

## SYMBOLY MAPPING TO UNSTRUCTURED DATA: A DATA SCIENCE PERSPECTIVE

Dr Inigo Jauregi - Data Scientist  
RoZetta Technology

Financial market data, such as Tick Data, is consumed by analysts all around the world. It is “the most granular high-frequency data available” [1], vital for compliance activity, high-frequency/algorithmic trading and market microstructure analysis.

With the increasing accessibility to data, market players such as hedge funds, private equities, individual portfolio managers and more are looking for alternative sources to introduce and provide market advantage and investment opportunities [2]. Unstructured Alternative Data sources are the likes of geolocation data, credit card transactions, corporate jet tracking, shipping trackers, social media posts, product reviews, email receipts, company announcements, news and more. Alternative data can be viewed as any data in any format that can provide insights on the performance of a company or instrument.

Symbology mapping to unstructured data is a process of linking data to relevant tick symbols, in a specific time frame. The result is an enrichment of the tick data that can help to provide further context and understanding. In this article, we describe how this mapping is performed on unstructured textual data (i.e. newsfeeds, social media, company announcements) leveraging advances in natural language processing (NLP) research.

# DATA MAPPING PROCESS

There are four high-level steps involved in mapping tick data symbols to unstructured data:

1. Data Preprocessing
2. Named Entity Recognition (NER)
3. Entity Linking
4. Document weighting

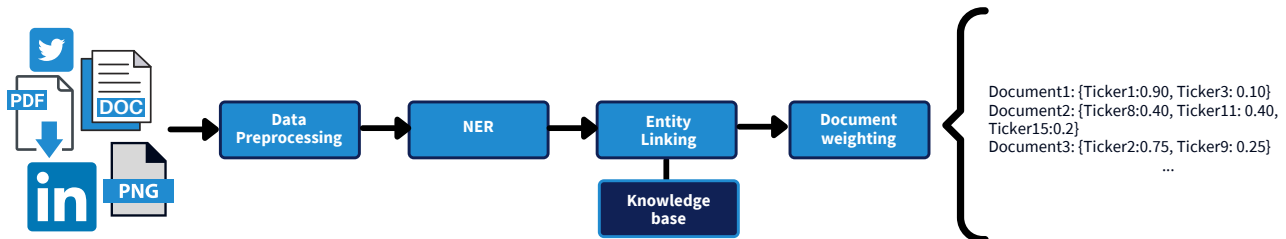


Figure 1: Tick data mapping to unstructured data process

## 1. DATA PROCESSING

As in every NLP task, the first step is to clean or pre-process the data. Initially, the data has to be converted into a format that the models can understand (e.g. txt, JSON, CSV...). This step can vary depending on the type of data we are using. For example, company announcements are generally in PDF format and through an optical character recognition (OCR) or alternative conversion tool the PDF is converted into 'txt' files; instead, if we are viewing websites, tools such as the python package *beautifulsoup* can be used to intelligently extract the important text for analysis; or if using an API to extract data (i.e. Twitter API), there is a need to correctly filter irrelevant information from the response of the API.

Assuming that the data is already in a “machine-readable” format, there are two basic pre-processing steps that almost every NLP system shares: **tokenization** and **vocabulary selection**.

Tokenization involves breaking a string of written text (i.e. documents, paragraphs) into smaller units such as sentences or words. For example, if we apply both sentence and word segmentation to the paragraph in Figure 2,

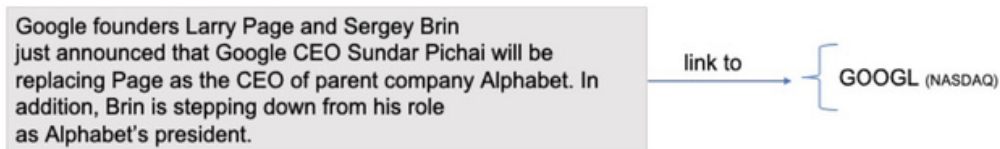


Figure 2: Tick symbol mapping to a news article from Techcrunch

the result is:

```
[
  ['Google', 'founders', 'Larry', 'Page', 'and', 'Sergey', 'Brin', 'just', 'announced', 'that',
  'Google', 'CEO', 'Sundar', 'Pichai', 'will', 'be', 'replacing', 'Page', 'as', 'the', 'CEO', 'of',
  'parent', 'company', 'Alphabet', '.'],
  ['In', 'addition', ',', 'Brin', 'is', 'stepping', 'down', 'from', 'his', 'role',
  'as', 'Alphabet', '\quotes', 'president', '.']
]
```

This converts the string into a list with lists of words. Note that punctuation is also segmented into individual tokens. This allows the NLP models to recognize smaller units of information and facilitates the recognition of patterns. Tokenization, which is a trivial mental process for humans, is not that simple in computers, particularly in some written languages that don't have explicit word boundaries (e.g. Chinese). Luckily, there are many libraries available to perform tokenization on multiple languages such as **NLTK**, **Spacy**, & **Jieba** among others.

