

POLITECNICO DI TORINO

Master's Degree in ICT for Smart Societies



**Politecnico
di Torino**

Master's Degree Thesis

A combined rule-based and machine
learning approach for blackout analysis
using natural language processing

Supervisor

Prof. TAO HUANG

Candidate

GAO YU

October 2022

Summary

In the field of natural language processing, traditional information extraction methods involve lexical and syntactic analysis to extract words and parts of speech from sentences to establish semantics. This development of the new artificial intelligence branch makes it suitable for automatic tracing and analyzing blackouts in the power systems, which is very costly to the society. Therefore, the purpose of this thesis is to develop a model for extracting useful information from texts of power industry to conduct effective blackout analysis. To achieve this goal, we proposed a combined traditional rule-based and machine learning approach. A critical step was to build training data and clean data. We considered blackouts using information about when, where, and what equipment and installations failed. The dataset was generated related to blackouts by scraping websites and using OCR to get text documents. More specifically, first, blackout data was collected, and appropriate training data was created through several steps including sentence extraction, relation, and named entity extraction for tagging purposes. Then, a recognition model for a given entity type could be built based on the constructed vocabulary. From experiments, given the blackout texts, we demonstrated how to build a model to extract the desired entities, i.e. time, location, faulty facility, etc. The best results and provable evaluation metrics were obtained by continuously optimizing the model. This research helps to highlight and perceive useful information from outage incidents to specific facilities. The framework proposed by this study can surely migrate to other specific fields and can certainly improve the quality of incident analysis and provide practitioners with technical support for specific tasks.

Acknowledgements

I would like to express my great gratitude and appreciation to my supervisor Prof. Tao Huang, who kindly accepts me to participate in this work and provides me help and guidance patiently all the time.

Secondly, I want to say thank you to all my friends who trust and accompany me over the years.

The most important thing is to thank my family who give me the biggest support and understanding no matter what happens. Especially my parents, they always be my strongest backing and warmest harbor.

Thank you so much!

Gao Yu

Table of Contents

List of Algorithms	VI
List of Figures	VII
List of Tables	IX
Acronyms	X
1 Introduction	1
1.1 Background	1
1.2 Goal	2
1.3 Thesis Organization	2
2 Related Work	4
2.1 Natural Language Processing	4
2.1.1 Named Entity Recognition	6
2.1.2 Methods for NER	6
2.2 Tools for NLP	9
2.2.1 SpaCy	9
2.3 Power System Ontology	11
3 Methodology	13
3.1 System Architecture	13
3.2 Information Mining	14
3.2.1 Data Collection for Blackout List	17
3.2.2 Text Scraping	18
3.3 Pre-processing	19

3.3.1	Data Cleaning	19
3.3.2	Sentence Detection	20
3.3.3	Tokenization	21
3.3.4	Stop Words Removing	21
3.3.5	Lemmatization	22
3.3.6	Dependency Parsing	22
3.3.7	Shallow Parsing	23
3.4	Creating NE Tagged Corpus	24
3.4.1	Named Entity List	25
3.4.2	Text Classification	28
3.4.3	Corpus Generation	31
3.5	SpaCy NER Model	33
3.5.1	SpaCy Training Data Format	35
3.5.2	Data Preparation	35
3.5.3	Entity Annotations	36
3.5.4	Training Custom NER Model	37
3.6	Evaluation Metrics	39
3.7	Implementation	40
3.7.1	Transfer Learning	40
3.7.2	Refine the model	41
4	Results and Analysis	43
4.1	Result of Blackout List	43
4.2	Results of Text Classification	45
4.2.1	Semi-automatic Classification	46
4.2.2	Classification Tool	47
4.2.3	Choice of Classification Methods	48
4.3	Size of the Tagged Corpus	48
4.4	Tuning Hyper-parameters	49
4.4.1	Summary of the Best Results	56
4.5	Result of Model Test	58
4.6	Blackouts analysis	63
4.6.1	Temporal Analysis	63

4.6.2	Reason Analysis	64
4.6.3	Geographical Distribution Analysis	67
4.6.4	Scope of Impact Analysis	70
5	Conclusion and Future Work	72
5.1	Conclusion	72
5.2	Limitation and Future Work	73
	Bibliography	75

List of Algorithms

1: Snippet of code for request	17
2: Example on how to get Google Search results with Python	18
3: Example on how to download article according to URL	19
4: Snippet of code to split concatenated words	20
5: Snippet of regex to remove HTML tags from text	20
6: Sample code for tokenizing a text	21
7: Sample code that shows the stop words	21
8: Example on get content from Wikipedia webpage	26
9: Create an NLP object from pre-trained language model	29
10: Example on how to classify entities automatically	29
11: Snippet of code to create training set with EntityRuler	36
12: Snippet of code to annotate entities with spaCy	36
13: Snippet of code to set up the pipeline	38
14: Snippet of code to get entities label	38
15: Snippet of code to disable other pipes except ner	38
16: Example on how to update entity recognizer	39

List of Figures

1: An example of NER visualization results	6
2: The taxonomy of DL-based NER	8
3: SpaCy library architecture	10
4: SpaCy language processing pipeline	10
5: Framework structure	14
6: Key components of information mining	15
7: Dependency parse demo	23
8: Process of corpus building	25
9: Architecture of the text recognizer CRNN	27
10: Query results of the phrase on Ludwig Guru	32
11: The process of 1D CNN	33
12: Training process of spaCy statistical model	34
13: Dataset distribution	37
14: Partial URLs about the blackout events	44
15: Results of semi-automatic classification	46
16: Evaluation results of classification model	47
17: Word cloud created by MonkeyLearn	47
18: The frequency distribution of entities appearing in the corpus	49
19: Losses of training model obtained from tuning batch size	53
20: Losses and F1-score curves during training	54
21: Losses of training model obtained from tuning dropout rate	55
22: The performance of each label with the best hyperparameters	58
23: Annotated and predicted results for most important tags	60
24: Example results of NER part I	61
25: Example results of NER part II	62
26: The trend of selected blackouts in power systems	63

27: Statistics of the causes of the selected blackouts	64
28: Percentages of the causes for selected blackouts	66
29: Trends of four threat categories	66
30: Global selected blackouts distribution	68
31: USA selected blackouts distribution	69
32: Frequency of four threats in selected USA states	70

List of Tables

1: Partial OntoPowSys concept hierarchy	12
2: Examples of lemmatization	22
3: Category list of labels	45
4: Number of tasks completed by different classification methods	46
5: The number of sentences collected from Ludwig	49
6: Default hyper-parameters of spaCy NER	51
7: Testing results obtained from tuning batch size during training	52
8: The results of train set obtained from tuning number of epochs	53
9: Testing results obtained from tuning dropout rate during training	55
10: Best hyper-parameters of the custom NER model	56
11: The results for testing data with best hyper-parameters	57
12: Comparison between predicted and annotated NE using spaCy	59
13: List of selected wide-scale blackouts	71

Acronyms

AI Artificial Intelligence

API Application Programming Interface

BOW Bag of Words

CNN Convolutional Neural Networks

CRF Conditional Random Fields

CRNN Convolutional Recurrent Neural Network

CTC Connectionist Temporal Classification

DL Deep Learning

DL Description Logics

GRU Gated Recurrent Unit

HMM Hidden Markov Model

HTML HyperText Markup Language

HTTP HyperText Transfer Protocol

JSON JavaScript Object Notation

LSTM Long Short-term Memory

MEM Maximum Entropy Model

MSE Mean Square Error

ML Machine Learning

NER Named Entity Recognition

NLG Natural Language Generation

NLP Natural Language Processing

NLU Natural Language Understanding

NN Neural Network

OCR Optical Character Recognition

OWL2 Ontology Web Language

POS Part-of-speech

RNN Recurrent Neural Network

SaaS Software as a Service

SVM Support Vector Machines

URL Uniform Resource Locator

1D CNN one-dimensional Convolutional Neural Network

Chapter 1

Introduction

1.1 Background

The power system is characterized by a high degree of automation. It is mainly composed of electrical equipment, power transformation, distribution facilities, transmission lines, and power plants. Nowadays, people's lives and social development depend on electricity. Therefore, the power system is an important foundation to ensure and promote social stability and economic development. The safety of the power system is a crucial factor in order to ensure stability and normal operation of the system. Judging from some representative cases of the blackout that have occurred in the past, when the power system is affected by various factors that cause the system to be unsafe, it may lead to a large-scale power outage. It will cause huge economic losses, even seriously endanger people's lives and national security. For example, in 2011, the blackout of southern California, Arizona and Baja California regions [1] cost about one hundred million dollars [2]. At the end of July 2012, a major power outage occurred in India for two consecutive days. Nearly half of the national population was affected by this event. During the blackout, railway systems and subway service were shut down and the urban transportation systems were paralyzed. More severely, the civil water supply was interrupted and more than two hundred miners were trapped underground [3].

From the analysis of the power outages that have occurred in the past, there are many reasons for the safety problems in the power system, which can be roughly divided into internal factors and external factors, including grid loopholes, natural disasters, human errors, and even acts of war [4] etc. For some possible threats and future development of the power grid, it is very meaningful to statistics and analysis the power outages that have occurred in the past.

1.2 Goal

The goal of this work is to develop a model that can accurately identify and extract useful information (including when, where, and why a power failure occurred) from unstructured texts. This can be done by using text mining algorithms for data collection and, subsequently, the use of custom Named Entity Recognition (a branch of Natural Language Processing) and Optical Character Recognition (OCR) techniques to process the texts about blackouts, such as news reports, media information or professional accident reports within power industry, etc.

For this goal, there are few dataset that can be used directly, the existing data is very fragmented, and the unstructured text contains a lot of unnecessary details. And create a usable dataset needs to be collected and processed a lot before it can be put into a Machine Learning model. This makes text analysis work more difficult and time-consuming.

The main objective is to extract information based on text content. A trained model will further process the text and highlight key phrases to analysis the results and set the goals for the future development.

1.3 Thesis Organization

The thesis is organized as followings:

Chapter 1 is introduction part, which outlines the research background and goals of the thesis briefly.

Chapter 2 describes some basic concepts, and gives an overview of related works and techniques.

Chapter 3 details the methodology in this work. It covers several major aspects from information mining, data cleaning to text classification and model creation and optimization.

Chapter 4 presents all the obtained results containing the data collected for building the model, parameters selection during model training and analysis of test.

Chapter 5 provides a general summary of the work, shows some limitations and future developments to improve the overall quality of the study.

Chapter 2

Related Work

In this section, we start by explaining the basic concepts of NLP and ML. Then, we interpret the NER process for automatically extracting useful information from text. Finally, we give an overview of framework which we used and contribution of electric power ontology in this work.

2.1 Natural Language Processing

Natural Language Processing (NLP) is one of the most challenging research directions in the field of artificial intelligence (AI) that helps computers understand, interpret, and manipulate human language. NLP draws from multiple disciplines, including computer science and computational linguistics, and it acts as a bridge between machine language and human language to achieve human-computer communication. NLP is characterized by two core fields: Natural Language Understanding (NLU) and Natural Language Generation (NLG) [5] [6].

- Natural Language Understanding (NLU) makes that machines have the ability to understand language like humans. After the emergence of NLU, machines can distinguish different intentions from expressions in various natural languages, instead of relying on rigid keywords. Nowadays, almost

all applications related to text language and speech will use NLU, such as machine translation, machine customer service, smart speakers, etc.

- Natural Language Generation (NLG) is a sentence generation process that converts data in non-linguistic format into a language format that humans can understand. It can improve content production with personalized and understandable text.

For many applications it is necessary to draw on techniques belonging to both categories, in order to combine the analysis of the texts with the generation of new documents, created starting from the information that has been managed to extract.

Some commonly researched tasks of NLP are following:

- **Lexical Analysis:** Tokenization, Morphological Analysis, Part-of-speech Tagging, etc.
- **Sentence Analysis:** Chunking, Parsing, Language Modeling, Sentence Boundary Detection, etc.
- **Semantic Analysis:** Word Sense Disambiguation, Semantic Role Labeling, Word/Sentence/Paragraph Vector, etc.
- **Information Extraction:** Glossary Extraction, Event Extraction, **Named Entity Recognition**, Slot Filling, etc.
- **High-level Tasks:** Machine Translation, Question-Answering System, Text summarization, etc.

The Information Extraction, i.e. the activity related to the extraction of information from unstructured data, is the main reason for the interest shown in this work towards the NLP.

2.1.1 Named Entity Recognition

The fundamental form of text data is unstructured, and information extraction plays a very critical role in understanding the logical meaning of sentences. Named entity recognition is often used as a first step in the task of information extraction. It refers to locating, classifying and extracting "named entities" from text. We represent the words in a sentence as Named-entities, which represent real-world objects with proper names. The named entity recognition process can be divided into two steps [6]:

- Determine the boundaries of the entity
- Assignment of labels

Most Natural Language Processing libraries and tools already have predefined categories of entities as they are considered the most common. In this case we speak of Named Entity Generic and the categories that belong to it are personal names, dates, and locations. An example of NER is shown in *Figure 1*.

a severe windstorm knocked out power to 1.2 million CARDINAL customers in parts of France GPE on January 23, 2009 DATE

Figure 1: An example of NER visualization results

There are other generic entities, but which ones and how many depend on the type of software used and obviously on the language. It is also possible to develop a Domain-specific Named Entity, or a system for recognizing specific entities, chosen on the basis of the analysis context [7].

2.1.2 Methods for NER

Since NER can help answer many real-world questions, it is often mentioned and used in many domains of NLP. Methods for implementing NER can be roughly divided into two categories: traditional and deep NER approaches [8].

Traditional approaches: requires lexical analysis, syntax analysis and a rule-base for manually coded relations between words.

- **Rules-based**

Rule-based methods rely on semantic grammar rules manually formulated by linguists and domain experts to identify various types of named entities through rule matching. Although rule-based methods can achieve good results on specific corpora (exhaustive dictionaries and limited size), building these rules is not only time-consuming and difficult to cover all rules, but also has poor scalability and portability. Some famous rule-based NER methods are LaSIE-II [9], NetOwl [10], Facile [11], etc.

- **Unsupervised Learning**

Unsupervised learning methods utilize lexical resources, models and statistics obtained on large corpora to infer named entity types by using clustering [12]. The dependence on supervised information is reduced, and only a few seed rules (general rules or shallow grammar knowledge) are needed.

- **Feature-based Supervised Learning**

Feature-based supervised learning methods transform NER into sequence labeling tasks through supervised learning. Each training sample is represented by designed features according to the labeled data, which are provided by human experts from the domain. And then trains a model through Machine Learning algorithms to predict the entities labels. But the limitation is that it requires an annotated corpus for the domain of interest and artificially constructed to select effective features [13]. The main algorithms of supervised NER, which are hidden Markov model (HMM) [14], Decision Trees [15], maximum entropy model (MEM) [16], support vector machines (SVM) [17], conditional random fields (CRF) [18].

Deep Neural Network (NN) Approaches

Deep learning (DL) is a field of machine learning that is composed of multiple processing layers to learn representations of data with multiple levels of abstraction [19]. There are three advantages of applying deep learning techniques for NER: deep neural networks can learn more complex features from data by

performing non-linear transformations; Deep learning can automatically learn features from raw data; Training end-to-end deep neural NER models via gradient descent is recommended [20]. DL-based NER with the general architecture is shown on *Figure 2*, which are distributed representation, feature extractor and label decoder.

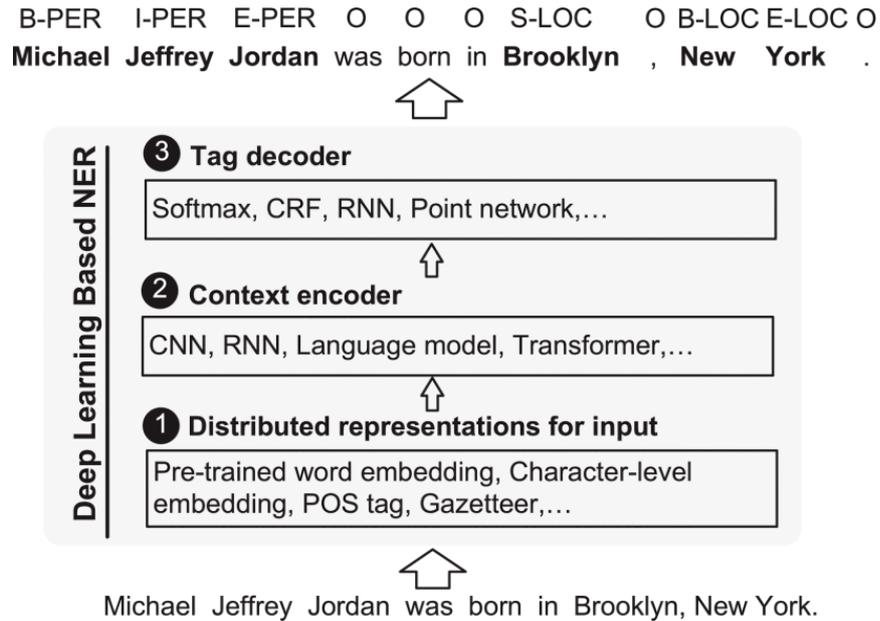


Figure 2: The taxonomy of DL-based NER [20]

Distributed representation maps words or characters into low-dimensional real-valued dense vectors, where each dimension represents a hidden feature, which can automatically learn semantic and syntactic features from text. There are three types of distributed representation: Word Embedding [21], Character Embedding [13], and hybrid representations [22]. The feature extractor learns the context feature representation by receiving the vector of the previous layer. Commonly used feature extractors are Convolutional Neural Networks (CNN) [23], Transformer [24], Recurrent Neural Network (RNN) [25], Gated Recurrent Unit (GRU) [26] and Long Short-Term Memory (LSTM) [13]. The label decoder is the last stage, which takes context features as input to generate label sequences. The common decoding methods are Softmax [21], CRF, and RNN.

