Deep learning for specific information extraction from unstructured texts

Intuition Engineering Follow
Jul 21, 2018 · 8 min read

This is the first one of the series of technical posts related to our work on iki project, covering some applied cases of Machine Learning and Deep Learning techniques usage for solving various Natural Language Processing and Understanding problems.

In this post we shall tackle the problem of extracting some particular information form an unstructured text. We needed to extract our users' skills from their Curriculam Vitaes (CVs) even if they are written in an arbitrary way such as "was deploying quantitative trading algorithms on production server".

This post has a demo page, check our model's performance on your CV.

Linguistic models

Modern linguistic models (ULMfit, ELMo) use unsupervised learning techniques such as creating RNNs embeddings on large texts corpora to gain some primal "knowledge" of the language structures before a more specific supervised training step. In some cases on the contrary you need a model trained on a very specific and small dataset. These models possess nearly zero knowledge of the general linguistic structures and work only with special text features. A classic example would be a naive sentiment analysis tool for movie reviews or news datasets—the simplest working model could operate only with "good" or "bad" adjectives synonyms and some emphasising words presence. In our research we took advantages of both approaches.

Generally when some corpus of texts is analysed we are looking at the whole vocabulary of each text. Popular methods of texts vectorisation, such as *tfidf, word2vec or GloVe* models are using the whole document's vocabulary to create its vector except of stopwords (such as articles, pronouns, and other quite generic language elements bringing little semantic sense in such a statistical averaging procedure). If there is a more specific task and you have some additional information about the texts corpus, you could probably state that some information is more valuable than the other. For example, to perform some analysis on a corpus of cooking recipes it would be important to extract ingredients or dish names classes from the texts. The other example is extracting professional skills from the CVs' corpus. If we could vectorise each CV by relating it with a vector of extracted skills it would let us perform much more successful industry positions clustering, for example.

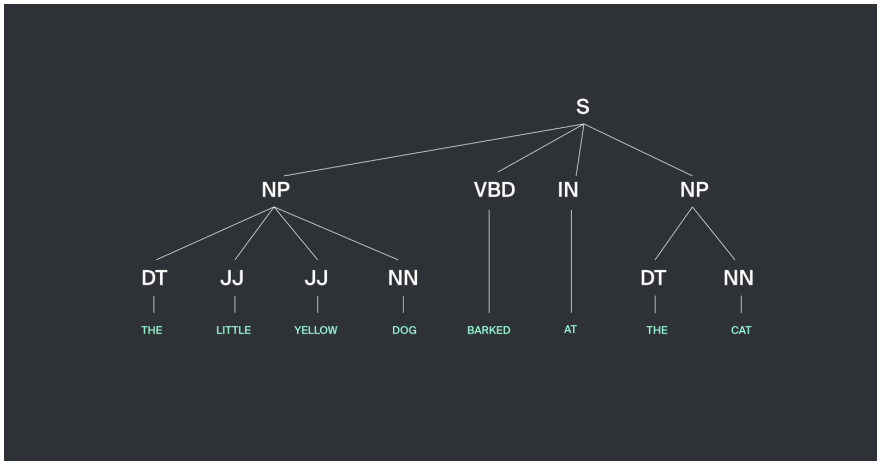
Example:

CV: *Data scientist, hands on expertise in machine learning, big data, development, statistics and analytics. With my team of data scientists implemented Python machine learning model ensembles, stacking, and feature engineering demonstrating high accuracy rates in predictive analytics. Created a recommender system using Doc2Vec words embeddings and neural networks.*

Extracted professional skills: *machine learning, big data, development, statistics, analytics, Python machine learning model ensembles, stacking, feature engineering, predictive analytics, Doc2Vec, words embeddings, neural networks.*

Step 1: Parts of speech tagging

The task of entities extraction is a part of text mining class problems —extracting some structured information from an unstructured text. Let us take a close look at the suggested entities extraction methodology. As far as skills are mainly present in so-called noun phrases the first step in our extraction process would be entity recognition performed by NLTK library built-in methods (checkout Extracting Information from Text, NLTK book, part 7). Part of speech tagging method extracts noun phrases (NP) and builds trees representing relationships between noun phrases and the other parts of the sentence. NLTK library has a number of tools performing such phrase decomposition.



NLTK book, chapter 7, pic 2.2: An example of a simple regular expression based NP Chunker.

We can define a model as a regular expression giving the sentence decomposition (for example, we can define a phrase as a number of adjectives plus a noun) or we can teach a model on a labeled number of texts from NLTK with extracted noun phrases examples in them. This step results in receiving a number of entities among which some are the target skills and some are not—besides skills CV could contain some other entities such as places, persons, objects, organisations, whatever.

