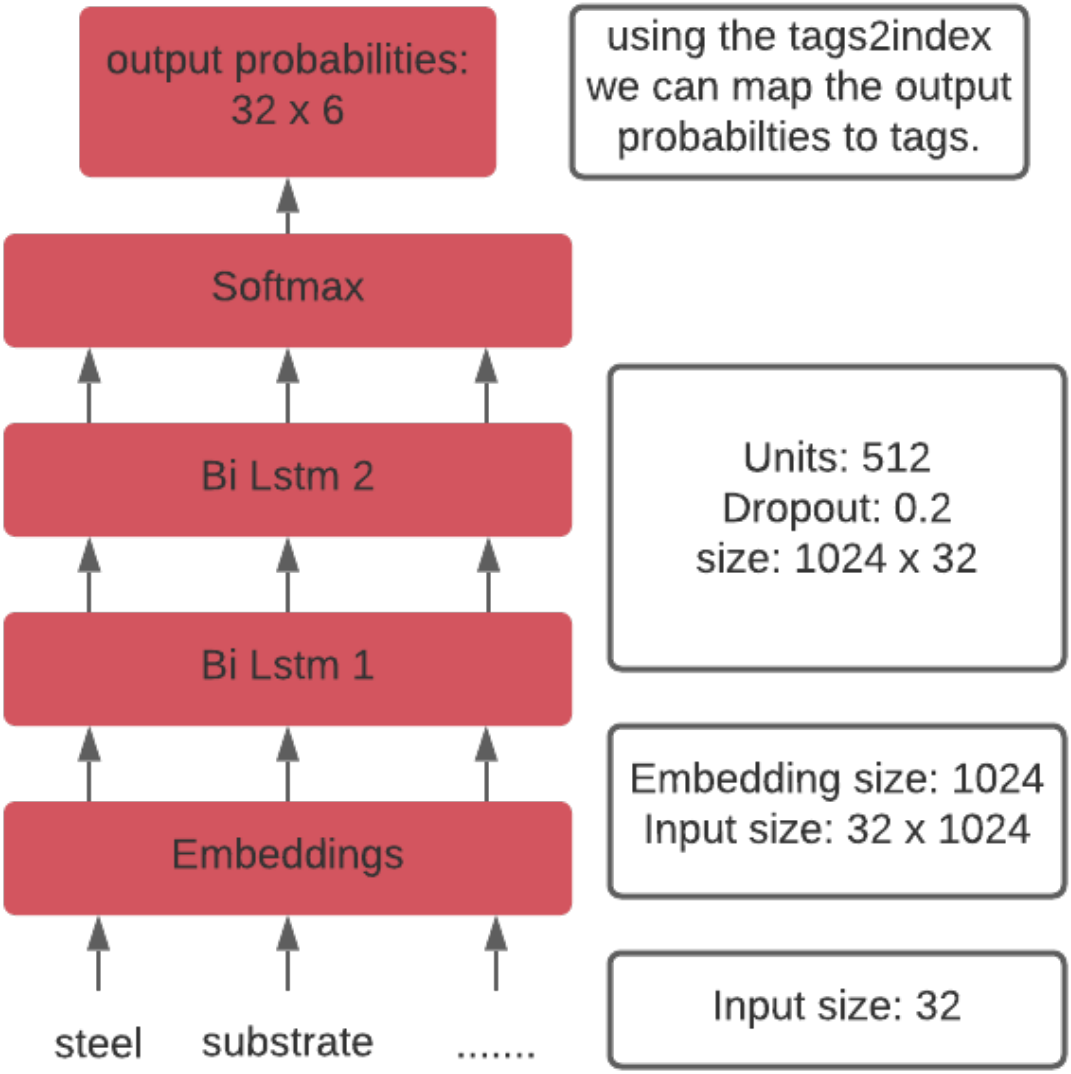
Figure 3.1



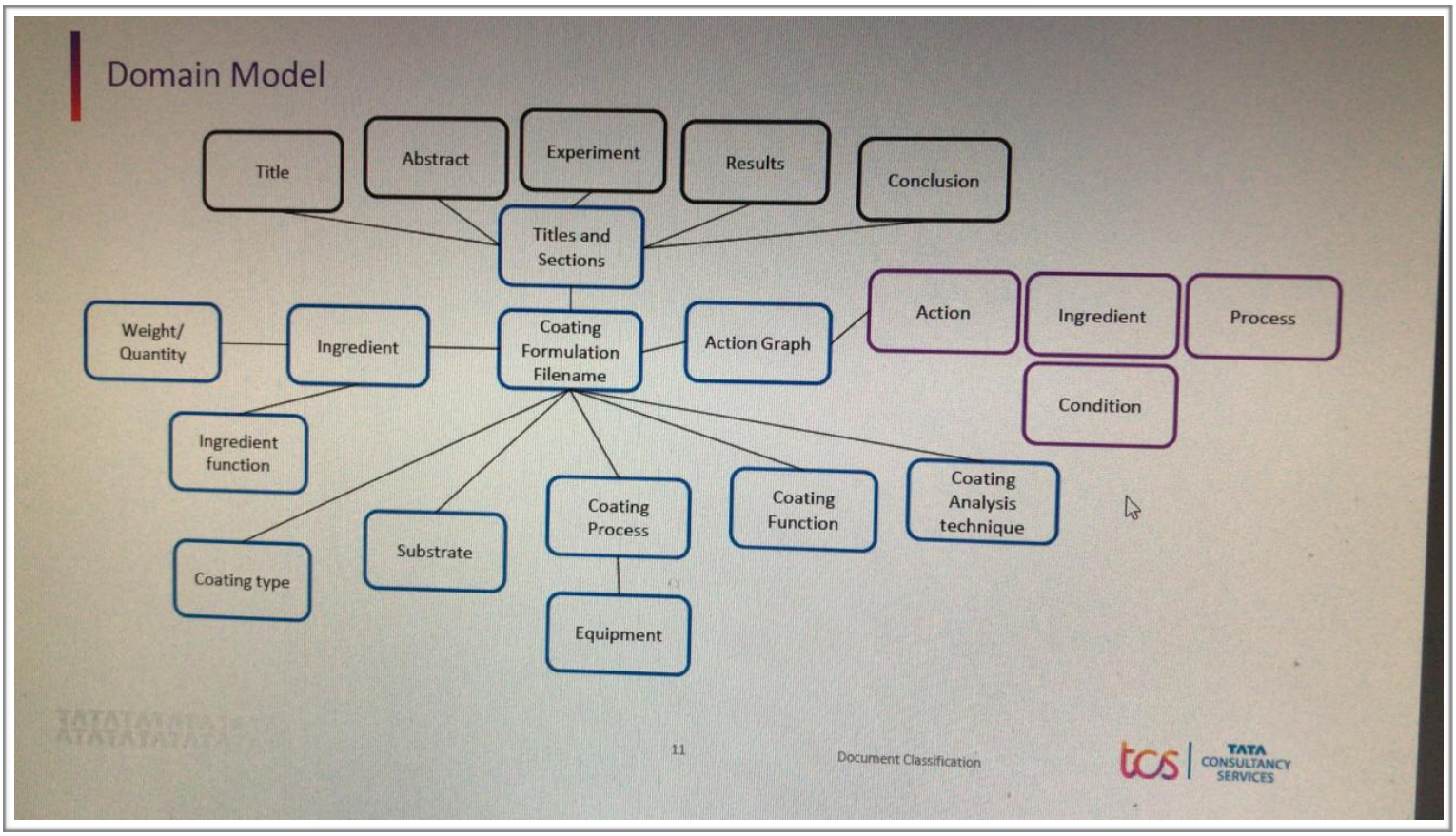
We used the trained **Spacy NER model** as our final model as it was able to detect all types of entities with good accuracy and was able to handle the skewness in the dataset well.

**Seq2Seq Model Description**

Figure 3.2



# Domain Model



## Search and Annotation Tool

Our final aim was to extract the entities given in the domain model. Till now we were able to extract the basic entities. We figured out that the basic entities along with some keywords could help us populate our domain model. For example, To identify 'substrates' in a coating process, we could search for the sentences where the word 'substrate' would occur. Then we could manually annotate the name of the substrate in that sentence. This was the motivation behind the Search cum Annotation tool.

**Search Tool:** The search tool provides support for firing queries in different formats to the end user. The motivation behind this was that the end user could query and observe trends and understand how text was structured in the dataset. Once identified the user can fire specific queries that would return results closest to what he/she wants.

**Queries Supported:** The user can write queries in 4 different formats
**Boolean:** It consists of word(s) and there different combinations
For example, The user can search for
**A single word:** 'substrate'
**Multiple words:** 'stainless steel'
**Multiple word combinations:** 'stainless|carbon steel' (here it searches for the combinations 'stainless steel' and 'carbon steel'

**Field:** In addition to words, one can also search for a particular lemma, any predefined entity from molecules, actions, processes, quantities and conditions or words having a specific POS (Part of speech) tag. The user can fire single, multiple and multiple word combinations as in the case of Boolean queries.