2.2 Tools for NLP

There are many different libraries and NLP tools are developed for building the NER system:

- Apache OpenNLP [27]
- GATE (General Architecture for Text Engineering)
- TensorFlow
- NLTK (Natural Language Toolkit)
- SpaCy

Each of these tools has peculiarities and novelties compared to the software that are normally used most, such as OpenNLP, TensorFlow and spaCy which have pre-trained NER models that can be imported, used and can be modified or customized according to requirements. When considering the overall performance of all the models, Python's spaCy gives a higher accuracy and the best result [28].

2.2.1 SpaCy

SpaCy [29] provides an open-source Python library and neural network model for NLP tasks. The tasks include processing linguistic features such as tokenization, part-of-speech tagging, lemmatization, rule-based matching, named entity recognition, dependency parsing, sentence boundary detection, similarity detection, etc. The model is stochastic and the training process is done based on gradient and loss function. The pre-trained model can be used in transfer learning as well. The library architecture is shown on *Figure 3*. The Language class processing and converting text with multiple languages to Doc object called nlp, which has one or more pipeline components for many tasks as shown on *Figure 4*. Strings, word vectors and lexical attributes are centralized in the Vocab. The Language class, the Doc object and the Vocab are core data structures in spaCy.



Figure 3: SpaCy library architecture [30]

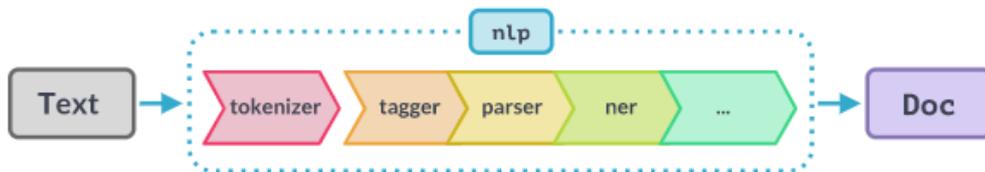


Figure 4: SpaCy language processing pipeline [30]

2.3 Power System Ontology

Ontology describes an abstract simplified view of a specific domain by making concepts and relationships explicit, and it identifies concepts relevant to representing the domain. That is, by defining terms and axioms, in some machine-readable form language can be shared by a group. The term ontology is used differently in multi-domain. In the field of power systems, the main purpose of the ontology is to be developed a task or event ontologies to achieve specific goals. For example, Pradeep et al. defined high level event ontology for power system comprising seven concepts to standardize power system events [31]. In [32], through development of an upper ontology to integrate several electricity market and system models, which explains the basic concepts of all available information to help relevant actors understand and respond to the complex and changing power system.

The OntoEIP [33] is capable of presenting knowledge in electrical engineering, logistics etc. The ontology utilizes the design philosophy of OntoCAPE [34]. Devanand et al. utilized the knowledge management approach for constructing a domain ontology for power system called OntoPowSys using a Description Logics (DL) syntax and the Ontology Web Language (OWL2) [35]. It extends the classes and properties hierarchy defined in the electrical power system modules of OntoEIP. The ontology of OntoPowSys contains the important terms and concept names of power system, as shown in *Table 1*. Among all of these, the typical properties are things such as electrical equipment, system, scalar quantity, etc. which are examples used in the thesis for information extraction.

Term name	Concept definitions
Power system	PowerSystem \sqsubseteq CompositeSystem
Bus node	BusNode \sqsubseteq ElectricalEquipment
Electricity line	ElectricityLine \sqsubseteq System
Energy meter	EnergyMeter \sqsubseteq ElectricalEquipment
Power load	PowerLoad \sqsubseteq ElectricalEquipment
Reactive power	ReactivePower \sqsubseteq ScalarQuantity
Power generator	PowerGenerator \sqsubseteq ElectricalEquipment
Rated Current	Rated Current \sqsubseteq ConstantProperty

Table 1: Partial OntoPowSys concept hierarchy [35]

Chapter 3

Methodology

This chapter describes methods for retrieving and analyzing collected unstructured data. Details for preprocessing and transforming the dataset after collection are explained. The whole process of building and optimizing the model is mainly described.

3.1 System Architecture

Since we focus on training a model to find the concept of date, location, electrical equipment, and some information related to electrical system for blackout analysis, a system was established to accomplish this goal. The overall architecture presents in *Figure 5*.

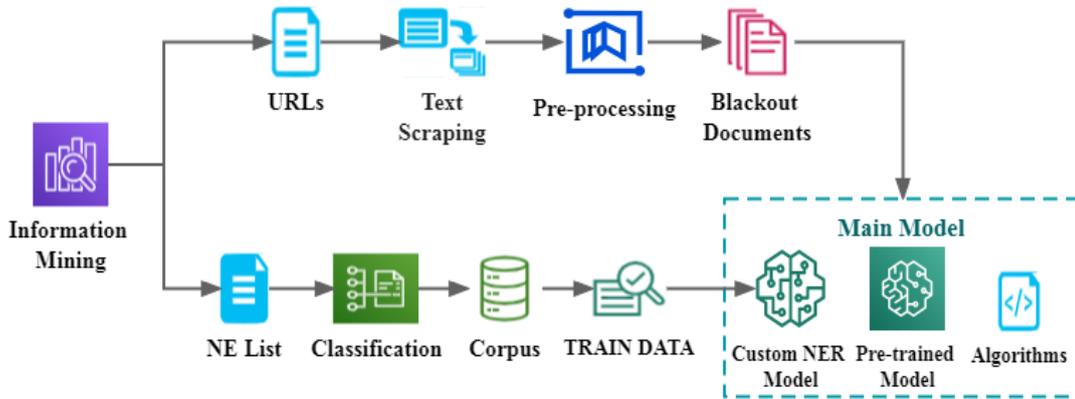


Figure 5: Framework structure

The entire work was divided into two parts that information mining and model training. We started with information mining, which is divided into two branches. One is to mine the entities we are interested in and classify them so that each entity has an appropriate label, in order to build a corpus for train the NER model. Another branch is mining texts about blackout event. Crawling text information according to URLs related to blackout. Finally, the preprocessed blackout files were transferred to a main model consisting of a custom NER model, a pre-trained model and optimization algorithms for completing blackout analysis.

3.2 Information Mining

Information Mining is also called “Web Information Mining”, it can be broadly defined as the discovery and analysis of structured or unstructured data that can be quantitative, textual, and pictorial in nature from the World Wide Web. We set forth a summary representation of the elements of Information Mining [36] as shown in *Figure 6*. According to different mining objects, network information mining can be divided into network content mining, network structure mining and network usage mining.

When performing Information Mining, it consists of following steps:

- **Resource discovery:** to retrieve the required network documents
- **Information selection and preprocessing:** to automatically select and pre-process specific information from the retrieved network resources
- **Generalization:** find common patterns within a single and multiple web sites
- **Analyze:** identify or interpret the patterns mined

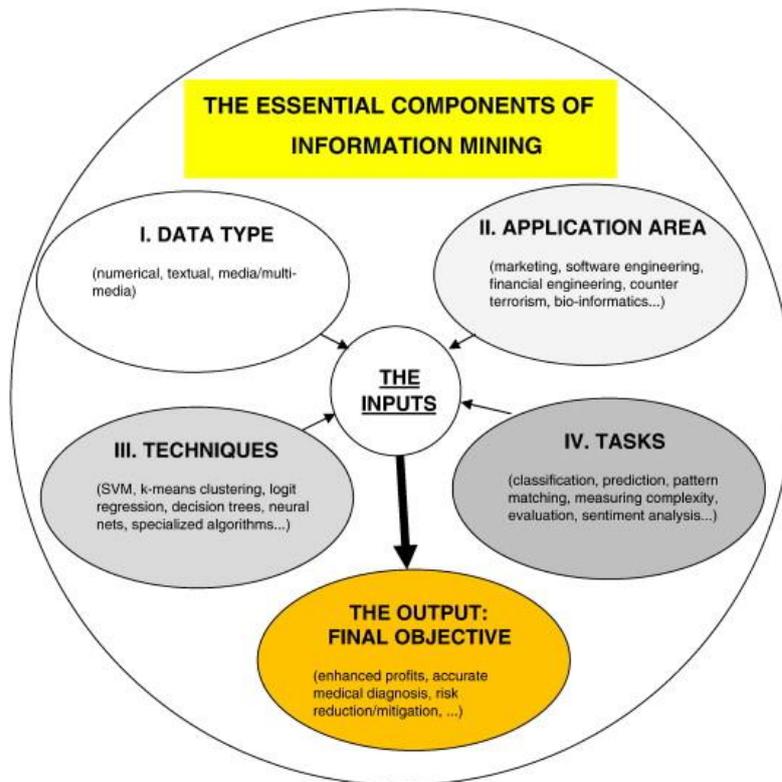


Figure 6: Key components of information mining [36]

In this section, we focus on network content mining to collect textual data about power outage, i.e. blackout on the Internet with machine learning as well as

statistics algorithms. We write python scripts to get Google Search results by using requests and BeautifulSoup, which are python library.

Google Search

As the most-visited website in the world, Google Search is a search engine provided by Google Inc. The main purpose of it is to search for text in publicly accessible documents provided by web servers. Additionally, it offers many different options for customizing searches, using symbols to include, exclude, specify, or require certain search behaviors. The algorithms of Google Search organize the hundreds of billions of pages in search index to provide the most relevant and useful results based on a user's query, all in an instant. The order of search results is returned by PageRank, which is a priority rank system.

Python

Python is an interpreted high-level general-purpose programming language. Python code is very concise and highly readable. Its language structure and object-oriented method help developers express ideas with less code, which greatly improves work efficiency. It is dynamically-typed and garbage-collected. When a part of the memory space occupied by a program is no longer accessed by the program, the program will return this part of the memory space to the operating system with the help of the Garbage Collection algorithm. Python code does not need to be compiled, can be run directly by the python interpreter, and can modify the properties of classes in real time, which is a great advantage compared to other programming languages that require a compiler.

Requests

Requests is a simple HTTP library for python. It allows user send HTTP requests without add query strings to URLs or to form-encode POST data manually. Keep-alive and HTTP connection pooling are 100% automatic thanks to another python HTTP library called urllib3. The response is serialized JavaScript Object Notation (JSON) format, which is used to represent general data based on JavaScript object syntax. So user can easily access values in the object by key.

3.2.1 Data Collection for Blackout List

The first step for blackout analysis is data collection. Filtering and preprocessing large amount of data to avoid noise and outliers are the main issues that we have to face at this stage.

There are many methods of data collection, which can be roughly divided into manual, automatic and semi-automatic. Among them, regarding manual data collection, it is a relatively primitive method that requires a lot of time and staff. In general, the data obtained are accurate and reliable because noise and duplicates are avoided. Therefore, it is often used when the data range is large and the number is small. For automatic data collection, it is the fastest method, especially when we collect huge amounts of network information. At the same time, this method has high requirements on hardware and software facilities, and needs to set constraints and filtering conditions in order to obtain expected results. Today, the most popular and legal methods are downloading of open dataset, reading API, and web crawlers with some third parties. About semi-automation method, it is a combination of manual and automatic method. Analysts need to choose the appropriate method for specific data at different stages of data collection.

In this work, we choose semi-automation method to prepare the blackout data. There are two datasets here. The first dataset is from the crawling of blackout-related websites. The blackout-related URLs were crawled from Google search results using a python script. Firstly, install the requirements and import it into python script. To perform a search, expects the query to be in the parameters of the URL. Additionally, all spaces must be replaced with '+'. To build the URL, we properly format the query and put it into the 'q' parameter, as shown in *Algorithm 1*.

```
1 import urllib
2 import requests
3 from bs4 import BeautifulSoup
4 query = input("Enter something to search:")
5 query = query.replace(' ', '+')
6 URL = f"https://google.com/search?q={query}"
```

Algorithm 1: Snippet of code for request

The different search results will be return for mobile and desktop. So depending on the use case, we need to specify appropriate user-agent in the headers. The request was successfully when the status code returns a '200'. Then we put it into `BeautifulSoup`, which is a python library that allows parse content and extracting data from HTML files as shown in *Algorithm 2*.

```
1 results = []
2 for g in soup.find_all('div', class_ = 'r'):
3     anchors = g.find_all('a')
4     if anchors:
5         link = anchors[0]['href']
6         title = g.find('h3').text
7         item = {
8             "title": title,
9             "link": link
10        }
11 results.append(item)
12 print(results)
```

Algorithm 2: Example on how to get Google Search results with Python

3.2.2 Text Scraping

After we successfully get the URL list about the blackout event, what we need to do is to parse HTML source code and then save it. To In order to get the specific HTML source code according to the URL, python provides a variety of methods, mainly including library of `urllib` and `requests`. After comparing the results of accessing HTML between the two libraries, we decided to use `requests`, since it can get more complete HTML source code.

After processing the HTML using `requests`, the complex code needs to be parsed and only the content we need is extracted. Python also provides many methods for parsing HTML, regular matching, `BeautifulSoup`, `Xpath`, etc. Here we use `BeautifulSoup` as shown in *Algorithm 3*.

```

1 res = requests.get(url)
2 html = res.text
3 soup = BeautifulSoup(html, 'html5lib')
4 for script in soup(["script", "style", "aside"]):
5     script.extract()
6 text = " ".join(re.split(r'[\n\t]+', soup.get_text()))

```

Algorithm 3: Example on how to download article according to URL

3.3 Pre-processing

In textual data science tasks, any raw text needs to undergo detailed preprocessing, and then digest it using algorithms. After the data collection stage, the preprocessing stage will be performed in order to be more organized, structured and usable by computers for avoiding the analysis that might lead to erroneous results.

3.3.1 Data Cleaning

After the collection phase, data cleaning is performed to remove all unnecessary data that may be adding noise to the data set. The text files are obtained from websites according to the URLs list, and some of the content may have the following conditions:

- Concatenated words
- HTML tags
- Non-text content
- Non-English characters and punctuations
- Abbreviation

For the first case that might arise in the text we mentioned above, where words are not separated by spaces. As shown in *Algorithm 4*, it can be solved by a python library `Wordninja`, which is probabilistic splitting of concatenated words

using NLP based on English Wikipedia uni-gram frequencies. Separating each word with a space helps the word segmentation of the text, and also lays the foundation for accurate NER.

```
1 import wordninja
2 split = wordninja.split(text)
```

Algorithm 4: Snippet of code to split concatenated words

Python's regular expression (or regex) is used to filter meaningless content, such as HTML tags contained in text, non-text content, non-English characters, etc. For abbreviation, like don't, I'm, 's, regex also be used for replace acronym with full format. A regular expression (or regex) is a sequence of symbols that identifies a group of strings. It is used as search pattern to scan texts and find and manipulate strings. As shown in *Algorithm 5*, in python the library `re` allows the manipulation of regular expressions.

```
1 import re
2 cleanr = re.compile('<.*?>')
3 text = re.sub(cleanr, ' ', text)
```

Algorithm 5: Snippet of regex to remove HTML tags from text

3.3.2 Sentence Detection

Sentence detection is a process of locating the beginning and end of sentences in a given text. It can divide the processed text into linguistically meaningful units that will help process the text to perform tasks such as part-of-speech (POS) tagging and entity extraction. In spaCy, the `sents` property is used to extract sentences. User can customize the sentence detection to detect sentences on custom delimiters. For our case, it is correctly able to identify sentences in English language, using a full stop (.) as the sentence delimiter.

3.3.3 Tokenization

Tokenization is the process of forming tokens (sequence of characters) from input stream and it can be defined as the next step after sentence detection in NLP. It allows identifying the basic units in input text, and these basic units called tokens. Tokens can be identifiers, operators, keywords, symbols and constants. For this purpose, the *token* property from the spaCy python library is used which returns a list of the words (punctuation and numbers included) of the text by iterating on the *Doc* object. Meanwhile, spaCy provides various attributes for the Token class as shown in *Algorithm 6*.

```
1 import spacy
2 for token in doc:
3     print(token, token.idx, token.text_with_ws, token.is_alpha,
           token.is_punct, token.is_space, token.shape_, token.is_stop)
```

Algorithm 6: Sample code for tokenizing a text

3.3.4 Stop Words Removing

Stop words are common words that have no benefit for analysis of NLP. In fact, search engines automatically ignore these words when inserting them. For this purpose, the *stop_words* from the spaCy python library is used as shown in *Algorithm 7*. In this work, we need to fine-tune the provided stop words list according to the characteristics of the power domain. In particular, remove from the list some prepositions that may form phrases with verbs, such as ‘in’, ‘out’, ‘up’, ‘down’, etc. They are indicative of NER in *Section 3.5 SpaCy NER Model*. To do this, we customized a python script to remove from the default stop words those that are useful for our work.

```
1 import spacy
2 print(spacy.lang.en.stop_words.STOP_WORDS)
```

Algorithm 7: Sample code that shows the stop words

3.3.5 Lemmatization

Lemmatization is a process which converts complex forms of words into its basic grammatical form, in which the basic form of the word is extracted through additional dictionary lookup.