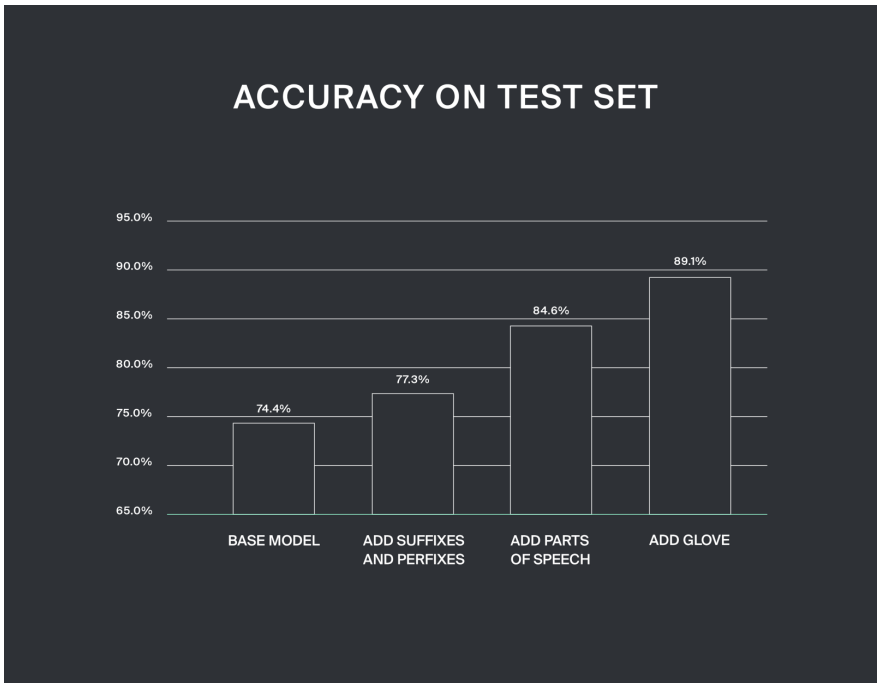
Step 2: Deep learning architecture for candidates classification

The next step is entities classification. Here the objective is quite simple—to tell skills from “not skills”. The set of features used for training is composed regarding the structure of the candidate phrase and the context. Obviously, to train a model we had to create a labeled training set, we did it manually for 1500 extracted entities with skills and “not skills” among them.

> *We have never tried to fit our model to some finite set of hardcoded skills, the core idea behind the model was to learn semantics of skills in English CVs and to use the model for extraction of the unseen skills.*

Each word’s vector is comprised of such binary features as occurrence of numbers or other special characters (skills often contain numbers and symbols: C#, Python3), capitalisation of the first letter or of the whole word (SQL). We also check if a word appears in the English language vocabulary and in some thematic lists such as names, geographical names, etc. The final model using listed features has shown 74.4% correct results on the test set of entities. Usage of another binary feature describing presence of the popular English prefixes and suffixes in a candidate improved model’s performance up to 77.3% correct results on test set. And the addition of one-hot vectors encoding parts of speech to the models’ features set boosted our results to 84.6%.

A reliable semantic words embedding model could not be trained on a CV dataset, it is too small and narrow, to mitigate the problem you should use words embeddings trained on some other, really large dataset. We used GloVe model vectors with 50 dimensions which improved our models’ performance up to 89.1% correct results on a test set. **You can play with the final model in our demo by uploading a text from your CV.**



