**How to improve research engines with Deep Learning?**

Nabil FEGAIERE, Atypical Skills CEO & Founder

nabil.fegaiere@gmail.com

1. **Introduction:**

Many companies producing and managing bibliographical databases or documents databases are shifting from legacy systems to new modern open source software. Those software are now called search engines. The most used may be is Elasticsearch. Legacy systems, like Minisis for example, had been used to process structured, semi-structured and unstructured data. They are based on the principle of non-normalized tables (N1NF: Non-First Normal Form). They mainly lack statistical models and scalability. In the other side they use many dictionary and rules for natural language processing. The most famous are stop words dictionary and thesauri use to manage relationship between words.

1. **Search engines context usage:**

One of important differences between legacy systems and the new search engines is the data origins. Legacy is used only to process data produced internal by companies. Search engines are used to process and manage data produced in internal and also from external. The most important use case is processing data coming from social media. Processing data from social media implies the following properties:

* Volume: in the most case we ingest a huge volume of data
* Variety: document to ingest could be word documents, PDF files, Json records etc.
* Terminology: a great number of terms, chosen by document producers and not known in advance.
* Veracity: Origin documents may contain many syntactic errors and many abbreviations that are not previously known by companies.

1. **Search Engines capabilities:**

Most search engines provide the following capabilities:

* Indexing: this is the main capabilities, those systems could index very quickly a great number of documents
* Queries: users could query created indexes, using some terms, in order to retrieve documents containing terms of the query.
* Probabilistic models: when indexing each term are assigned weights using some probabilistic models. The most used are tf/idf and BM25.
* Analysers: Textual data are analysed using natural language processing (NLP). In general, they are limited to morphological analysis and the analyser will generate a list of normalized or un-normalized tokens. Those analysers do not go beyond this level and we do not obtain information about grammatical categories, nor the taggers and part of speech (POS).
* Reformulation: is the capability to declare some dictionaries containing clusters of words having some relationships and could be added or substituting some terms figuring in the query.

1. **The Hard work:**

Which terms have to be used by in queries? There is a great number of documents and do not have an idea about used terms. If the user put some similar terms the system will not be able to retrieve the corresponding documents. So, we will a bad rate of recall. Fortunately, search engines provide the possibility to create and use synonym files.

But the question is: How to create those files?

* Manually: it will be a hard work. We had to list all terms (about Millions or billions) and create clusters
* Automatically: How to do it? Which technology to use?

1. **NLP & Deep Learning:**

Many deep learning models training have been proposed and used in NLP domains. One of the famous and latest is “word embedding”. The first algorithm proposed by google is called “word2vec”. Than Followed by Stanford “glove” and yahoo “fastText”.

1. **Pre-trained models:**

Some pre-trained models could be downloaded from internet sites. It is do download the google, Stanford and yahoo models.

* Google create a model containing 3 million words trained on a corpus of 300 billion words from google news.
* Stanford creates a model containing 400000 words trained on internal corpus.
* Yahoo creates a model trained on Wikipedia documents.

The process of training a model is a long running process. First, we have to clean and normalize the data. Second, we had to train the model on a huge corpus. We had to spend many days just for training the model.

1. **Pre-trained models & synonym files:**

To create synonym files and use them with search engines like Elasticsearch, we designed and developed an algorithm using as input one or more pre-trained models and generate as output a comma separated files in which each line corresponds to a cluster containing synonym words. For this purpose, we use Gensim. It is a python library that could be used to read pre-trained models. There is also a similarity function that, given a word, could return a list of top n most similar words. Each word of this list has a weight indicating the degree of similarity. And the list is sorted in descending order so the first is the most similar word and the last is the least similar word.

We defined to parameters to adjust the efficiency of the algorithm:

* Maximum cluster size (MCS): the number of most similar words returned by the similarity function.
* Similarity level (SL): the minimum of the weight value used to filter similar word to be added to the cluster. In fact, similar words returned by the similarity function of Gensim they will not be considered synonyms if they do not have a weight greater than SL.

1. **Trained models & synonym files:**

Pre-trained models are not suitable for specific domains. To improve our system, we also train models using word embedding algorithms defined previously. But the major issue is about the database size. We must have a large number of documents in our database. The advantage of pre-trained models, that they are trained on a huge volume of documents.

1. **Conclusion:**

We are trying to improve system performance by using more syntactic and semantic information. Word embedding use a convolution with one dimension, we are designing an extension using CNN with multi-dimension convolution using tags, grammatical categories, part of speech (POS), syntactic and lexical relationships between words, named entities (NER). For this we are using spaCy, a very good python library for natural language processing (NLP).

1. **Bibliography:**
2. Hobson L., Cole H., Hannes M.H 2018. Natural Language Processing in Action. Manning Publication
3. Brownlee J. 2018. Deep Learning for Natural Language Processing. Machine Learning Mastery
4. Chollet F. 2018. Deep Learning with Python. Manning Publication
5. Brownlee J. 2018. Deep Learning with Python. Machine Learning Mastery