For Example: The user can search for

**Single/Multiple/Mutiple word combos:** ‘word=substrate’/ ‘word=stainless steel’/ ‘word= stainless|word=carbon word=steel/ ‘lemma=coating/ tag=‘NN’/ Using|by entity=Process’

**Multiple pairs:** entity=Molecules entity=Process (searches for sentence with occurrence of either one of the entities)

**Capture Groups:** These enable us to capture any field query in a variable.

For Example:

Ingredient: entity=Molecules process_name:entity=process

Substrate:word=stainless|word=carbon word=steel

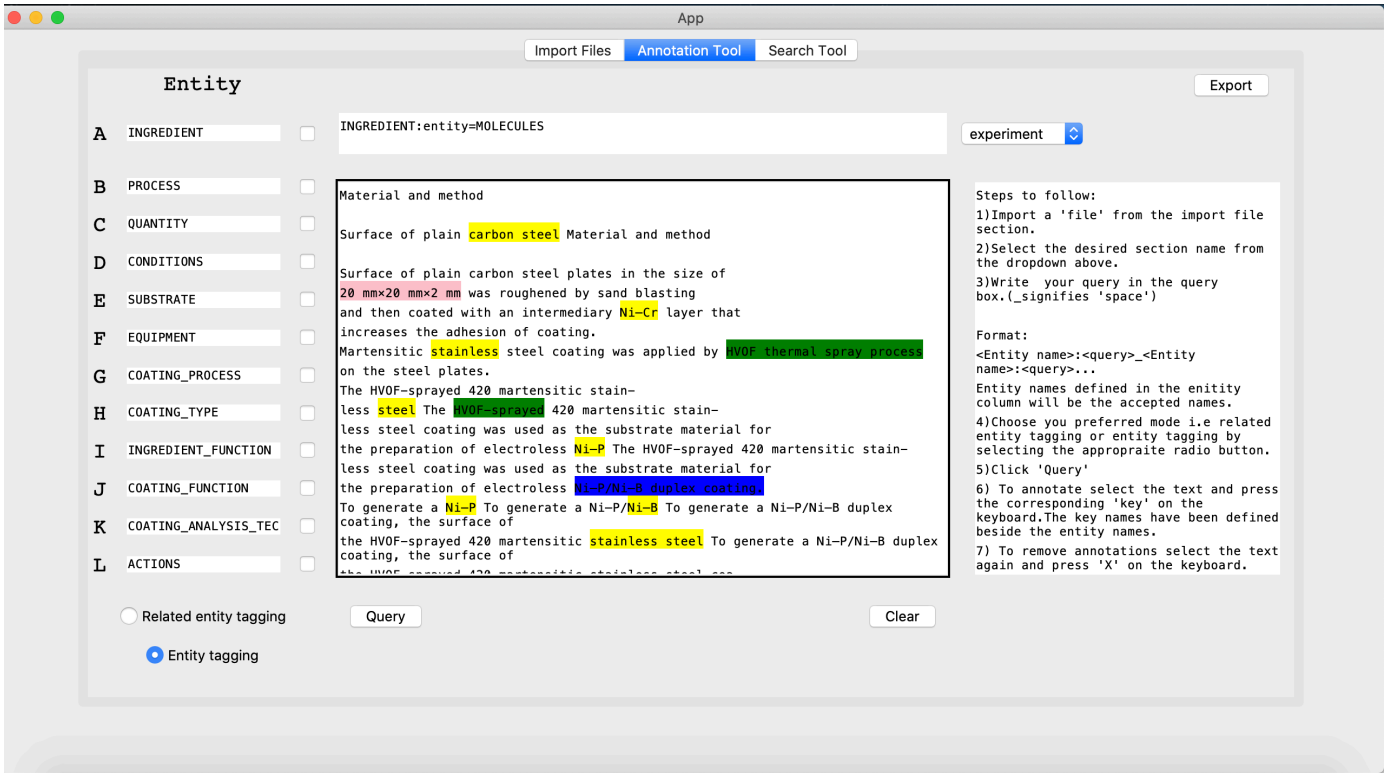
**Sequential:** In the above mentioned queries, the order of occurrence of words wasn’t considered, but in sequential queries, one can search for Boolean/Field/Capture groups in a particular order.