Vocabulary selection is the step whereby a dictionary is formed that the subsequent NLP system (i.e. NER, entity linking) will use to learn the task. The problem this presents however is that there are too many words, in any language, for the system to handle due to memory and computing limitations. Therefore, in some sense, vocabulary selection is equivalent to feature selection (as it is understood in other machine learning fields other than NLP), in that words that are not part of the selected vocabulary will be ignored by the model or will be mapped to a common UNK (unknown) token. Usually, the vocabulary of an NLP system is limited to the most frequent words (30,000-100,000) in the training data.

Alternatively, the vocabulary can be formed at the subword level using **Byte Pair Encoding** (BPE) [3] or **sentencepiece** [4]. These algorithms will learn common lexical patterns in a particular language and break words into smaller pieces. The benefit of this approach is two-fold: 1) a reduced, memory efficient vocabulary and 2) the problem of having “unknown” tokens is removed, as any new word in the data can always be formed as a combination of these sub-words. Text tokenized into sub-words would look this way – where the “@@” symbols indicate there has been a word split.

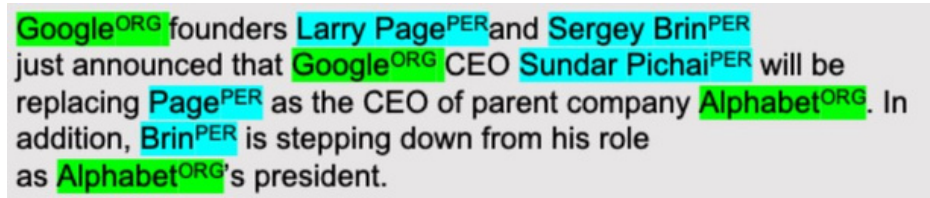
```
[
  ['Google', 'found@@', 'ers', 'Larr@@', 'y', 'Page', 'and', 'Ser@@', 'gey', 'Brin',
  'just', 'announc@@', 'ed', 'that', 'Google', 'CEO', 'Sundar', 'Pichai', 'will', 'be', 'replac@@',
  'ing', 'Page', 'as', 'the', 'CEO', 'of', 'parent', 'com@@', 'pany', 'Alpha@@', 'bet', '.'],
  ['In', 'add@@', 'ition', ',', 'Brin', 'is', 'stepp@@', 'ing', 'down', 'from', 'his', 'role',
  'as', 'Alpha@@', 'bet', '\quotes', 'president', '.']
]
```

Additional pre-processing methods can also be applied to the data such as text normalization, noise removal, stop word removal, word stemming, word lemmatization, and the like. (see [5] for more information). But all these techniques are very task dependent and should only be used if they contribute to improving the accuracy of the models.



## 2. NAMED ENTITY RECOGNITION (NER)

After the data has been preprocessed, the next step is to identify key information (entities) in the text that will help link the document to any relevant tick symbols. Mentions of company names (ORG), **geolocations** (LOC), **person names** (PER) such as CEOs or **product names** (PRO) are some examples of the entities we are looking for.



Google<sup>ORG</sup> founders Larry Page<sup>PER</sup> and Sergey Brin<sup>PER</sup> just announced that Google<sup>ORG</sup> CEO Sundar Pichai<sup>PER</sup> will be replacing Page<sup>PER</sup> as the CEO of parent company Alphabet<sup>ORG</sup>. In addition, Brin<sup>PER</sup> is stepping down from his role as Alphabet<sup>ORG</sup>'s president.

Figure 3: Detected NER in text. (ORG) stands for Organization and (PER) Person

A simple approach to tackle this problem is to maintain a collection or *gazetteer* of common named entities. Each time there is a string match with an entry in the gazetteer the chunk of text is labelled as an entity of a particular class. Yet there are obvious limitations with this approach. First, entities that are not in the gazetteer will not be detected. This will require updating the gazetteer regularly, which is not scalable considering the millions of new company names, acronyms, products, people names... that appear in the data in high frequency. Second, gazetteers do not capture misspellings on the name that potentially a human writer may make when generating the document. And third, often it is difficult to distinguish the ambiguities of a word from simple string matching without more context from the sentence (e.g. *Santiago*, is this referring to a location or a person?, *Apple* is a fruit or the company?).

Consequently, most researchers and companies have moved to using machine learning models based on neural networks such as LSTMs [6] or Transformers [7]. These trained networks can model contextual dependencies between words in a sentence and can detect predefined named entities in text. Due to the fact they “understand” patterns in language, they can identify previously unseen entities, having a much better generalization capacity. See this typical example:

*“ABC is a XYZ based company.”*

From the context, the neural network will know that **ABC** is probably a company and **XYZ** a location. Commonly a BIO tagging scheme (B-Beginning, I-Inside, O-outside) is also introduced to identify the boundaries of the entities (i.e. single token entity vs multi-token entity).