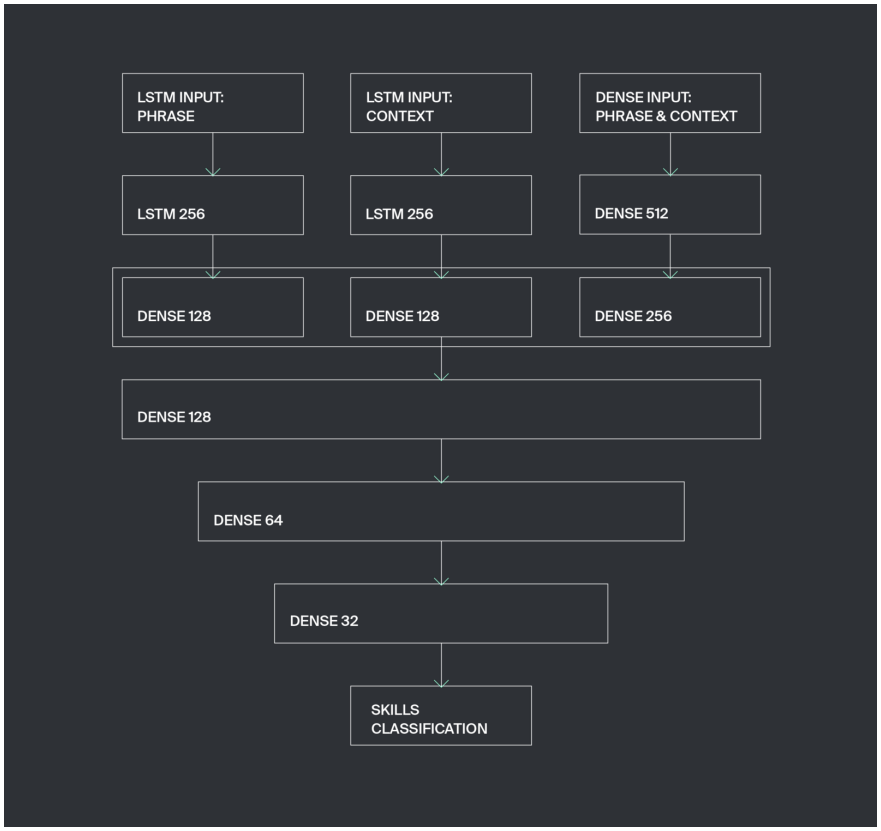
Popular part of speech taggers (NLTK POS tagger, Stanford POS tagger) often make mistakes in the CV's phrases tagging task. The reason is that often a CV text neglects grammar in order to highlight experience and to give it some structure (people start sentences with a predicate, not with a subject, sometimes phrases miss appropriate grammatical structure), a lot of words are specific terms or names. We had to write our own POS tagger solving the aforementioned problems.

The classification is performed with a Keras neural network with three input layers each designed to take special class of data. The first input layer takes a variable length vector comprised of the described above features of the candidate phrases which could have arbitrary number of words. This feature vector is processed with an LSTM layer.

The second variable length vector brings the context structure information. For the given window size n we take n neighbouring words to the right and n words to the left of our candidate phrase, vector representations of these words are concatenated into the variable length vector and passed to the LSTM layer. We found that the optimal n=3.

The third input layer has fixed length and processes the vector with the general information about the candidate phrase and its context—coordinatewise maximum and minimum values of word vectors in the phrase and its context which, among the other information, represent the presence or absence of many binary features in the whole phrase.

We called this architecture SkillsExtractor, here it is.



Skills Extractor network architecture

It's Keras implementation looks the following way:

The best result in model training has been achieved with the Adam optimiser with learning rate decreased to 0.0001. We chose binary_crossentropy as the loss function because the model is designed to classify in two classes.

To make its' usage handy, let us add fit method performing neural net's training and automatic stopping with the usage of cross validation and predict function, forming the prognosis for the feature vector of the candidate phrase.

pad_sequences function converts a list of feature sequences to a 2d array with width equal to the longest sequence in the list. This is made to bring variable length data going to the LSTM layers to the format needed for model training.

onehot_transformfunction converts the target values 0 and 1 to one-hot vectors [1,0] and [0,1]

As long as the numbers of words in an entity and in its context are arbitrary, the usage of a sparse fixed length vector does not look reasonable. Thus the recurrent neural nets processing vectors of the arbitrary lengths here come as a handy and quite natural solution. Our tests proved that the architecture using dense layers to deal with fixed length vectors and LSTM layers for handling varied length vectors is optimal.

Several architectures have been tested with different combinations of dense layers with LSTM ones. The resulting architecture configuration (the size and number of the layers) showed the best results on cross-validation test which corresponds to the optimal usage of training data. Further model tuning could be performed by increasing the training data set size along with scaling the layers size and number appropriately, doing latter with the same dataset would lead to model overfitting.

Results

Examples of extracted skills

All the CVs used for model training were from the IT industry. We were delighted to see that our model also showed quite reasonable performance on the dataset of CVs belonging to other industries such as Design and Finance. Obviously processing CVs with entirely different structure and style would result in a lower model performance. We would also like to mention that our understanding of the “skills” concept could vary from someone elses’. One of the hard cases for our model would be telling skills from new companies names as skills are often equal to software frameworks and sometimes you cannot tell wether this is a startup name mentioned or a new JS framework or a Python library. Nevertheless in most cases our model could be a useful tool for automatic CV analysis and along with some statistical methods could solve wide range of data science tasks on an arbitrary CV corpus.