For example, organizes, organized and organizing are all forms of organize. Here, organize is the lemma. The inflection of a word allows expressing different grammatical categories like tense (organized vs. organize), number (trains vs. train), and so on. Lemmatization is a necessary step since it helps reduce the inflected forms of a word so that they can be analyzed as a single item. Meanwhile, It can also help normalize the text. spaCy has the attribute *lemma_* on the *Token* class. The *Table 2* shows different examples of how the lemmatization approach changes the words.

Original	Lemma
Were	Be
Lines	Line
Organized	Organize
Generation	Generate

Table 2: Examples of lemmatization

3.3.6 Dependency Parsing

Dependency Parsing is a process to analyze the grammatical structure in a sentence and find out related words as well as the type of the relationship between headwords and their dependents. The dependencies can be mapped in a directed graph representation:

- Words are the nodes.
- The grammatical relationships are the edges.

Dependency parsing helps to understand the roles that words play in the text and the relationships between different words. When the relationship between

named entities in a sentence needs to be discovered, a pre-trained model can be used to identify the relationship between words or word chunks.

Figure 7 shows that the subject of a sentence is proper noun `Bus nodes` and it has a subordinate relationship with `electrical network`. It represents an example of relation in our raw corpus for the property ‘has’ which is used in power system ontology concept.

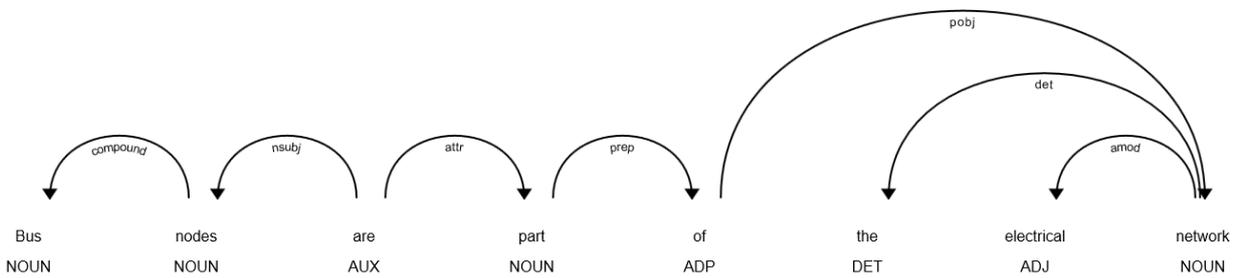


Figure 7: Dependency parse demo

3.3.7 Shallow Parsing

Shallow parsing, or chunking, is the process of extracting phrases from unstructured text. Chunking groups adjacent tokens into phrases on the basis of their POS tags. There are some standard well-known chunks such as noun phrases, verb phrases, and prepositional phrases.

Noun Phrase Detection

A noun phrase is a phrase that has a noun as its head. It could also include other kinds of words, such as adjectives, ordinals, determiners. Noun phrases are useful for explaining the context of the sentence. They help you infer what is being talked about in the sentence. spaCy has the property `noun_chunks` on `Doc` object, it can be used to extract noun phrases.

By looking at noun phrases, we can get information about the text. For example, ‘a blackout event’ indicates that the text mentions a blackout, while the date 21 July lets us know that blackout occurred for 21 July. We can figure out

whether the blackout is in the past or the future. If there is a location is detected that tell us the position of the blackout.

Verb Phrase Detection

A syntactic unit consisting of at least one verb is called a verb phrase, and it helps to understand the action involved in the noun. This verb can be followed by other blocks, such as noun phrases. SpaCy has no built-in functionality to extract verb phrases, so we need a library called `textacy` which used to extract verb phrases based on grammar rules.

In the text of electricity field, verb phrase is an important signal. For example, cut off, break down, etc. Empirically, verb phrases such as these often appear in sentences involving faulty equipment, so this step is important for analyzing blackout events.

3.4 Creating NE Tagged Corpus

We focus on creating NE categories about power industry because others such as date and location are relatively easier to recognize and the power industry categories actually suffer from the shortage of an NE tagged corpus.

Information in various languages is already co-hosted in written form on the web, and the amount has recently increased almost infinitely. The network can be viewed as an infinite linguistic resource that contains various NE instances with different contexts. The key idea is to use a precompiled list of NEs to automatically label such NE instances with appropriate class labels. However, due to word ambiguity and boundary ambiguity of NE instances, there should be some general and language-specific considerations in this tokenization process. To build a corpus containing specific entities, its generation process consists of three steps. The process first uses data collection techniques to obtain useful NE entries from documents and secondly assigns them to appropriate label. Next, sentences are generated and searched by inputting entities with categories to linguistic

search engine. Extracting each sentence returned by the search engine to construct corpus. *Figure 8* explains the whole process of generating the NE corpus.

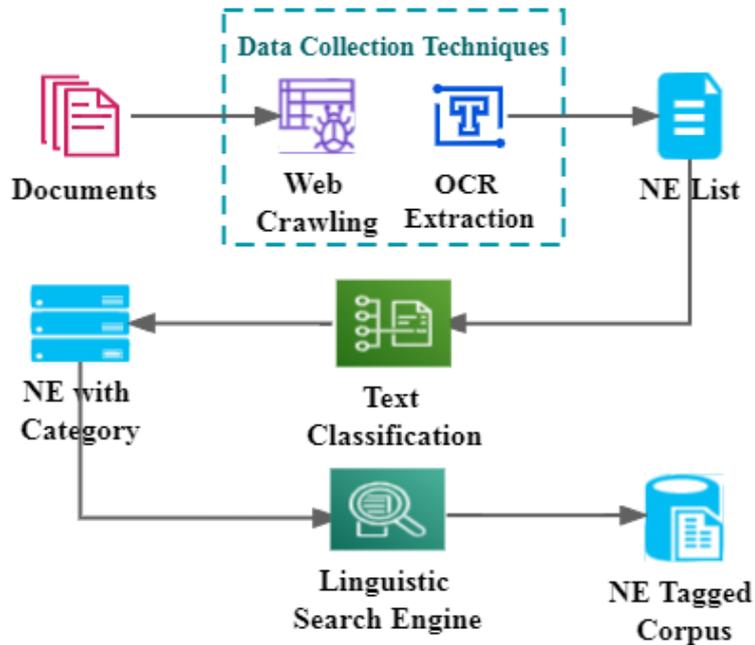


Figure 8: Process of corpus building

3.4.1 Named Entity List

Randomly collecting files from the web does not serve our purpose. Because not all web documents contain NE instances relevant to our work, and we currently do not have a list of all these entities. So building a NE list is an important step to create an NE corpus. The building of the NE list, which is sets of special keywords in domain of electrical, is required and then saved in text files as the second dataset of our work. These keywords can be multiple words or chunks, which are useful for tokenizing word chunks for sentence generation in *Section 3.4.3 Corpus Generation*.

Web crawling

Part of the data crawled from Wikipedia of the Electrical and Electronic Engineering Glossary. We called a python library of Wikipedia API, which supports extracting texts, sections, links, categories, translations, etc. from Wikipedia. The code snippet for our work is shown in *Algorithm 8*.

```
1 import wikipediaapi
2 wiki_wiki = wikipediaapi.Wikipedia(
3     language = "en",
4     extract_format = wikipediaapi.ExtractFormat.WIKI
5 )
6 p_wiki= wiki.wiki.page("Glossary of electrical and electronics
7 engineering")
8 print(p_wiki.text)
```

Algorithm 8: Example on get content from Wikipedia webpage

OCR extraction

OCR is a technology that can quickly and accurately recognize text information from images, has a long research history and wide range of application scenarios. In this work, due to the imperfect records of electrical-related vocabulary, for another part of data, they are power-related words that exist in some electric power industry .pdf format files. In order to extract the vocabulary accurately and completely, we adopted the screenshot ORC function in Quicker¹ software.

Although the use of traditional OCR technology to model specific scenes has achieved good recognition results, the model will fail once it leaves the preset scene, so scene text recognition based on deep learning is more popular. There are two methods based on deep learning. One is divided into two stages of text detection and text recognition. Another is to complete text detection and recognition at the same time through an end-to-end model. Quicker OCR use an end-to-end trainable neural network was proposed by Shi et al, called Convolutional Recurrent Neural Network (CRNN), which integrates feature

¹ <https://getquicker.net/>

extraction and sequence modeling, the network architecture as shown in *Figure 9* [37]. The architecture consists of three parts:

- Convolutional layers: constructed by taking the convolutional and max-pooling layers from a standard CNN model to extract a feature sequence from the input image.
- Recurrent layers: use deep bidirectional LSTM to predict a label distribution for each frame.
- Transcription layer: translates the per-frame predictions made by RNN into the final label sequence.

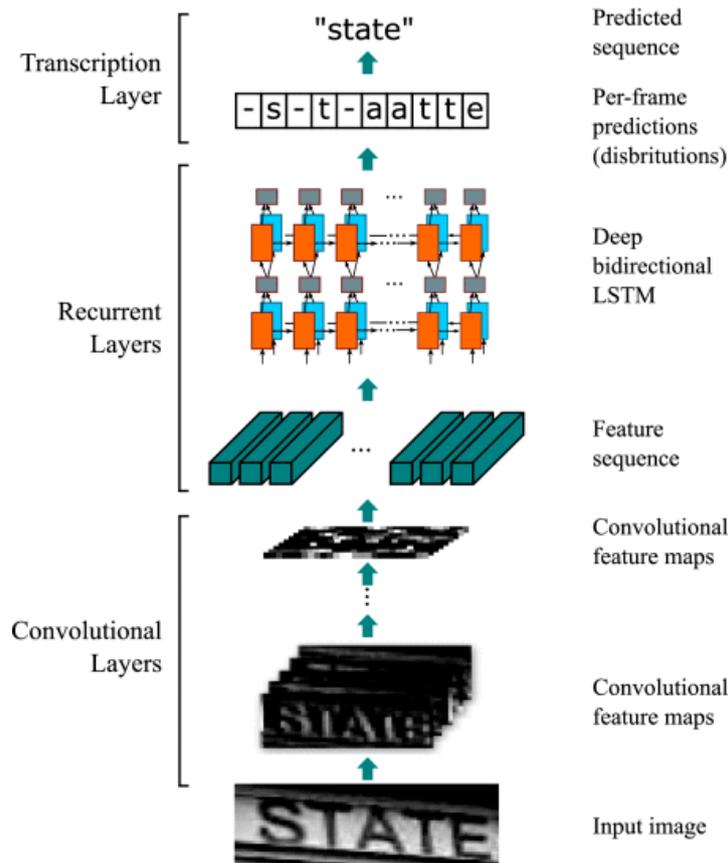


Figure 9: Architecture of the text recognizer CRNN [37]

For transcription layer, it adopts the conditional probability defined in the Connectionist Temporal Classification (CTC) [38] loss to avoid the inconsistency between prediction and label. The objective is to minimize the negative log-likelihood of conditional probability of ground truth shown in

Formula 1, where I_i is the training image, I_i is the ground truth label sequence and y_i is the sequence produced by the recurrent and convolutional layers from I_i [37].

$$O = - \sum_{I_i, I_i \in X} \log p(I_i | y_i)$$

Formula 1: Objective function of label sequence

3.4.2 Text Classification

Text classification is a common technique used in on basic NLP tasks as well. The important task of creating the corpus is to generate tags for each seed in the NE list. To perform this task, we first use semi-manually classify and then compared it with machine learning algorithms. The semi-manually classify relies on ontology of electrical domain and spaCy library, while the machine learning algorithm developed by an AI platform MonkeyLearn², which is more efficiently than previous method.

Semi-manually

This stage is called semi-automatic classification because it consists of two parts, one is to use NLP model to process unstructured text, in order to achieve as many entity classification tasks as possible. The other is to organize and standardize some entities that cannot be automatically classified by algorithms through

² <https://monkeylearn.com/>

manual review of data or through empirical cognition, and finally group them into corresponding categories.

At first, we roughly determine 19 very fine label categories according to the attributes of the ontology to automatically label as many entities as possible. We defined a specific function for each category label and use the small English pre-trained model that is `en_core_web_sm` provided by spaCy as an NLP tool for semi-automatic classification, as shown in *Algorithm 9*.

```
1 import spacy
2 nlp = spacy.load("en_core_web_sm")
```

Algorithm 9: Create an NLP object from pre-trained language model

Each of these specific functions contains restrictions on whether an entity can be selected as that category, and these restrictions are actually specified according to the inherent properties and word formation rules of the entity. For example, in *Algorithm 10*, filter entities that satisfy the ‘MATH’ label requirements by determining whether all tokens are within the `theory` which is constraints we defined.

```
1 def is_math(doc):
2     for token in doc:
3         if token.lower_ in theory:
4             expressions.remove(doc.text)
5             MATH.append(doc.text)
6         return True
7     return False
```

Algorithm 10: Example on how to classify entities automatically

In our case, not all entities can be automatically classified by the algorithm to match the corresponding label, because the pre-set constraints for each class only apply to a subset of entities for which word formation rules can be found. Although it will take a lot of time, in order to ensure the reliability of the classification task. For those unlabeled entities, we have to classify and summarize them under the corresponding labels manually. Since automatic

classification is not hundred percent correct, the classified entities also need to be manually checked.

Classifier tool

Since manual classification in the semi-automatic stage consumed a lot of time, in order to alleviate the time cost, we decided to use an online codeless classifier. The success of methods for manual classification relies on large and high-quality labeled examples, which are often labor-intensive and time-consuming. We utilized MonkeyLearn analysis platform as the foundation for classifier mechanisms, which is software as a service (SaaS) with machine learning.

MonkeyLearn uses several algorithms that combine to create the entire system. According to MonkeyLearn itself, SVM is a supervised machine learning model that uses linear classifier. After providing an SVM model with sets of labeled training data for each category, they were able to classify new data. Classification methods mostly use bag of words (BoW) and the kernel function of SVM [39].

Moreover, it also utilizes other algorithms to support its platform by implementing probabilistic classifier to predict certain words that have been annotated previously. The classifier based on Naïve Bayes theorem, shown in *Formula 3*, which begins with the conditional probability described in *Formula 2*. Naïve Bayes is essential for inferential statistics and many advanced machine learning models. Bayesian inference is a logical method to updating the probability of a hypothesis based on new evidence. It allows users to answer questions of frequency statistical methods, which have not been developed [40]. Essentially, Bayesian theorem supposes that $P(A)$ is the "A Priori" or marginal probability: they are calculated independently of each other. Calculate the conditional probability $P(B|A)$ by using Naive hypothesis: all attributes are statistical independent, but since it is not always true, it may affect the model quality generating lower accuracy results.

$$P(A | B) = \frac{P(A \cap B)}{P(B)}$$

Formula 2: Conditional Probability

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

Formula 3: Bayes Theorem

The system can train a text classifier to associate input text with its assigned corresponding category by using mathematical models and these machine learning algorithms, in the process the classifier generates "rules" to classify new input unlabeled text data. Furthermore, classification process can be done by applying API which provided by MonkeyLearn to predict the aspect of extra testing data.

3.4.3 Corpus Generation

Based on the existing list of NEs and the categories, we need to extract sentences from web documents which necessarily contain at least one NE instance for creating training raw corpus for named entity training. Using whole paragraphs in document is not sensible, since each paragraph is too long and there are lots of irrelevant sentences or words which can create a lot of noise in training data. However, manually extracting sentences from description fields is a time consuming task. An et al. [41] submitted the NE entities as queries to a search engine to obtain URLs and utilized web robot visits the web sites in the URL list to fetch the corresponding web documents. The main task of this step is that we have to specifically find the sentences that contain these named entities. For our case, we decided to generate sentences for each entity belongs to the already constructed NE list with the help of an online corpus, namely Ludwig Guru³.

³ <https://ludwig.guru/>

In this work, we employed Ludwig Guru, a popular linguistic search engine, as a tool to find sentences associated with the entities. As a unique search engine developed by a large number of linguists and computer experts, Ludwig provides users with many functions such as contextualized translation, advanced search options etc. We focus on one of the powerful functions, which is to find context provided by reliable and inspiring English sources for the words we entered. As shown in *Figure 10*, the source of the phrase is generated on the right and it allows user to explore more the sentence source. The searched phrase is underlined in blue so as to make the user easy to spot the word/phrase. This not only solves the condition that the generated sentence contains at least one entity, but also ensures the professionalism and traceability of the sentences in our custom corpus. However, since some entities involve professional vocabulary and even proper nouns in the power industry, they only appear as words and do not exist in a complete sentence, so not all entities in the list are successfully completed the task of sentences generation in Ludwig Guru.