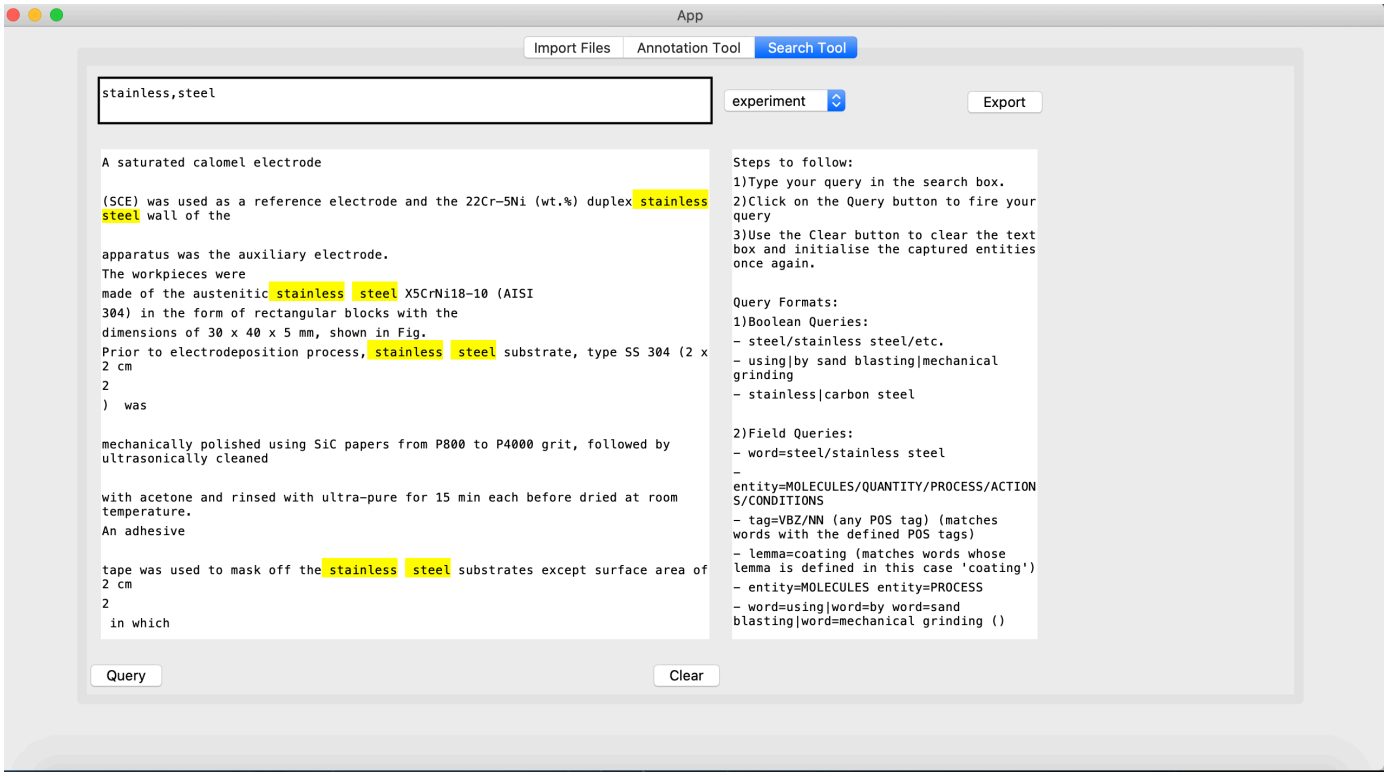
For example:

stainless, steel (Here only those sentences will be returned who have the word stainless followed by steel, so the order will be strictly followed)

**Annotation Tool:** The annotation tool provides support to search and annotate predefined entities that are outlined in the domain model. Support is provided to the user to annotate standalone entities or define relationships between them by annotating them in sequence. For example, the user can annotate names of substrates or can annotate molecule-quantity pairs. The annotations can be exported in CSV format.

# Annotation Tool




# Query Tool

# Code Base References

https://spacy.io/usage/training#ner

https://medium.com/@manivannan_data/how-to-train-ner-with-custom-training-data-using-spacy-188e0e508c6

https://sklearn-crfsuite.readthedocs.io/en/latest/tutorial.html

https://github.com/nxs5899/Named-Entity-Recognition_DeepLearning-keras/blob/master/Named_Entity_Recognition_ELMo.ipynb

https://github.com/creme-ml/creme

https://github.com/marcolagi/quantulum

http://chemdataextractor.org/

https://github.com/allenai/allennlp

https://docs.python.org/3/library/tk.html

https://en.wikipedia.org/wiki/Part-of-speech_tagging