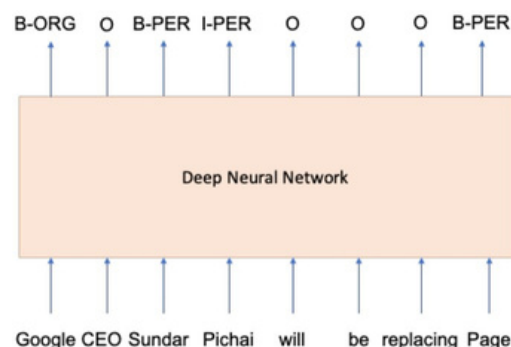


Figure 4: Neural NER model with BIO tagging

The challenge of course with neural networks is that they require **supervised training** in order to learn to accurately identify named entities, which often is a problem for many data scientists, as annotating data is costly and time-consuming. Using pre-trained NER models such as Spacy, Polyglot, Amazon Comprehend or Google API, among others, could be a viable solution, but these models are often trained in a different type of data and their performance may not be good enough to solve the problem.

An alternative solution is to leverage the latest advancements in **transfer learning** for NLP. This involves using large **pre-trained language models** (LM) (massive neural networks with millions of parameters) such as BERT [8], GPT-2 [9], GPT-3 [10] or ELMo [11], that have been trained by others over millions of text records in an unsupervised manner. These large LMs can be then fine-tuned to learn the specific NLP contextual tasks (e.g. NER) with considerably less data annotation required. Pretrained LM's are widely available to download from sites such as **HuggingFace**.

### 3. ENTITY LINKING

After the key information has been located in the document, we can then link the document to specific tick symbols. However, here we face several challenges. What if the NER model identifies the mention of the company name **Alibaba**, is it referring to a particular branch of Alibaba? Should the document be linked to the tick symbol from Hong Kong (HKG) or New York (NYSE) Stock Exchanges? Or both? What if the model identifies a name of a person associated with multiple companies? Should we link the document to all of them or only one of them? Or to none of them? The entity linking model should be able to disambiguate all these scenarios.

In order to do so, first, a common **knowledge base** is required, where all the existing tick symbols and the available metadata (e.g. companies associated, location, people, products) are stored. This may be a hash-table, a relational database, or a graph database. Nowadays graph databases are becoming a preferred choice, as they allow for better structured relational information between entities, powerful recursive path querying and faster data access/update. No matter which type of database is used, it is key to keep them up to date with the most recent information regarding tick symbols and related entities.

Once the database is ready, a simple way to link entity mentions may be **fuzzy string matching**. This is an approximate string matching between a named entity and an entry in the database. Fuzzy string-matching methods measure the distance between two sequences with metrics by calculating the distance between them or mapping them to a common phonetic representation. Some common approaches are Levenshtein distance, Hamming distance, the Jaccard similarity or Metaphone. Applying this matching, we can rank possible candidates from the database, and the entry with the highest matching score above a certain threshold could be linked to the entity. As an example, if we aim to match the entity Google, it will rank highly database entries such as **Google LLC, Google Ventures** or **AdMob Google Inc**, and it will give a low score to organizations that have a completely different name (e.g. **Apple**).

Once the entity has been linked, using the attributes and relations stored in the knowledge base, we can identify which is the primary tick symbol. In this case, from the database, we know that Google LLC's parent company is Alphabet Inc, which is a publicly held company with the tick symbol GOOGL in NASDAQ.

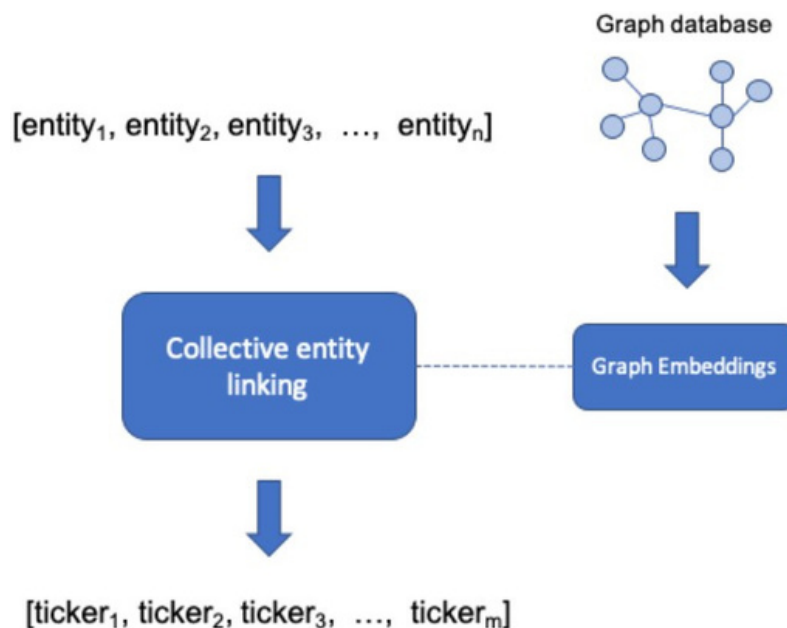
Other alternatives to fuzzy matching could be using indexing approaches based on n-gram matching or machine learning methods, which could be trained to learn how to score the similarity between two strings.