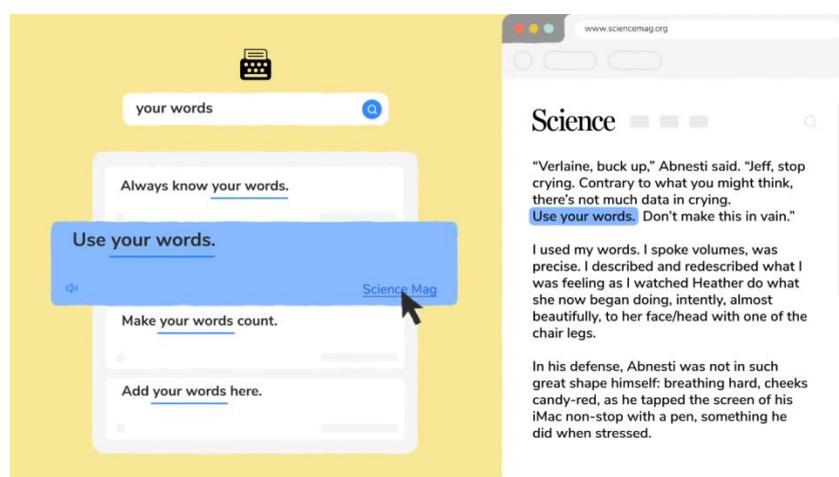


Figure 10: Query results of the phrase on Ludwig Guru [42]

We edited a python script so that computer can automatically send the seeds in batch to the search engine to accomplish query with pre-compiled NE list. For sentences that can be queried that contain at least one seed, we scraped the results as our custom corpus for *Section 3.5 SpaCy NER Model*.

3.5 SpaCy NER Model

SpaCy is primarily an NLP library for machine learning, but its main advantage is the ability to use rule-based NER methods as well. It is essential that combination of rule-based and machine learning-based approaches to develop a robust NER system for a specialized domain. The spaCy NER environment uses a word embedding strategy using a sub-word features and Bloom embedding and one-dimensional Convolutional Neural Network (1D CNN).

- **Bloom Embedding:** It is similar to word embedding and more space optimized representation. It gives each word a unique representation for each distinct context it is in.
- **1D CNN:** It is applied over the input text to classify a sentence/word into a set of predetermined categories. The realization process is shown in *Figure 11*.

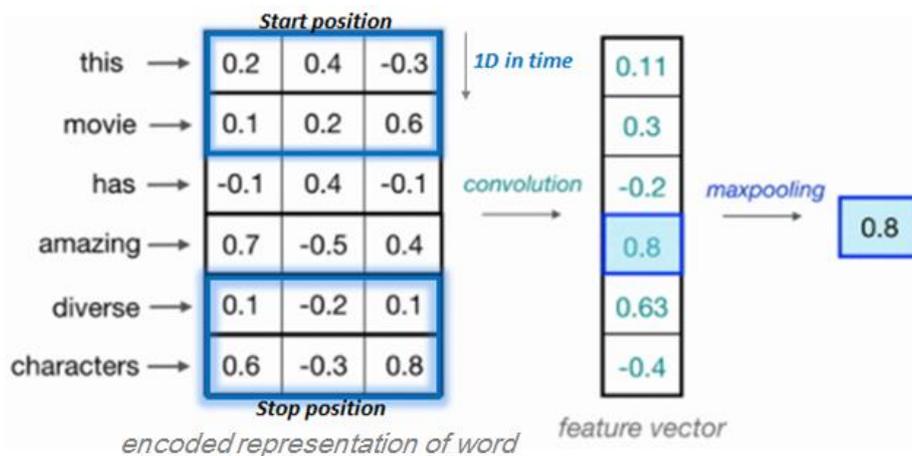


Figure 11: The process of 1D CNN

SpaCy not only makes working with machine learning models easy, but it greatly reduces the complexity of training custom machine learning models. To train a model, a standard training data is cornerstone, because all subsequent work is based on it. For spaCy statistical model, the training data is required to be

annotated which contain both examples of text and their labels. The model then displays the unlabeled text and makes prediction. The whole process as shown in *Figure 12*, the model can give feedback on the predictions made by itself in the form of the error gradient of the loss function, since we have identified the correct labels in the training data. Based on the error gradient, the model compares the predicted label with the actual label and adjusts its weights to make correct action have a higher score than before.

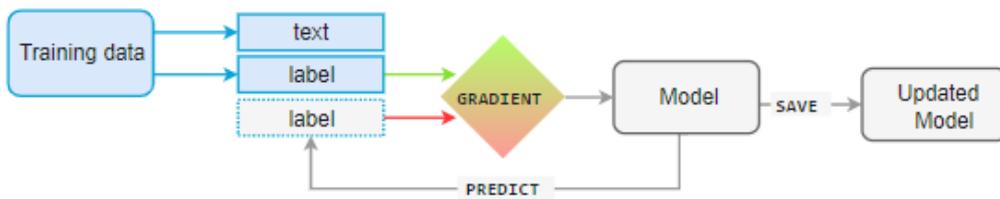


Figure 12: Training process of spaCy statistical model [43]

In spaCy, Mean squared error (MSE) is used as the loss function of the model. It is the most commonly used loss function for regression to measure how well a model performs on unseen dataset. The loss is the mean overseen data of the squared differences between true and predicted values, or writing it as *Formula 4*, where \hat{y} is the predicted value.

$$L(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^N (y - \hat{y}_i)^2$$

Formula 4: Calculating mean squared error loss

During the model training, we tried to minimize an expected value of loss function on our training data. A smaller loss value means better performance of the model and vice versa.

3.5.1 SpaCy Training Data Format

According to requirements of spaCy for the training set, the input data should be in the following format:

```
TRAIN_DATA = [(TEXT AS A STRING, {"entities": [(START, END, LABEL)]})]
```

Manually converting training data to the format spaCy requires is very difficult. The researcher must count the number of characters to specify where the entity starts and ends. Another problem appears that using Python's built-in string functions to get start and end characters. Because using string functions to calculate character positions differs from how spaCy reads the character offsets during training procedure. This means that spaCy remove labels that are inconsistent with the start and end of the label in training. In order to reduce the time and effort cost of manual annotation, some tools have been developed, such as Prodigy⁴, doccano [44], etc. Especially Prodigy, a paid scriptable annotation tool developed from the makers of spaCy, namely Explosion AI. It is efficient that data scientists can do the annotation themselves, enabling a new level of rapid iteration. Prodigy is a good choice if the task is on a budget as it fits seamlessly into the spaCy workflow. For our case, we automatically generated a basic training set using rule-based methods.

3.5.2 Data Preparation

The data structure for training machine learning models is a list, where each index contains a text (a sentence, paragraph or whole text). In general, the length of text depends on the target to be achieved through ML-NER, while the size of the text will affect the training process. It is precisely because of these potential problems, we try to avoid using long text when creating the corpus in *Section 3.4.3 Corpus Generation*, but choosing each index contains a sentence. We made a machine learning training set by EntityRuler.

⁴ <https://prodi.gy/>

EntityRuler is a spaCy factory that allows creating a set of schemas with corresponding labels. We pay attention to it as a component of the model pipeline in this section. For example, in *Algorithm 11*, in order to incorporate EntityRuler into a model, it must be created as a new fitting and added to the model. Then, user can save the new model with EntityRuler to disk for recalling later.

```

1 ruler = EntityRuler(nlp, overwrite_ents = True) #create EntityRuler
2 pattern = [{
3     "label": TAG OF SEED,
4     "pattern": SEED OF NE LIST
5     }] #build entity and mode list
6 ruler.add_patterns(patterns) #add mode to EntityRuler
7 nlp.add_pipe(ruler) #join EntityRuler to the pipeline
8 nlp.to_disk("RULE NAME") #save rule

```

Algorithm 11: Snippet of code to create training set with EntityRuler

3.5.3 Entity Annotations

Besides the sentences in corpus, another element required for training data is a list of entities of text, including their start and end positions and labels in text. The NE tagged list we have done in *Section 3.4.1 Named Entity List*. So the main task we need to solve in this section is to complete entity annotation by using all known entities (seeds) to label all collected training sentences. In general, two nested loops (one for sentences and one for seeds) can be used to implement full labeling task with a complexity $O(|\text{Sentences}| * |\text{Seeds}|)$ [45]. However, it would take a long time to label a large amount of sentences for entity seeds, even with the help of multi-thread implementation. Fortunately, a built-in function in spaCy has the ability to automatically get character offsets according to the rules specified by calling the new model with EntityRuler, as shown in *Algorithm 12*.

```

1 nlp = spacy.load("RULE NAME") #calling the new EntityRuler model
2 doc = nlp(sentence) #traverse the corpus
3 for ent in doc.ents: #extract entities
4     entities.append([ent.start_char, ent.end_char, ent.label_])

```

Algorithm 12: Snippet of code to annotate entities with spaCy

Since spaCy takes training data in JSON format, it is necessary to convert the training data into a spaCy-readable form after completing entity annotation. During training, these annotations will allow CNN (architectural structures behind the spaCy machine learning training) to learn from the data and correctly identify the practiced entities being trained.

3.5.4 Training Custom NER Model

We divided our data set into two subsets. One subset is training data which will be used to fit our model. The other subset is test data which will be used to evaluate our model. In our dataset, 80% of the data was used for the training phase and 20% of the data was used for the testing phase. The dataset distribution of each entity label is shown in *Figure 13*.

- Training set includes 80% of the data set and has 8200 samples.
- Testing set includes 20% of the data set and has 2003 samples.

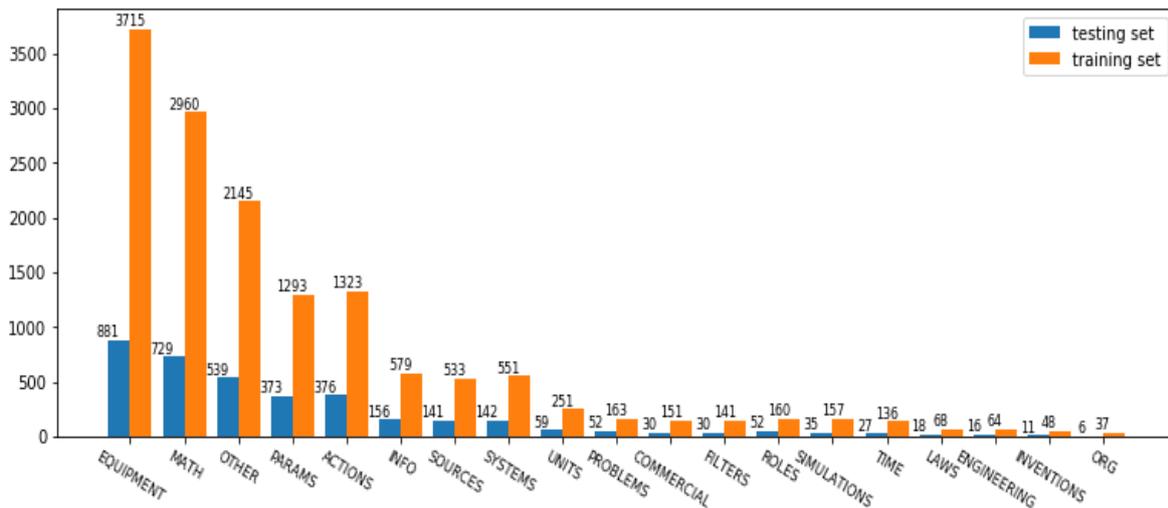


Figure 13: Dataset distribution

To train a NER model defined by ourselves, we create an empty class with English and add the entity recognizer which is the built-in pipeline components to the pipeline if there is no existing. This procedure is shown on *Algorithm 13*.

```
1 nlp = spacy.blank("en")
2 if "ner" not in nlp.pipe_names:
3     ner = nlp.create_pipe("ner")
4     nlp.add_pipe(ner, last = True)
```

Algorithm 13: Snippet of code to set up the pipeline

In fact, every decision made by spaCy is a process of prediction. For all predictions are based on the examples learned by the model during training. Therefore, in order to make the spaCy model aware of all custom tags, we can add the new entity from our annotated data to the entity recognizer using *add_label* method, as shown in *Algorithm 14*.

```
1 for _, annotations in TRAIN_DATA:
2     for ent in annotations.get("entities"):
3         ner.add_label(ent[2])
```

Algorithm 14: Snippet of code to get entities label

It is essential to acquire other pipeline components than the entity recognizer and disable them to remove any impact on the training process, as shown on *Algorithm 15*, as we are only focusing on entity extraction.

```
1 other_pipes = [pipe for pipe in nlp.pipe_names if pipe != "ner"]
2 nlp.disable_pipes(*other_pipes)
```

Algorithm 15: Snippet of code to disable other pipes except ner

We trained our model for a number of iterations so that the model can learn from it effectively. At each iteration, the training data is shuffled to ensure the model doesn't make any generalizations based on the order of examples. Pass raw texts and dictionaries of annotations to update the model for each iteration. An example of use is shown in *Algorithm 16*.

```

1 nlp.update(
2     [text], #sentences in TRAIN_DATA
3     [annotations], #Labels in TRAIN_DATA
4     drop = dropout_rate,
5     sgd = nlp.begin_training(),
6     losses = losses
7 )

```

Algorithm 16: Example on how to update entity recognizer

3.6 Evaluation Metrics

For evaluation of hyper-parameters and models, precision, recall and F1-score have been used in NLP system. In particular, the binary classification problem is referred to in the context of NER and entity extraction from text [46]. In fact, each token in text needs to be evaluated whether it can be recognized. Therefore, we focus on positive examples, thus choosing precision and recall, which are mainly evaluate quantitative aspects of recognition, but also qualitative aspects [46] [47].

How many positive predictions are correct is indicated by precision. The precision value is calculated as *Formula 5*, where we find the ratio between true positive and total positive predictions gives the precision result.

$$precision = \frac{TP}{TP + FP}$$

Formula 5: Precision calculation formula

The recall also known as sensitivity is the number of positive predictions divided by the sum of all positives in the system should predict. The recall formula is in the following (*Formula 6*).

$$recall = \frac{TP}{TP + FN}$$

Formula 6: Recall calculation formula

The recall or precision is given the highest priority depends on the reference target, since in most cases they are opposite, i.e. high sensitivity values correspond to low precision values and vice versa. In order to balance the two measures and obtain a value that provides an overall indication of the performance of the system, the F1-score or F-measure is also used, calculated as the harmonic average between precision and recall, as shown in *Formula 7* [46].

$$F1 - score = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Formula 7: F1-score calculation formula

3.7 Implementation

3.7.1 Transfer Learning

Transfer learning is one of the most influential recent breakthroughs in NLP. Before transfer learning became popular, almost all NLP models were trained entirely from scratch. While the standard approach to natural language processing has changed dramatically these days, the safest approach is to download some mature pre-trained models and fine-tune them for specific NLP tasks. Because these transfer learning models have learned a lot of unlabeled text, which have learned a lot about language: understand the meaning of words and sentences, coreference, grammar, etc.

However, most transfer learning models are huge, and they have so many parameters that they are rather slow and resource intensive. It is clear that more traditional, smaller models with relatively few parameters may not be able to handle all the NLP tasks thrown at them. But for simple text recognition or sequence labeling tasks, small models are sufficient to handle. In order to meet the NLP in different situations, spaCy offers three pre-trained models of different magnitudes for the English language, which are large, medium and small sizes. In our case, we choose a small-sized English model to perform NLP tasks on some unspecified entities, such as dates, locations, numbers, etc.

It is necessary to combine the custom NER model with existing off-the-shelf model from spaCy to implement the NLP tasks simultaneously. Our goal is adding our custom NER component to standard model for constituting a comprehensive model. The first step is to load two models that we mentioned and set the standard off-the-shelf spaCy model as main model. Then add the custom label to main string store to ensure that custom entities are recognized by the synthetic model. The process is shown in *Algorithm 17*.

```

1 main_nlp = spacy.load("en_core_web_sm") #Load pre-trained model
2 new_vocab = ["CUSTOM LABELS"]
3 for item in new_vocab:
4     main_nlp.vocab.strings.add(item) #add new vocabulary to main nlp
5 model
6 custom_ner_nlp = spacy.load("custom_ner_model")
7 ner = custom_ner_nlp.get_pipe("ner")
8 main_nlp.add_pipe(ner, name="custom_ner", before="ner") #custom pipe
   has primacy over the main NER

```

Algorithm 17: Example on how to add custom functions to spaCy pipeline

3.7.2 Refine the model

The optimization of the model starts from two aspects, one is to improve the accuracy of the model, and the other is to shorten the time for recognition. We use rule-based approach with high accuracy to achieve.

Text preprocessing has been done in Section 3.3 *Pre-processing* for each blackout text input to the model waiting to be recognized. Most of our target entities are composed of different nouns or noun groups, and thanks to property of spaCy, noun chunks in the article and their root words can be easily obtained, as shown in *Algorithm 18*.

```

1 for chunk in doc.noun_chunks: #get noun chunk, index and root
2     print(chunk.text, noun_chunks.start, noun_chunks.end,
           chunk.root.text)

```

Algorithm 18: Noun chunk analysis using spaCy

To determine whether there is an unlabeled target entity in the article, we divide it into the following two steps:

- Compare index position of the noun chunk with the entity that has been recognized by the model. If index positions coincide, it means that the target entity is correctly labeled and recognized by the model. Otherwise, the model may not recognize target entity. In this case, it is necessary to compare the root of the noun chunk with entities in the NE list for further judgment.
- Add those new noun chunks to the NE list after looking for entities in the NE list that have the same root, and then tokenized them to build a rule-based NER model. Meanwhile, sentences containing these new noun chunks can be also extracted from the blackout text for training a new custom NER model.

In this way, the NE list can be improved to the greatest extent and the defect of insufficient corpus data can be also compensated. Combining the rule-based NER model with the custom NER model trained by machine learning methods avoids the tedious work of secondary training and improves the accuracy of the model while shortening the recognition time.

Chapter 4

Results and Analysis

This chapter describes all the results including the processing of the blackout text, the classification of the text in the entity list, the general composition of the corpus, and the training and testing of the model. The last section covers the statistics after the model performs the NER task, and conducts a complete blackout analysis in terms of trends, causes, geographies, and scope of influence.

4.1 Result of Blackout List

Having a corpus of blackout events is not easy for several reasons. Since electricity is the foundation of people's livelihood and even related to the stable operation of the country, the investigation results of many accidents are secret. Documentation used in other work and other studies was either created specifically for project purposes or provided by power companies licensed to use the data. There are also many non-English texts, and even choosing a translation requires expert help. In order to deal with these difficulties, it was decided to obtain as much open-source information as possible on the Internet, most of which are news reports, and a few are professional power failure analysis reports.

In order to get the blackout results returned by a Google search as accurate as possible, we use the following as search terms:

- Massive/Prolonged/huge blackout
- Major power/electricity outage
- Power system cascading event
- Power cut/failure
- Lose power

The search engine returned results based on the given keywords, but some of the content was not related to the blackout event we are referring to here. These irrelevant materials include temporary amnesia due to drunkenness, events held to defend human rights, broadcasts of sports events and game server failures. In order to ensure the accuracy of the analyzed data, 234 satisfactory results were obtained after screening and filtering. Part of them is shown in *Figure 14*.

	title	link
0	What would happen in an apocalyptic blackout? ...	https://www.bbc.com/future/article/20191023-w...
1	Texas was minutes away from monthslong power o...	https://www.texastribune.org/2021/02/18/texas...
2	The Next 'Carrington Event' Could Cause a Glob...	https://www.inverse.com/article/34137-solar-s...
3	- BLACKOUT! ARE WE PREPARED TO MANAGE THE ...	https://www.govinfo.gov/content/pkg/CHRG-114h...
4	How Power Outages Are Affecting California B...	https://www.bloomenergy.com/bloom-energy-outa...
5	The 2003 Northeast Blackout--Five Years Later ...	https://www.scientificamerican.com/article/20...
6	How the Grid Copes When a Nuclear Power Plant ...	https://www.scientificamerican.com/article/ho...
7	Crain's Look Back: 2003 blackout was a dark da...	https://www.crainscleveland.com/energy-and-en...
8	Blackout hits Northeast United States - Histor...	https://www.history.com/this-day-in- /blackou...
9	Northeast blackout of 2003 - Wikipedia	https://en.wikipedia.org/wiki/Northeast_black...

Figure 14: Partial URLs about the blackout events

The texts obtained from the blackout list here can be roughly divided into news reports and power industry accident reports. In addition to web pages, texts are also in .pdf file. After the crawler and OCR convert the text into text documents in txt format and store them locally for later NER tasks.

4.2 Results of Text Classification

Here we present the classification results in *Section 3.4.2 Text Classification*. Since our purpose is to build a custom NER model, a corpus of entity-labeled is required and each entity's label needs to be specified. However, there is no specific classification standard for the power industry and there are many ways of classification, such as with a wide range and fine division. So it makes our work difficult to carry out. We had to artificially specify the names of the tags and then group them into the corresponding categories based on the properties of each entity itself. As mentioned before, we used both manual and automatic tool to classify, but they presuppose having a list of labels, as shown in *Table 3*.

Class Name	Class Label
Equipments and Installations	EQUIPMENT
Actions and Operations	ACTIONS
Mathematics, Physics	MATH
Errors, Data and Status	INFO
System Parameters	PARAMS
Market and Economics	COMMERCIAL
Systems and Networks	SYSTEM
Sources and Waste	SOURCES
Roles and Players	ROLES
Other/Unlabeled	OTHER
Standards and Rules	LAWS
Tests and Simulations	SIMULATIONS
Units of Measurement	UNITS
Inventions	INVENTIONS
Filters	FILTERS
Time	TIME
Organizations	ORG
Problems	PROBLEMS
Engineering Branches	ENGINEERING

Table 3: Category list of labels

4.2.1 Semi-automatic Classification

As shown in *Table 4*, we have a total of 1776 entities in the entity list, and each entity needs to be classified according to the categories in *Table 3*. There are 984 entities that can be automatically classified by algorithm, and the remaining 828 unlabeled entities can only be manually assigned to the corresponding categories.

Total entity	Classification method	
	Automatically	Manually
1776	984	828

Table 4: Number of tasks completed by different classification methods

From *Figure 15* we can clearly see the results of semi-automatic classification of entities using custom algorithms and manually. Among them, the entities belonging to the ‘EQUIPMENT’ class are the largest category in the entity distribution, which is also an important purpose of our training of custom NER. From the ordinate of the histogram, we can see that there are 19 label types. The reason is to avoid data type too single to have negative impact on the performance of the NER model trained.

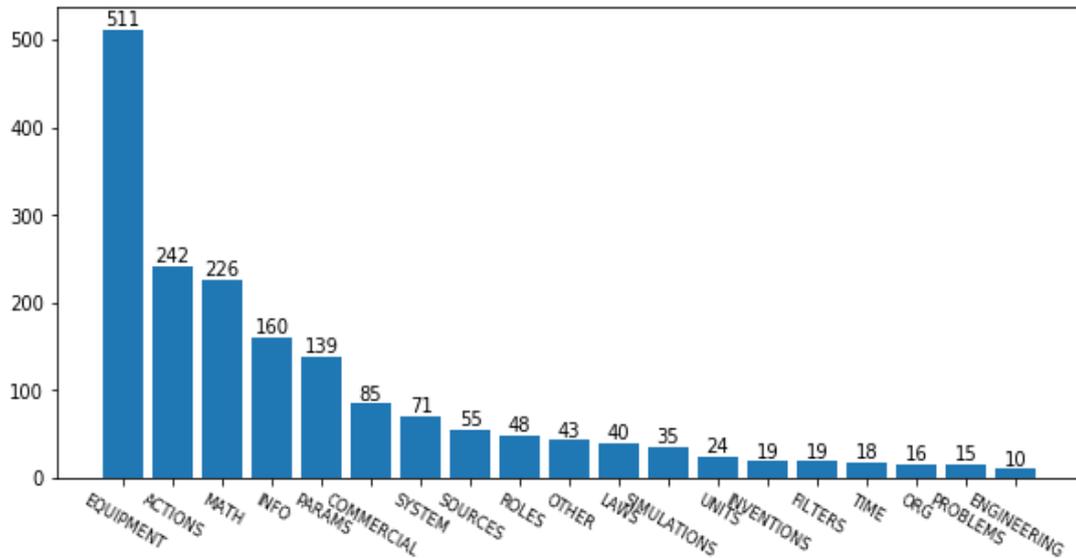


Figure 15: Results of semi-automatic classification

4.2.3 Choice of Classification Methods

Through the analysis of the above results, the two classification methods we adopted have their own merits. For semi-automated classification, the traditional method is used. The classification is completed by observing the word formation rules of the entity and using the NLP library. And another part that cannot be automatically classified by algorithms can only be done manually. Since many words have multiple attributes, the classification criteria are less common. For the method of training classification models using the online platform, it is time-saving and easy to use because of the intuitive and concise operation interface. However, it requires a large amount of training data with classification labels to make the model performed well.

To sum up, taking into account the classification criteria we have formulated is not very accurate. In our case, the reliability of the data determines whether the next steps can be carried out with relative accuracy. Although the semi-automated method is more time-consuming than the fully developed system platform, we decided to use the results of semi-automatic classification as a part of data to build an NE tagged corpus to train a NER model in next stages.

4.3 Size of the Tagged Corpus

As an important task, we tried to investigate how large corpus we should generate to obtain satisfactory performance. We collect as many sentences containing entities as possible in Ludwig. Ideally, we would like each entity to be contained by 6 to 7 sentences. However, in practice, not all entities can be fully queried to have been used in sentences. Because some of the entities we collected include highly specialized and compound words, some even involve mathematical operations such as $(n-1)$. The final corpus size is shown in *Table 5*. Since each entity that appears in the corpus would be labeled accordingly for NER model training. We only selected sentences that contain complete entities to construct the corpus and label entity in sentences accurately.

Total	Contain full entity	Contain partial entity
16795	12551	4244

Table 5: The number of sentences collected from Ludwig

Figure 18 shows the distribution of occurrences of entities in the corpus. There are 33.9% entities never appeared in the corpus, because they were not included in the sentences completely or the sentences related to them were not queried through Ludwig. In our case, this part of the entity is excluded. About 66% of the remaining entities are fully present in the corpus, and most of them appeared between 1 and 10 times. Only 6.19% of entities were found more than 50 times because they were or contained common words.

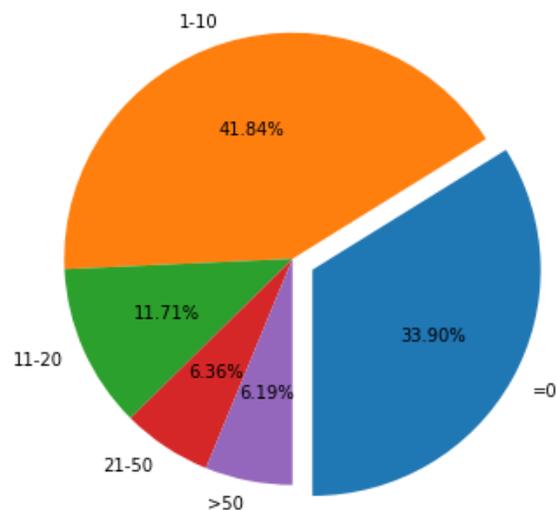


Figure 18: The frequency distribution of entities appearing in the corpus

4.4 Tuning Hyper-parameters

To get the best results, it is necessary to tune the hyper-parameters. We tried possible combination of different features and compared their results as a basis for choosing the best result.

The hyper-parameters for spaCy NER model as following:

- **Learning rate:** It determines step size for each iteration and controls how much the model weights are adjusted with respect the loss function. In general, its value range can be set between 0 and 1.
- **Regularization:** Deep learning may suffer from overfitting, i.e. high variance. Regularization helps to avoid the degree of overfitting of the model, or reduce network error.
- **Gradient clipping:** Because larger updates to the weights of neural network model can cause numerical overflow or underflow, i.e. gradients exploding. Gradient clipping is the clipping of gradient values outside of a preset range for updating error derivative before the network propagates the error backwards. It is used to update the weights to ensure that the network is non-divergent.
- **Optimizer:** It guides each parameter of the loss function (objective function) to update the appropriate size in the correct direction in the process of deep learning back propagation, so that the updated parameters make the value of the loss function continuously approach the global minimum.
- **Number of Epochs:** It is a hyper-parameter that indicates how many training times the machine learning algorithm needs to complete for the entire training data set. The overfitting conditions could be caused by higher epochs.
- **Batch size:** It determines the number of samples in a training session, and its size affects how well and how fast the model is optimized. A smaller batch size updates weights more frequently. Moreover, it directly affects the usage of GPU memory. That is, larger batch size requires more computational power.

- Dropout rate: It is an effective regularization technique used for preventing overfitting and helps boost the performance of the CNN. The stability and robustness of model are greatly improved because dropout is set to randomly stop certain neurons from participating in operations each training session. Generally speaking, the range of dropout is between 0.3 and 0.5.

Since those parameters with spaCy default settings, as shown in *Table 6*, have always performed well in the training of the spaCy model. Without changing the default parameter values of the platform, in our case, we only adjusted the three parameters, which are batch size, number of epochs and dropout rate.

Hyper-parameters	Values
Learning rate	0.001
Beta1	0.9
Beta2	0.999
L2 Regularization	1e-6
Optimization algorithm	Adam
Loss function	Mean Squared Error

Table 6: Default hyper-parameters of spaCy NER

The first parameter needs to tune is the batch size. We used mini-batch Gradient Descent as mentioned before. It divides into a data set less than a total number of samples. If the bath size is large, it may consume a lot of memory and needs more computational power. Since the batch size can be a number that doesn't change, or a schedule, like a sequence of compounding values, which has shown to be an effective trick [48]. To sum up, we tried using a composite rate of 1.001, starting from 1 or 4 and ending with 4, 32, 64, 128 as the maximum batch size, respectively.

Compounding batch sizes (start-max)	Precision (%)	Recall (%)	F1-Score (%)
1-4	95.408	96.161	95.783
1-32	95.021	97.169	96.083
4-32	94.949	96.733	95.833
1-64	94.462	96.597	95.518
4-64	94.271	96.325	95.287
1-128	94.486	96.107	95.290
4-128	95.417	96.923	96.164

Table 7: Testing results obtained from tuning batch size during training

From *Table 7*, we can see that the larger the gap, the better the performance of the model when there is little difference between the starting batch value and the maximum batch value. For example, composite batches with values of 1-32 and 1-64 have higher F1-score than 4-32 and 4-64 respectively, which means the model performed better. When the maximum batch value is increased to 128, the smaller the gap between the starting value and the maximum value, the better the model performs, and the highest value is obtained in the whole results, that is, the model obtained 96.164 F1-score when the composite batch size is 4-28.

As shown in *Figure 19*, the smaller the batch size, the slower the loss function value converges. When the maximum batch size is 128, the value of the loss function is a little smaller than others. This also verifies once again that a max batch size of 128 is the best choice for our model.

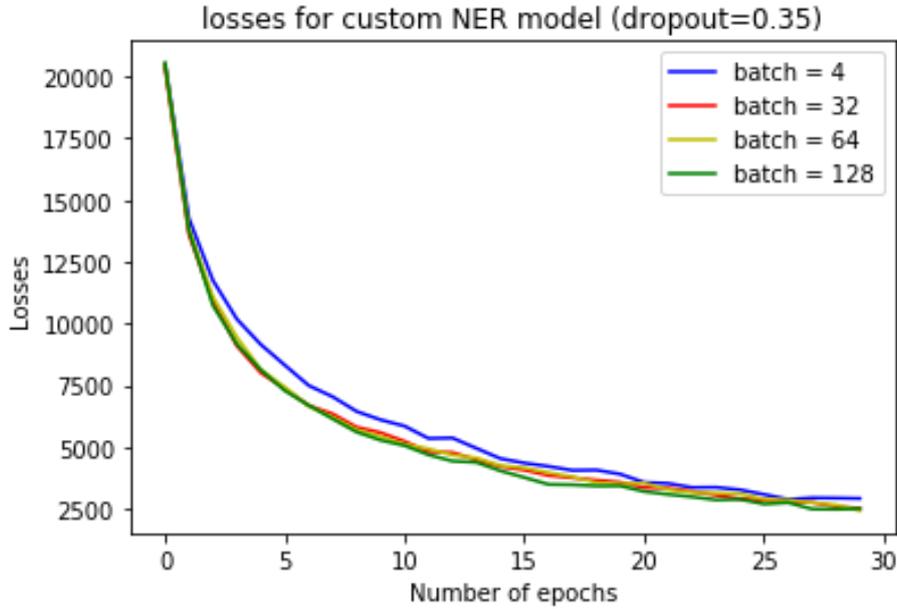


Figure 19: Losses of training model obtained from tuning batch size

The next parameter needs to tune is number of epoch. Each epoch includes one forward and backward pass of the entire training data set. The reason why we need more than one epoch is that in order to learn the pattern of the training dataset, the training dataset has to be looped through the neural network more than once. We tried 30, 60 and 100 epochs in our experimentation.

Number of epochs	Precision (%)	Recall (%)	F1-Score (%)
30	98.796	98.513	98.654
60	99.732	99.478	99.605
100	99.816	99.876	99.835

Table 8: The results of train set obtained from tuning number of epochs

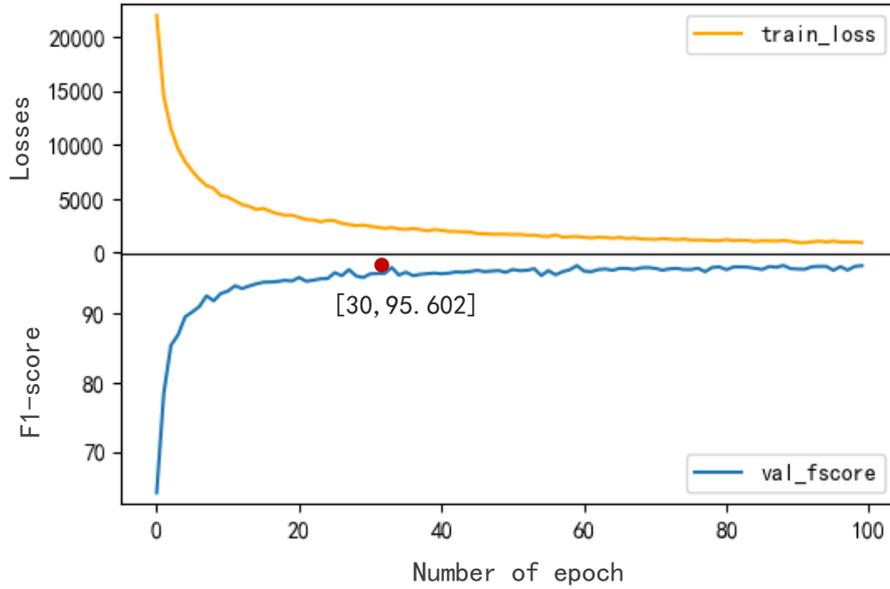


Figure 20: Losses and F1-score curves during training

The results shown in *Table 8* seem to be that the more epochs are trained, the better the performance of each evaluation metric. However, it is not rigorous to only look at the evaluation indicators of the training set. When we analyze the curves in *Figure 20*, it is not difficult to find that the losses decrease rapidly in the early stage of model training. When the epoch number exceeds 20, the loss value gradually stabilizes and decreases at a very slow rate as the epoch number increases. Conversely, the F1-score of validation set rises above 90 shortly with training beginning, and reaches its highest value of 95.602 when the number of epochs increases to 30. After that, although the loss value for training is still slowly decreasing, the F1-score for validation is almost stagnant, which means that the model has been overfitted. We have to stop the training process of the model before overfitting. So far, combined with the values of loss and testing results, we decided to use 30 as the optimal number of epochs.