However, all previously mentioned approaches suffer from a similar problem to NER: it is difficult to confidently disambiguate an entity without the context provided in the surrounding words/sentences in the document. Coming back to our example, if we aim to link person entity **Larry Page** in isolation it would be a difficult task, as it is possible there are more people with the same name in the database, which one should we choose? Yet if we observe the co-occurring entities (i.e. **Google, Sergey Brin, Sundar Pichai, Alphabet**) we could assume quite confidently that it is referring to one of the founders of Google. Following this idea, more advanced approaches based on **collective entity linking** have been proposed recently [12]. These models leverage the co-occurring entities to rank better the candidates from the database. Often graph embeddings (learned from a graph database) are used to further improve the disambiguation process.

The result of this step is a list of tick symbols that have been associated to the named entities detected in the document. Note that a document may contain multiple repeated tick symbols linked to it, for example, if a company is mentioned several times.

#### 4. DOCUMENT WEIGHTING

Finally, there is the need to assess the importance of the documents for each linked ticker. In other words, the document will not be equally relevant for all the tick symbols associated.



Here it is possible to use a straightforward approach to weigh the importance of the document, which consists of counting the number of times each ticker is linked to an entity and normalizing those counts into a score between 0 and 1. As a result, the document will be more relevant for tick symbols that have a higher count.

The results should look similar to the output in Figure 1, where each document is linked to some tick symbols with a probability score.

## POTENTIAL FOR THE NEXT STEPS

This article has described an approach to symbology mapping to unstructured textual data. A number of applications will support competitive advantage and greater decision insight such as:

- Delivering the bridge to gain a competitive advantage by leveraging sources such as Alt Data together with financial and company information to enhanced decision making.
- Improving efficiency and insight, through the ability to intelligently navigate and summarize a collection of documents, extracting only key and relevant information.
- Applying sentiment analysis to provide tonal inference in the information extracted.
- Powerful automated extraction of relationships between directors, companies, locations and more.

**To find out more about our powerful data science, cloud technology, SaaS platform or managed service offerings, talk to us today.**



## REFERENCES

- [1] Quantpedia, "Working with high-frequency tick data – cleaning the data [online]." <https://quantpedia.com/working-with-high-frequency-tick-data-cleaning-the-data/>. Accessed: 2020-08-18.
- [2] Deloitte, "Alternative data adoption in investing and finance [online]." <https://www2.deloitte.com/us/en/pages/financial-services/articles/infocus-adopting-alternative-data-investing.html>. Accessed: 2020-08-18.
- [3] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," arXiv preprint arXiv:1508.07909, 2015.
- [4] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," arXiv preprint arXiv:1808.06226, 2018.
- [5] T. D. Science, "Nlp text preprocessing: A practical guide and template [online]." <https://towardsdatascience.com/nlp-text-preprocessing-a-practical-guide-and-template-d80874676e79>. Accessed: 2020-08-18.
- [6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.
- [7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in Advances in neural information processing systems, pp. 5998–6008, 2017.
- [8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.
- [9] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.
- [10] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., "Language models are few-shot learners," arXiv preprint arXiv:2005.14165, 2020.
- [11] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," arXiv preprint arXiv:1802.05365, 2018.
- [12] A. Parravicini, R. Patra, D. B. Bartolini, and M. D. Santambrogio, "Fast and accurate entity linking via graph embedding," in Proceedings of the 2nd Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA), pp. 1–9, 2019.

# UNLOCK THE FUTURE

At **RoZetta Technology** we support businesses derive value and insights from big data and analytics through our data science, cloud technologies, platform and managed services.

We have a strong reputation for building and operating world-class petabyte-scale data and analytic solutions for global financial, utility, energy, telecommunications and government clients. Our experience is unrivaled in dealing with the complexities of ingesting, curating and managing global exchange data.

Our team of data scientists and technologists use a research-driven, systems-design model that has delivered over \$4b in returns to our industry partners. Together with our clients, we continue to develop and refine next-generation technology solutions and information services.

[www.rozettatechnology.com](http://www.rozettatechnology.com)