Dropout technique randomly selects and removes the units during each iteration of training. When removing the units, in and out connections of the units are cut off. Dropout can be applied to the input layer and also the hidden layer. In

our case, 0.35 and 0.5 dropout rates have been used. *Table 9* shows the results of the custom NER model when we applied dropout to the network.

Dropout rate	Precision (%)	Recall (%)	F1-Score (%)
0.35	95.417	96.923	96.164
0.5	91.697	95.317	93.472

Table 9: Testing results obtained from tuning dropout rate during training

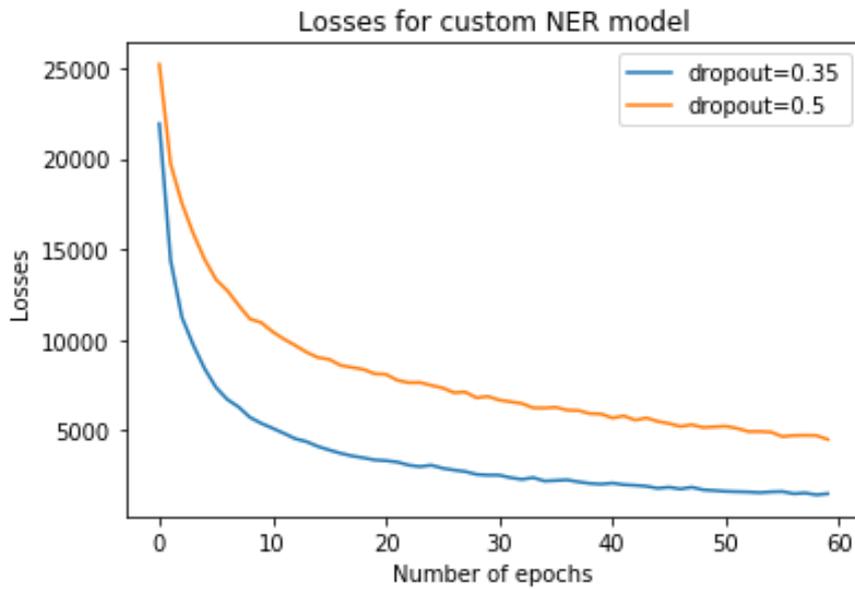


Figure 21: Losses of training model obtained from tuning dropout rate

When training the same number of epochs, the models with different dropout rates behave differently. Compared to the dropout rate of 0.5, our model achieves better performance with a dropout rate of 0.35. It can also be seen from *Figure 21* that a smaller dropout rate can make the loss curve converge faster. Although setting a larger dropout rate can avoid overfitting the model, it also means that we have to make the model learn more epochs, which will prolong the training time of the model. So in our case, we choose 0.35 as final dropout rate.

4.4.1 Summary of the Best Results

To summarize, we have tried to find the best results by tuning the hyper-parameters for custom NER model. We use the control variable method to find a suitable value for the model by adjusting the three model parameters, namely batch size, number of epochs and dropout rate, as shown in *Table 10*.

Hyper-parameters	Values
Batch size	4-128 (start-max)
Number of epochs	30
Dropout rate	0.35

Table 10: Best hyper-parameters of the custom NER model

Since implementing the multi-class classification and having more than two label classes. For each sentence on which the model is tested, we calculate precision, recall and f-score for each entity that the model recognizes. The values of these metrics for each entity are summed up and averaged to generate an overall score to evaluate the model on the test data consisting of 2003 sentences. The entity wise evaluation results can be observed below. It is observed that the results obtained have been predicted with a commendable accuracy. As shown in *Table 11*, for example, out of the 881 entities with the ‘EQUIPMENT’ tag, the NER model correctly tagged 96.935% (recall) of them. With precision, out of all the entities the model tagged with ‘EQUIPMENT’, only 93.743% of them had the ‘EQUIPMENT’ tag. Most of labels have F1-score around 95%, except label ‘ORG’ below 90%. And three labels have reached a high F1-score of 100%, which are ‘PROBLEMS’, ‘SIMULATIONS’ and ‘INVENTIONS’.

Labels	Precision (%)	Recall (%)	F1-Score (%)	Sample size
EQUIPMENT	93.743	96.935	95.313	881
MATH	97.401	97.668	97.534	729
OTHER	97.786	98.330	98.057	539
ACTIONS	95.090	97.872	96.461	376
PARAMS	94.241	96.515	95.364	373
INFO	98.052	96.795	97.419	156
SYSTEMS	92.568	96.479	94.483	142
SOURCES	93.662	94.326	93.993	141
UNITS	98.246	94.915	96.552	59
PROBLEMS	100.0	100.0	100.0	52
ROLES	98.113	100.0	99.048	52
SIMULATIONS	100.0	100.0	100.0	35
COMMERCIAL	90.625	96.667	93.548	30
FILTERS	90.0	90.0	90.0	30
TIME	96.154	92.593	94.340	27
LAWS	100.0	94.444	97.143	18
ENGINEERING	100	93.750	96.774	16
INVENTIONS	100.0	100.0	100.0	11
ORG	80.0	66.667	72.727	6

Table 11: The results for testing data with best hyper-parameters

In general, the number of samples is proportional to the F1-score, but combined with what is shown in *Figure 22*, it does not seem to fit this statement at all in our model. It might be caused by the uneven distribution of the amount of data per label. For some categories with full F1-scores, this means that the model is overfitting on these categories due to the low number of samples during training of the model over multiple epochs. But in order for a large number of class labels to be learned well by the model, we have to sacrifice these few labels that are less important to us.

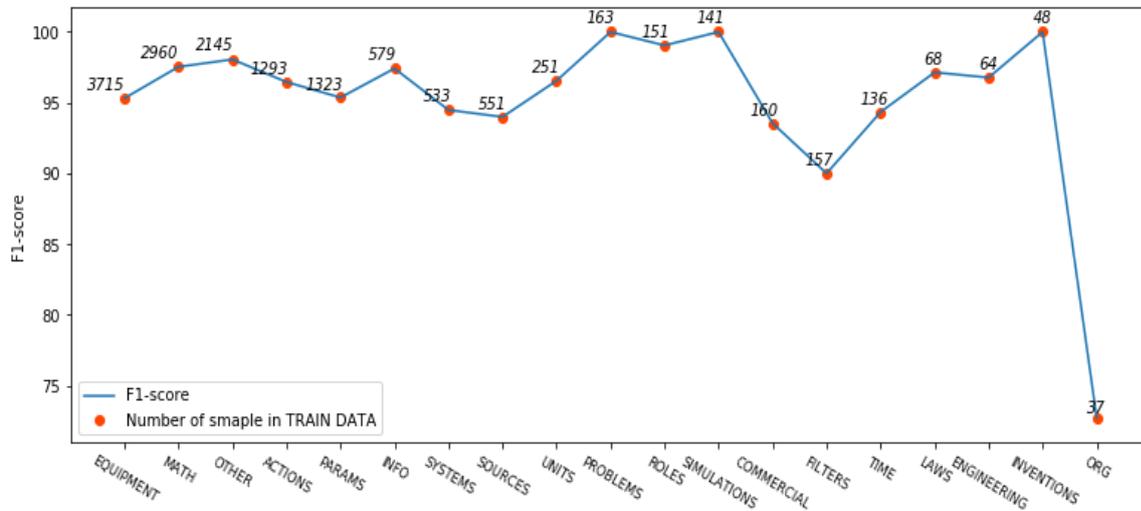


Figure 22: The performance of each label with the best hyper-parameters

4.5 Result of Model Test

In this work, the following three models were built using spaCy:

- `en_core_web_sm` which a pre-trained model to recognize built-in labels.
- Custom NER model to recognize custom labels.
- The comprehensive model to identify all entities.

Here we selected sentences with their corresponding labels from the corpus for training according to the functional properties of the model. And each model is trained until the loss stabilizes.

Type	Pre-trained Model		Custom NER Model		Pre-trained model & Custom NER model	
	train	test	train	test	train	test
#Annotated	5932	1328	15303	3524	20780	4729
#Predicted	9024	2025	15019	3665	23502	5414
Diff	3092	697	-284	141	2722	685

Table 12: Comparison between predicted and annotated NE using spaCy

Table 12 shows the correctness of the three models by spaCy method. The Annotation and Prediction rows show the count of data marked for training and the number of named entities predicted by the model, respectively. "Diff" represents the difference between "Annotated" and "Predicted". It is worth noting that the negative value in "Diff" is due to the model finding fewer entities than the number of annotation labels. The reason why the pre-trained model provided by spaCy can recognize more entities than our annotated labels is because it contains many built-in labels. For custom labels, the number of predicted labels missing is about 1.86% of training. Here it is necessary to state the exact boundary problem of spaCy's predicted phrases. The prediction may not exactly match the annotation, but it provides reference boundary or good estimate of the entities. If we count the exact number of positions it might be wrong. Thus, we lowered the metric and only counted the number of predictions. For the combined model composed of the first two models, it has the performance of predicting both built-in labels and custom labels.

There are 18 built-in labels in the small English model pre-trained by spaCy, among which labels representing locations (NORP, GPE, LOC), labels representing time (DATE, TIME) and labels representing numbers (MONEY, CARDINAL, ORDINAL) are helpful for blackout analysis. For the 19 custom labels, we are most concerned about 'EQUIPMENT', 'SYSTEMS', 'PROBLEMS', 'ACTIONS', 'PARAMS'. When creating a comprehensive model that recognizes all the types of labels including spaCy's built-in labels and custom labels, we obtained over 90% accuracy in both training and testing for each label.

Figure 23 shows the accuracy results for the most important labels mentioned above in the comprehensive model.

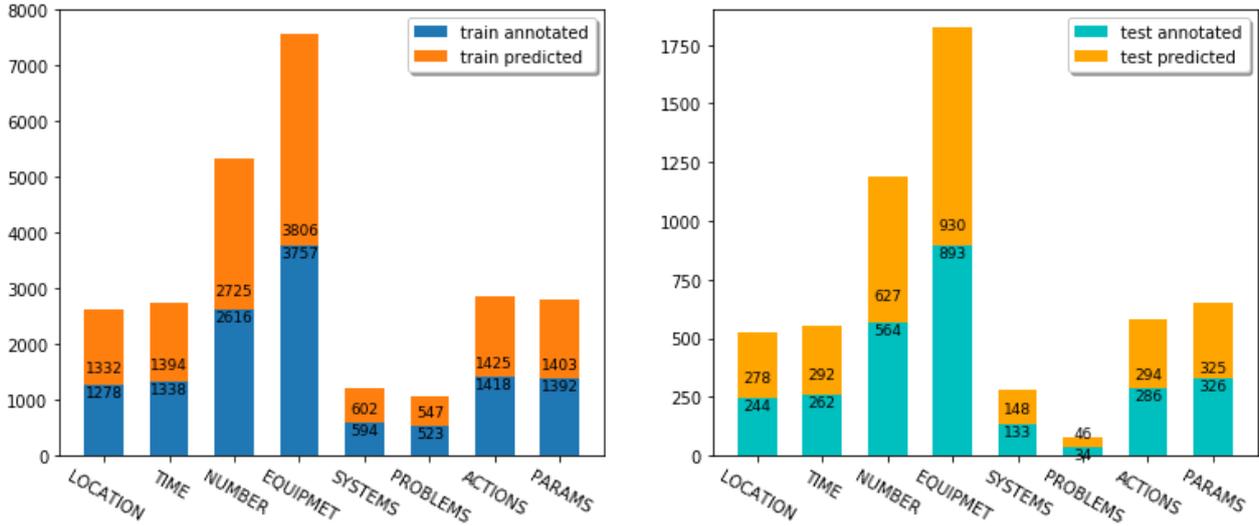


Figure 23: Annotated and predicted results for most important tags

Figure 24 shows the recognition results for an example. Figure 24(a) is the result of spaCy pre-trained model, which can well identify the time, place and digital content in the text. While Figure 24(b) represents the result processed by the custom NER model. The result of comprehensive model is shown in Figure 24(c). It can be clearly seen that the model performs well after integrating two well-trained models, and the important entities in the text that are useful for blackout analysis are accurately identified and marked. In addition to fully displaying the content recognized by the model, in order not to interfere with unimportant entities, we can also set the label categories we want to view based on different scenarios

on March 11 1999 **DATE** , widespread power outage occur in Brazil **GPE** affet 97 million **CARDINAL** people. chain reaction start lightning strike occur at 10:16 pm **TIME** at substation in Bauru **GPE** , São Paulo State **GPE** cause most of 440 **CARDINAL** kv circuit at substation to trip. South São Paulo **GPE** consumer experience overfrequency, cause because have more generation than power load, mostly because Itaipu **ORG** now connect to sub system, problem automatically solve by generator reduce power load.

(a)

on March 11 1999 , widespread power outage PROBLEMS occur in Brazil affect 97 million people. chain reaction PROBLEMS start lightning strike PROBLEMS occur at 10:16 pm at substation SYSTEMS in Bauru, São Paulo State cause most of 440 kv UNITS circuit at substation SYSTEMS to trip. South São Paulo consumer ROLES experience overfrequency PROBLEMS , cause because have more generation PARAMS than power load PARAMS , mostly because Itaipu now connect to sub system SYSTEMS , problem automatically solve by generator EQUIPMENT reduce power load PARAMS .

(b)

on March 11 1999 DATE , widespread power outage PROBLEMS occur in Brazil GPE affect 97 million CARDINAL people. chain reaction PROBLEMS start lightning strike PROBLEMS occur at 10:16 pm TIME at substation SYSTEMS in Bauru GPE , São Paulo State GPE cause most of 440 CARDINAL kv UNITS circuit at substation SYSTEMS to trip. South São Paulo GPE consumer ROLES experience overfrequency PROBLEMS , cause because have more generation PARAMS than power load PARAMS , mostly because Itaipu ORG now connect to sub system SYSTEMS , problem automatically solve by generator EQUIPMENT reduce power load PARAMS .

(c)

Figure 24: Example results of NER part I

It is worth mentioning that the processing results of different types of text NER models are different. For news report-type text, most of the available information can usually be obtained from the news headline because of its highly general nature. For example, each text has a publication time. Even if the specific time of the blackout event is not specified in the article, we can judge it based on the publication time combined with the content in the text. Rough statistical analysis can be done using such text. And for professional accident reports there is usually no explanatory text especially cascading blackouts. *Figure 25* shows example results of professional incident investigation report.

the Generation PARAMS Capacity mere 1349 DATE mw UNITS demand have touch 2839 CARDINAL mw UNITS collapse system by 10:05 AM TIME major part of Mumbai GPE and Thane have suffer one of bad power outage PROBLEMS in decade

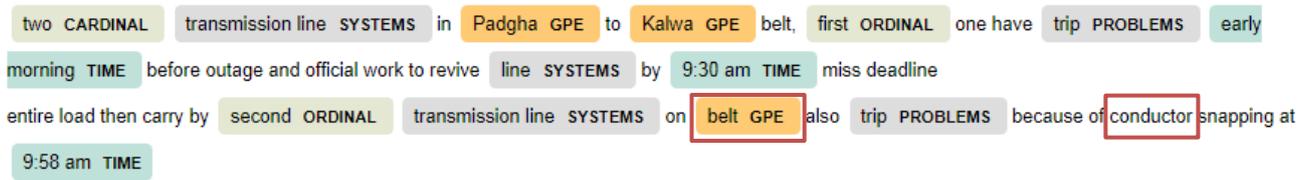
(a)

Operation backup protective relay EQUIPMENT take 230 CARDINAL kv UNITS line SYSTEMS out of service when load on line SYSTEMS exceed 375 CARDINAL mw UNITS relay EQUIPMENT set

(b)

this lead to cascade effect PROBLEMS trip line SYSTEMS interconnect Kalwa to Kharghar GPE substation SYSTEMS at 10 am TIME the Pune Kharghar 400 kv UNITS line SYSTEMS , another major line SYSTEMS , also trip PROBLEMS due to overload PROBLEMS , result in severe load drop ACTIONS in Mumbai GPE system

(c)



(d)

Figure 25: Example results of NER part II

In the obtained results we can see that most of the entities are correctly identified and labeled. It is attributed to high-quality corpus and text preprocessing. Since there are cases where sentences are of a schematic and fundamental nature, it is more important to have high-performance fundamental tasks that can correctly identify all tokens. In fact, in cases where expressions are not recognized in the correct way, errors are mainly due to tokenization not properly separating the text and insufficient corpus data resulting in incorrectly or missing entities being recognized. It must be admitted that building a pattern is a delicate operation that must be done precisely by an expert, or in any case with the help of in-depth research, to avoid omission or misidentification.

It is important to emphasize that the selected text has a considerable difficulty compared to the documents on which the model was trained. In addition to the refinement of vocabulary, there are also the use of abbreviations and synonyms. Furthermore, while the test results demonstrate that our model can be used for power text, it is partly due to the fact that the information extraction activity uses a rule-based approach to make the model explicit as the target entity. Fortunately, since those modules we use to build our models are based on neural networks, it is possible to train on more data to further improve performance. If necessary, it is possible to leverage more complex modules or optimize what already exists in this work.

4.6 Blackouts analysis

This section describes the outage analysis that has been performed. We will carry out statistical and analysis work from the aspects of the time, cause and place, and the scope of influence of the selected accident.

4.6.1 Temporal Analysis

To analyze the temporal trend of power outage incidents, we visualized the time in years of incident occurrence in the text obtained. We extract the time information from more than two hundred texts through the NER task, and obtain 178 independent blackout events through cross-validation due to the existence of multiple articles describing the same event.

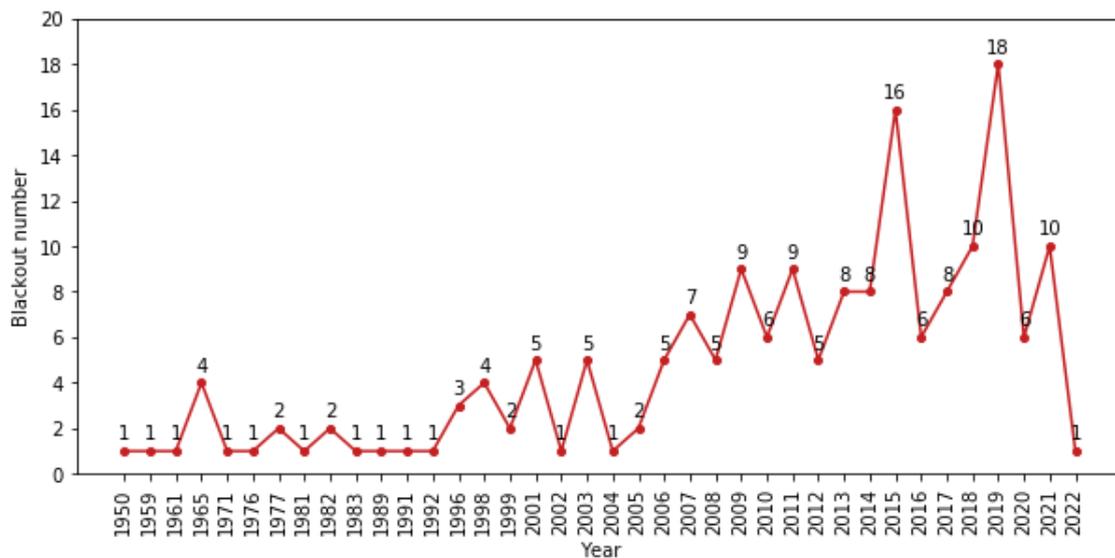


Figure 26: The trend of selected blackouts in power systems

From *Figure 26*, it can be seen that the collected outage data cover a range from 1950 to 2022, and show an upward trend with the changing times. Before 1996, the frequency of blackout has historically remained at almost the same level every year, almost once a year. The frequency kept in a moderately high level kept from 1996 to 2006. The blackout number has soared to a high level in the following decades since 2007, almost all of which are around 10 times. In 2019, it

reached as many as 18 times, and the sudden decrease in 2022 is due to the fact that this year has just begun. The rising trend of power outages in recent years shows that a variety of threats are increasingly affecting today's power systems.

4.6.2 Reason Analysis

In the power industry, potential causes that could lead to an unexpected event that compromises the power system are referred to as threats. Generally speaking, threats can be divided into four categories: natural threats, accidental threats, malicious threats and emerging threats [49].

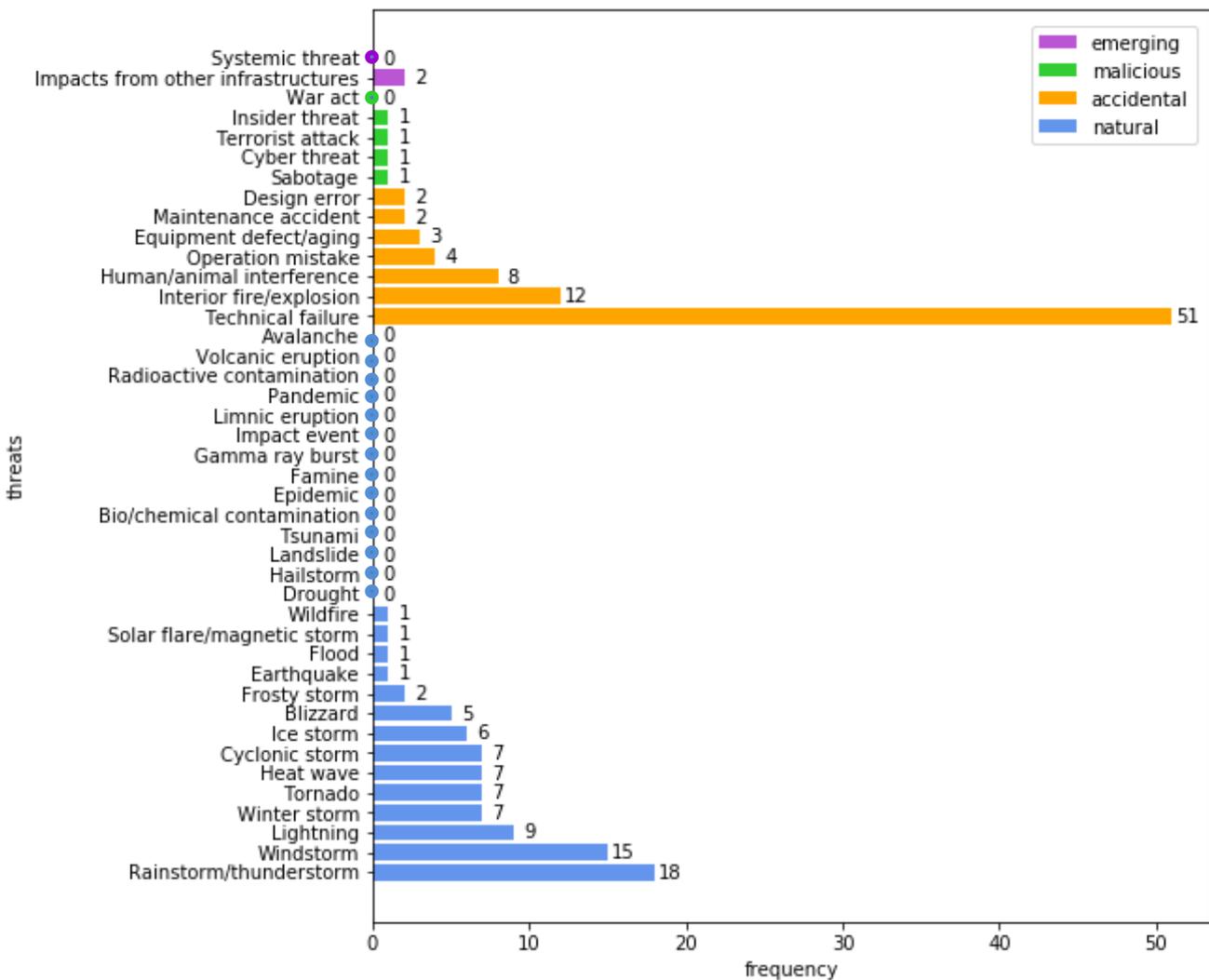


Figure 27: Statistics of the causes of the selected blackouts

The specific event names ascribed to the four threats are shown in *Figure 27*. It is easy to see that natural threats have the most sub-categories, including extreme weather-induced changes in climate conditions, geological hazards, non-human-caused disasters, and health disasters, etc. Most of the power grid security problems caused by natural threats in the power outage events we selected are concentrated in the part caused by extreme weather, especially the power outage accidents caused by windstorms and rainstorm/thunderstorms reached 15 and 18 times respectively. The frequency of blackout caused by geological disasters, space disasters and non-human-caused fires stay at a steady low level. Every subcategory of incidental threats was involved in the blackouts we selected, with the most frequent cause of power system failures being technical failures. Generally speaking, we describe a technical failure as a breakdown or ceasing of equipment caused by internal factors of the equipment with direct malfunctioning effects on each sector in power systems. Since technical failure covers a wide range, it has the longest column in the histogram, reaching 51 times. The frequency of unnatural fires or explosions caused by operating equipment/devices and installation failures caused by humans or animals is maintained at a high level. The rest of the accident causes that belongs to the category of operational fault are all around twice. The power system is interdependent with other system infrastructure. There are cases where faults in other systems spread and cause uncertainty of power system, and we call this type of incident emerging threats. As can be seen from the histogram, emerging and malicious threats are rare. Among them, malicious threats are all caused by human subjective operation or control. And according to the three layers of power system, malicious threats can be classified into physical threat, human threat, and cyber threat.

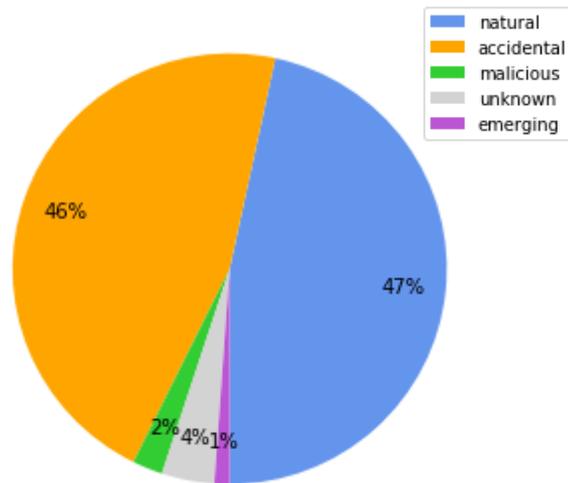


Figure 28: Percentages of the causes for selected blackouts

Overall, in *Figure 28*, natural and accidental threats account for more than 90% of the 178 power outages counted, which means that these two threats can be used as the main reasons for power outages. There are 4% of unknown represent a portion of the outage text that does not state the cause and has not yet found the truth.

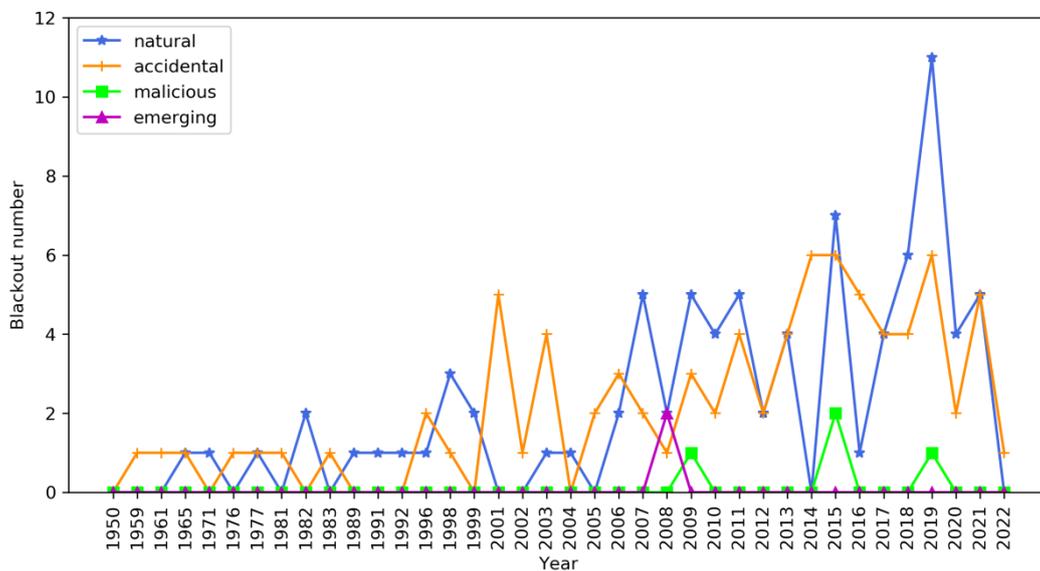


Figure 29: Trends of four threat categories

As shown in *Figure 29*, the four categories of threat trends are viewed in conjunction with a year timeline. Although the curve fluctuates to a certain extent, nature and accident are always in the lead and peaked in 2019. The emerging and malicious threats have only gradually affected the instability of the power system since 2008 and 2009. This means that with the progress of society and the development of the industrial industry, the power system is gradually forms an interrelated state with other systems. In this case, the stable operation of the power grid can be affected, for example, by cyber-attack or other industrial systems. Malicious threats and other emerging threats in the last decade are starting as a reason for blackouts.

4.6.3 Geographical Distribution Analysis

We also counted the regions where power outages occurred. *Figure 30* shows the worldwide distribution of the blackouts we selected. Most of the blackouts are concentrated in North America, including the United States and Canada. Among the 178 blackouts, the number of occurrences in the United States is as high as about 80. This was followed by Canada in the second batch, with power outages occurring between 16 and 32 times. The rest of the regions with very low frequency of grid power outages are mainly concentrated in Europe, Asia, South America and Oceania.

However, it is obviously that the blackout events we selected cannot fully represent the global power outage trend and frequency, which is caused by the singularity of the data. Since the data is in English as the search language, the obtained data is all in English, and there is very little outage information in English for many countries and regions that do not use English as the official language. This is the main reason why the vast majority of power outages in the global distribution map are concentrated in North America, while other regions are almost at the lowest power outage frequency level.

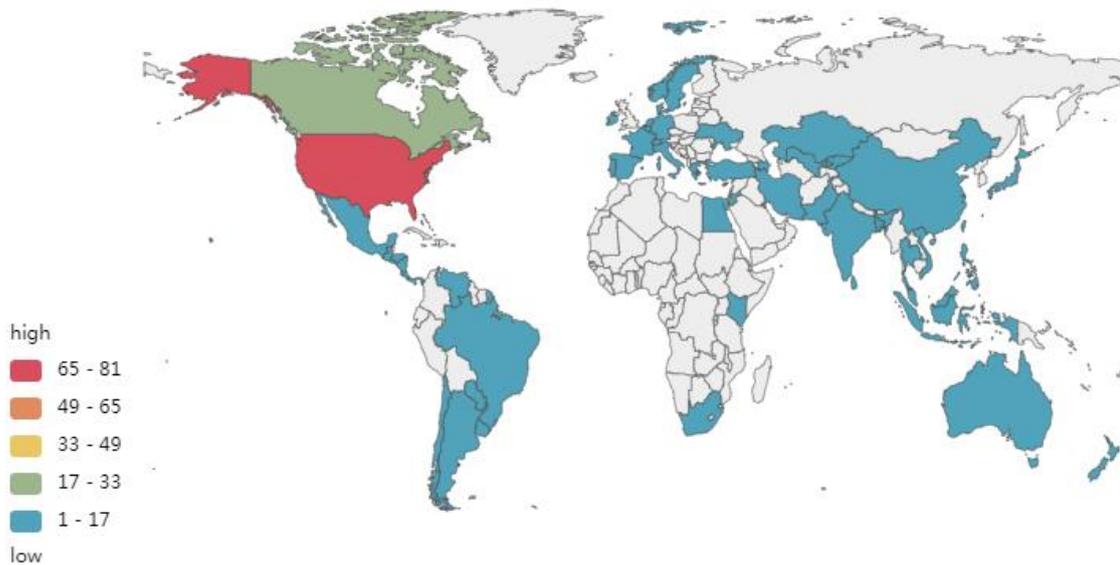


Figure 30: Global selected blackouts distribution

Among the power outages we selected, the sample size of blackout events in the United States is relatively comprehensive compared to other countries. Therefore, we think that it is worth to conduct a detailed analysis of the grid outage events that have occurred in different states in the United States.

Figure 31 is a map of the distribution of power outages we selected in the United States. On the whole, the southwestern coastal areas and northeastern regions of the United States are areas with high incidence of power outages, especially California and New York. Because the Northeast borders Canada and the two countries also have some common industrial infrastructure, grid incidents often affect each other. This has become one of the reasons for the high frequency of power outages in the northeastern United States. Due to the special geographical location of the western and southern region, climate change such as the formation of cyclones in the Pacific Ocean is one of the main reasons for power outages. In contrast, the southeastern and central regions of the United States have had few major grid incidents.

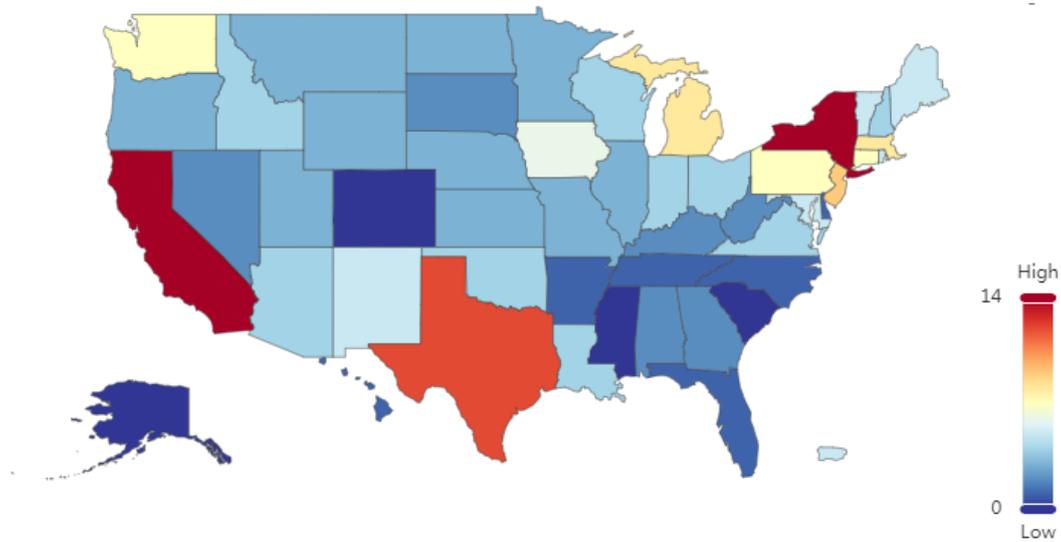


Figure 31: USA selected blackouts distribution

To further explore the relationship between geography and the four threats, 15 states with more than 5 blackouts were selected as objects from the 78 blackouts in the United States. As shown in *Figure 32*, in addition to Puerto Rico, 14 of the 15 states selected suffered from natural threats causing blackouts more frequently than accidental threats. Especially in Texas and Iowa, the natural threats were the culprit of blackouts in the text we collected. It is not difficult to find that these 14 states are located in the coastal belt and northeast of the United States. This means that locations with a greater likelihood of climate impact are also at higher risk of blackouts. The impact of malicious and emerging threats on the U.S. power system is not clearly represented.

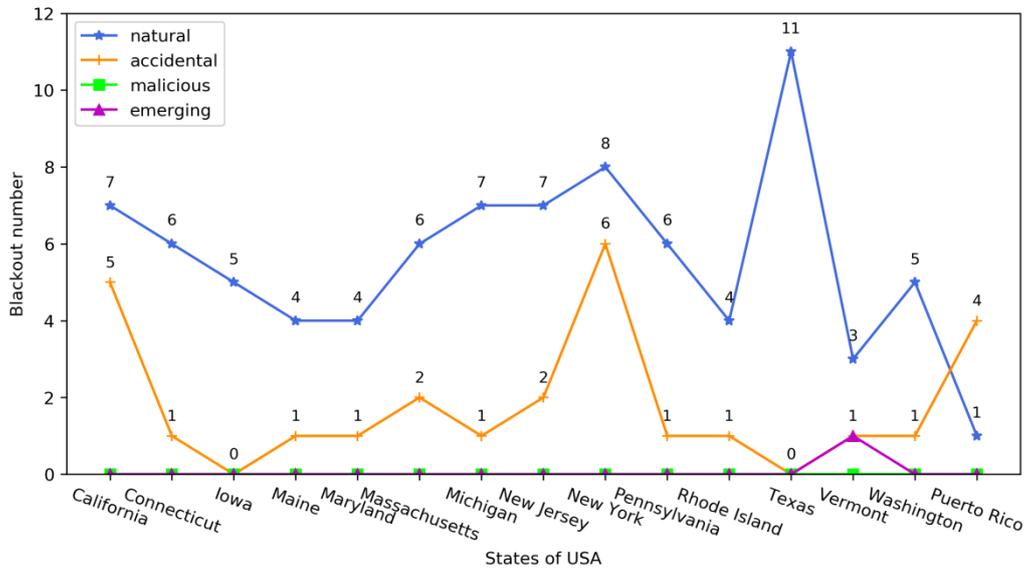


Figure 32: Frequency of four threats in selected USA states

4.6.4 Scope of Impact Analysis

Typically, a blackout event that affects more than one million people can be considered a major blackout. There are 94 out of 178 blackouts which we counted belong to wide-scale blackouts. Due to limited space, we only selected 33 cases that affected more than 10 million people, as shown in *Table 13*.

Over 100 million people were plagued by losing power in four Asian countries, and the top 4 are all in India. Almost all of them are due to technical failures. There are three events involved more than 3 countries and two of them were caused by rainstorm. The most widespread being the outage on November 4, 2006, which affected 15 million people in seven European countries due to power system design error. In general, 79% of the causes of widespread blackouts affecting more than 10 million people were accidental threats, 15% were due to natural threats, and only 6% were accidental threats.

People affected (millions)	Locations	Dates	Threats
700	India	July 31, 2012	Technical failure
620	India	July 30, 2012	Technical failure
230	India	January 1, 2001	Technical failure
230	India	January 2, 2001	Technical failure
212	Pakistan	January 10, 2021	Technical failure
160	Bangladesh	November 1, 2014	Technical failure
140	Pakistan	January 26, 2015	Terrorist attack
120	Indonesia	August 4, 2019	Technical failure
100	Indonesia	August 18, 2005	Technical failure
97	Brazil	March 11, 1999	Lightning
70	Turkey	March 31, 2015	Interior fire/explosion
60	Brazil,Paraguay,Uruguay	November 10, 2009	Rainstorm
60	Australia	January 30, 2010	Lightning
56	Italy	September 28, 2003	Windstorm
55	US,Canada	August 14, 2003	Operation mistake
53	Brazil	February 4, 2011	Technical failure
48	Argentina,Paraguay, Uruguay,Chile,Brazil	June 16, 2019	Rainstorm
40	Philippines	May 21, 2002	Technical failure
35	Philippines	April 7, 2001	Technical failure
32	Venezuela	March 7, 2019	Equipment defect/aging
30	US,Canada	November 9, 1965	Maintenance accident
30	Iran	May 20, 2001	Technical failure
30	Venezuela	July 22, 2019	Sabotage
24	China	August 15, 2017	Operational mistake
20	Egypt	September 4, 2014	Technical failure
20	India	October 12, 2020	Technical failure
21	Sri Lanka	March 13, 2016	Technical failure
21	Sri Lanka	August 17, 2020	Technical failure
15	France,Germany,Italy, Netherlands,Belgium, Spain,Portugal	November 4, 2006	Design error
15	Chile	March 14, 2010	Technical failure
10	Brazil	March 21, 2018	Technical failure
10	Kenya	June 7, 2016	Human/animal interference
10	Jordan	May 21, 2021	Technical failure

Table 13: List of selected wide-scale blackouts

Chapter 5

Conclusion and Future Work

In this chapter, the previous work are summarized and discussed. Based on the limitations of the proposed model, some possible future work is stated.

5.1 Conclusion

The thesis explores a NLP methodology for extracting information and preprocessing text from unstructured blackout materials. In particular, we demonstrate prototype using machine learning complemented by rule-based approach to recognize and extract content of interest to users from blackout records.

Therefore, the first step of the work is to crawl the web pages with the keyword "blackout" returned by the Google search engine, and preprocess them to construct a text dataset. Then, the specific vocabulary required building the NER model, the generation of the original corpus, and the preprocessing methods for annotation are introduced. To further extend potential of the model for extracting more information using NLP techniques, we customized pipeline and integrated a well-trained English language module developed by spaCy to build model perform name entity recognition task on the blackout dataset. The performance of

the model is discussed and the statistical results of selected blackout events are analyzed.

The results of this work may contribute to a deep understanding of blackout incidents that have occurred in the history, including failure analysis of the power system and multiple factors affecting the stable operation of the power grid, etc. It can even provide an effective analytical tool for those working in the power industry. In fact, the proposed methodology can not only be used to study and analyze outage-related information in the power industry, but can also be generalized to identify entities in other fields to study and analyze related topics. For those algorithms and techniques could be utilized for obtaining and preprocessing data.

5.2 Limitation and Future Work

The main limitation of the work is the lack of documentation of outage incidents and sufficient information to optimize the model. A general deficiency can be noticed from the collected dataset that the vast majority of recorded blackout incidents are from more developed regions of the world, which making the results not representative of the whole phenomenon. Secondly, the information that can be collected on the Internet is quite discrete, and multiple documents describe the same event or an event that is not completely recorded often occurs, which greatly increases the difficulty of processing and statistics. Moreover, the web crawling method we used does not work for the websites which need to pay. So the dataset is not comprehensive, although we have tried to find related information through various channels as many as possible.

Despite some regrets of data, algorithms and results, important results have been achieved in data extraction and collection methods. We can conclude that the results obtained are consistent with our expectations.

The thesis focuses on study the methods and tool that will be used at the power information extraction and processing stages to ensure abundant foundation is defined for conducting effective blackout analysis. There are some

issues need to be tackled with in the future work. Here several aspects can be investigated and explored are as flowing:

- Partner with a professional organization in the power industry or apply for a license to use the full dataset annotated by experts in the field.
- Use spaCy or other frameworks as input to train and generate a classification model to label new text. Combined with supervised techniques, it can improve the accuracy of the analysis and provide better results.
- Probe into more complex preprocessing models such as BERT with the help of transfer learning, and introduce the concept of word vector to make up for the lack data of original corpus by calculating the similarity between entities.

Bibliography

- [1] FERC and NERC staffs, Arizona-Southern California Outages on September 8, 2011: causes and recommendations, Tech. rep., Federal Energy Regulatory Commission (FERC) & North American Electric Reliability Corporation (NERC) (2012). doi:10.1093/toxsci/kft047.
- [2] National University System Institute for Policy Research, Blackout losses could top \$100 million (2011). <http://www.nusinstitute.org/press/in-the-news/Blackout-losses-could-top-100million.html>
- [3] The massive blackouts in India—the largest blackout in the world. <http://old.china5e.com/special/show.php?specialid=575> Accessed 01 Aug 2012
- [4] Halilcevic Suad S, Gubina Ferdinand, Gubina Andrej F. Prediction of power system security levels. IEEE Trans Power Syst 2009;24:368–77.
- [5] Rosati, S. Natural Language Processing. Politecnico di Torino.
- [6] Tono, R. «Natural Language Processing e tecniche semantiche per il supporto alla diagnosi: un esperimento». Tesi di Laurea Magistrale. Università degli studi di Padova, 2010.
- [7] Palshikar, G. K. «Techniques for Named Entity Recognition: A Survey». In: vol. 1.Gen. 2012, pp. 191–217. ISBN: 9781466636057. DOI: 10.4018/978-1-4666-3604-0.ch022.
- [8] LI Meng, LI Yanling, LIN Min. "Review of Transfer Learning for Named Entity Recognition." Jisuanji Kexue Yu Tansuo 15.2 (2021): 206-18.
- [9] K. Humphreys et al., "University of sheffield: Description of the laSIE-II system as used for MUC-7", Proc. 7th Message Understanding Conf., pp. 1-20, 1998.
- [10] G. Krupka and K. IsoQuest, "Description of the nerOWL extractor system as used for MUC-7", Proc. 7th Message Understanding Conf., pp. 21-28, 2005.

- [11]W. J. Black, F. Rinaldi and D. Mowatt, "FACILE: Description of the NE system used for MUC-7", Proc. 7th Message Understanding Conf., pp. 1-10, 1998.
- [12]NADEAU D, SEKINE S. A survey of named entity recognition and classification[J]. Computational Linguistics, 2007,30(1): 3-26
- [13]LI J, SUN A X, JOTY S R. SegBot: a generic neural text segmentation model with pointer network[C]//Proceedings of the 27th International Joint Conference on Artificial Intelligence, Stockholm, Jul 13-19, 2018: 4166-4172.
- [14]Grewal, Jasleen K, Martin Krzywinski, and Naomi Altman. "Markov Models — Hidden Markov Models." *Nature Methods* 16.9 (2019): 795-96.
- [15]De Ville, Barry. "Decision Trees." *Wiley Interdisciplinary Reviews. Computational Statistics* 5.6 (2013): 448-55.
- [16]J. N. Kapur, *Maximum-Entropy Models in Science and Engineering*, Hoboken, NJ, USA:Wiley, 1989.
- [17]M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt and B. Scholkopf, "Support vector machines", *IEEE Intell. Syst. Their Appl.*, vol. 13, no. 4, pp. 18-28, Jul./Aug. 1998.
- [18]J. D. Lafferty, A. McCallum and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data", *Proc. 18th Int. Conf. Mach. Learn.*, pp. 282-289, 2001.
- [19]Y. LeCun, Y. Bengio and G. Hinton, "Deep learning", *Nature*, vol. 521, no. 7553, 2015.
- [20]J. Li, A. Sun, J. Han and C. Li, "A Survey on Deep Learning for Named Entity Recognition," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 1, pp. 50-70, 1 Jan. 2022.
- [21]Strubell, Emma, Patrick Verga, David Belanger, et al. Fast and accurate entity recognition with iterated dilated convolutions[C]//Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Sep 9-11, 2017. Stroudsburg: ACL,2017: 2670-2680.
- [22]Huang, Zhiheng, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF models for sequence tagging[J]. arXiv:1508.01991, 2015.

- [23] COLLOBERT R, WESTON J, BOTTOU L, et al. Natural language processing (almost) from scratch[J]. *Journal of Machine Learning Research*, 2011, 12: 2493-2537.
- [24] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[C]//*Proceedings of the Annual Conference on Neural Information Processing Systems*, Long Beach, Dec 4-9, 2017. Red Hook: Curran Associates, 2017: 5998-6008.
- [25] SHEN Y Y, YUN H, LIPTON Z C, et al. Deep active learning for named entity recognition[C]//*Proceedings of the 2nd Workshop on Representation Learning for NLP*, Vancouver, Aug 3, 2017. Stroudsburg: ACL, 2017: 252-256.
- [26] YANG Z, SALAKHUTDINOV R, COHEN W. Multi-task crosslingual sequence tagging from scratch[J]. *arXiv:1603.06270*, 2016.
- [27] Apache OpenNLP. 2004. Apache Software Foundation. Retrieved from <https://opennlp.apache.org>.
- [28] Shelar, Hemlata, Gagandeep Kaur, Neha Heda, and Poorva Agrawal. "Named Entity Recognition Approaches and Their Comparison for Custom NER Model." *Science & Technology Libraries* (New York, N.Y.) 39. 3 (2020): 324-37.
- [29] SpaCy. (2020) Facts & Figures. Retrieved 26 March 2020. [Online]. Available: <https://spacy.io/usage/facts-figures>
- [30] SpaCy. "Library architecture," 2022. [Online]. Available: <https://spacy.io/api#section-nn-model>
- [31] Y. Pradeep, S. A. Khaparde and R. K. Joshi, "High Level Event Ontology for Multiarea Power System," in *IEEE Transactions on Smart Grid*, vol. 3, no. 1, pp. 193-202, March 2012
- [32] Santos G, Pinto T, Morais H, Sousa TM, Pereira IF, Fernandes R, et al. Multi-agent simulation of competitive electricity markets: autonomous systems cooperation for european market modeling. *Energy Conversion and Management* 2015;99:387–99. doi: 10.1016/j.enconman.2015.04.042.
- [33] Zhou L, Pan M, Sikorski JJ, Garud S, Aditya LK, Kleinlanghorst MJ, et al. Towards an ontological infrastructure for chemical process simulation and optimization in the context of eco-industrial parks. *Appl Energy* 2017;204:1284–98. doi: 10.1016/j.apenergy.2017.05.002.

- [34]Morbach J, Yang A, Marquardt W. OntoCAPE-A large-scale ontology for chemical process engineering. *Eng Appl Artif Intell* 2007;20(2):147–61. doi: 10.1016/j.engappai.2006.06.010.
- [35]Devanand A, Karmakar G, Krdzavac N, et al. OntoPowSys: A power system ontology for cross domain interactions in an eco industrial park [J]. *Energy and AI*, 2020, 1: 100008.
- [36]Ram Gopal, James R. Marsden, Jan Vanthienen. Information mining — Reflections on recent advancements and the road ahead in data, text, and media mining. Pages 727-731, ISSN 0167-9236. <https://www.sciencedirect.com/science/article/pii/S0167923611000376>
- [37]B. Shi, X. Bai and C. Yao, "An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 11, pp. 2298-2304, 1 Nov. 2017
- [38]A. Graves, S. Fernandez, F. J. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *ICML*, 2006. 4, 5
- [39]S. M. Tedjojuwono and C. Neonardi, "Aspect Based Sentiment Analysis of Restaurant Online Review Platform in Indonesia with Unsupervised Scraped Corpus in Indonesian Language," 2021 1st International Conference on Computer Science and Artificial Intelligence (ICCSAI), 2021, pp. 213-218
- [40]D. Berrar, "Bayes' Theorem and Naive Bayes Classifier," *Encyclopedia of Bioinformatics and Computational Biology*, vol. 1, pp. 403-412, 2018.
- [41]An J, Lee S, Lee G G. Automatic acquisition of named entity tagged corpus from world wide web[C]//The Companion Volume to the Proceedings of 41st Annual Meeting of the Association for Computational Linguistics. 2003: 165-168.
- [42]"Write smarter, write with Ludwig!" 2022. [Online]. Available: <https://www.youtube.com/watch?v=ZgIHx2aCcPE>
- [43]SpaCy. (2020) training. Retrieved 26 December 2020. [Online]. Available: <https://v2.spacy.io/usage/training>

- [44] Hiroki Nakayama et al. doccano: Text Annotation Tool for Human. Software available from <https://github.com/doccano/doccano>. 2018. url: <https://github.com/doccano/doccano>
- [45] Chou C L, Chang C H, Lin Y H, et al. On the Construction of Web NER Model Training Tool based on Distant Supervision[J]. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 2020, 19(6): 1-28.
- [46] Derczynski, L. «Complementarity, F-score, and NLP Evaluation». In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. 2016, pp. 261–266.
- [47] Powers, D. «Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation». In: *Journal of Machine Learning Technologies*2(1) (2008), pp. 37–63.
- [48] Smith S L, Kindermans P J, Ying C, et al. Don't decay the learning rate, increase the batch size[J]. *arXiv preprint arXiv:1711.00489*, 2017.
- [49] Bompard E, Huang T, Wu Y, et al. Classification and trend analysis of threats origins to the security of power systems[J]. *International Journal of Electrical Power & Energy Systems*, 2013, 50: 50-64.