|  |  |
| --- | --- |
|  |  |

Training occupation classifier

**Introduction**

Online job advertisements scraped from the Internet come from various job portals and enterprises’ websites. Different portals often code professions with different coding systems. Therefore, it is important to use an occupation classifier to deduce for each job advertisement the standard occupation code that is necessary to calculate job vacancies by occupations.

Since last ESSnet work package some classification work on text have been carried out either for occupation or other problems. Many countries considered supervised learning. One challenge was accessing good quality data. The more categories, the more labelled data are required. For example, the international standard classification of occupations (ISCO) defines 436 occupations on the detailed level. To classify on this detailed level, a big amount of training data are needed.

Advertisements’ titles and offers’ descriptions (if available) are often used to classify occupation. Titles usually give a concise description of the jobs which makes them computationally less expensive to use than full description. However, sometimes titles may not contain enough information to build a satisfying classifier.

Thus, the overall challenge is to find optimal algorithms considering available training datasets, which characteristics (such as size and features) depend from one country to another. Apart from a satisfying accuracy and F1-scores, such algorithms must also be not computationally costing and easy to maintain.

**Text classification**

Text classification is well studied in some areas like information retrieval. The task is to construct a classification model (classifier), which catches the features from the training text and their relations to the class label. The model is then used to predict texts with unknown class labels. The label is either explicitly assigned with one class, called hard version; or with probability value, called soft version.

Text encoding transforms the text into a numeric vector before the modelling. First, texts are tokenized, which means divided into units where each unit is called a feature. Bag-of-words is a common language representation. By n-grams one or serval words (letters) in a sequence is transformed into units. 1-gram is to use one word as a unit, 2-grams is to use two words in sequence as a unit and so on.

Feature counts, i.e. counts of the frequencies of a feature in the text, can be used as the feature weight. A piece of text is then represented with a vector of words counts in the text (Zheng and Casari, 2018). For example, the sentence “There are many countries participate in work package B” is transformed by 1-gram feature counts into [1, 1, 1, 1, 1, 1, 1, 1, 1], where each word appears one time in the sentence.

Tf-idf is another common method for weighting the words, which considers both the frequencies of a term in one text and the term frequencies in the whole text corpus (Zheng and Casari, 2018).

With bag-of-words model, a big and sparse matrix is generated from corpus. Feature selection can help efficiently chose the best features and decrease the matrix size for model training. Most of the time, the feature selection is done after stop-words removal and word stemming.

Word embedding is another language model transforming text, which represents words in dense vector space and can learn words relations by consider the text sequences[[1]](#footnote-1).

After text processing and language modelling, a classification model is chosen. Many models for text classification exists, such as decision tree, pattern (or rule) based classifiers, SVM classifiers, Neural Network classifiers and Bayesian classifiers.

Finally, the general process of text classification can be described as data gathering, data exploration, data preparation and feature selection, model training and evaluation (Google developers, 2020). Data gathering works on collecting and accessing to data needed for model training and evaluation. Data exploration focuses on knowing the data, answering questions like average number of words in text corpus and distribution of classes over dataset. Data preparation focuses on text cleaning and tokenization, removing stop words, word stemming/lemmatization and feature weighting and selection, which result in numeric vectors of text. Model training and evaluation are the process of constructing models from training data and choosing the best hyper-parameters and models which are then evaluated.

The following text reports practices of occupation classification from two countries, Dares for France and Statistics Sweden.

**Country Report: Dares (France)**

Since September 2018, Dares collects online job offers from multiple websites on a daily basis. To determine their occupations, Dares developed a Python package (available on ESSnet WPB’s Github) which can predict job offer’ occupation from its title.

Thus, a SVM classifier was trained and tested using more than 1,500,000 labeled offers published on the French National Employment Agency (in French, *Pôle emploi*) in 2019. On test data, F1-score and accuracy of this 532 labels’ classifier reaches 0.92.

Although many improvements are to be made, Dares considers those results good enough to publish statistical outputs such as labour market tightness indicators using automatically labelled offers.

***Introduction to French occupation nomenclature***

Dares needs to classify job offers it collects from the Internet into the French occupation nomenclature, abbreviated ROME (from the French version *Répertoire Opérationnel des Métiers et des Emplois*), in order to produce statistics such as labour market tightness, broken down by occupation.

The French occupation nomenclature is a hierarchical occupation nomenclature in 3 levels as shown below (Tab. 1).

**Table 1: ROME occupation nomenclature insight**

|  |  |  |
| --- | --- | --- |
| **Occupation level** | **Number of classes** | **Code example** |
| **Level 1** *(Grands domaines)* | 14 | **H** (Industry) |
| **Level 2** *(Domaines professionnels)* | 110 | **H12** (Industrial engineering studies and R&D) |
| **Level 3** *(Code ROME)* | 532 | **H1204** (Industrial design) |

***Data gathering***

Our dataset contains a collection of ~1,547,800 job offers published between January 23rd 2019 and December 31st 2019 by employers on the French National Employment Agency website[[2]](#footnote-2). Although collected job offers contains many variables, so far we have used job offers’ titles and occupation codes (code ROME) only to train our model.

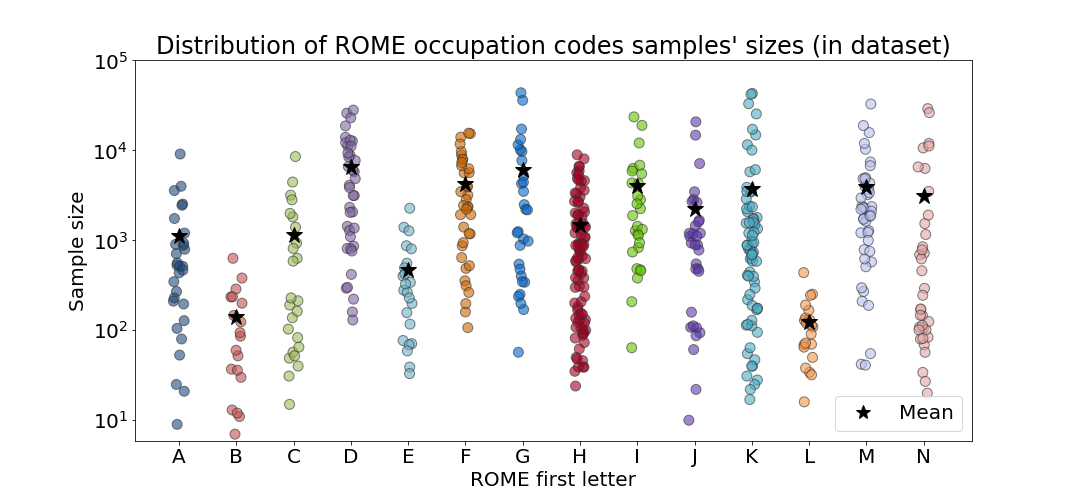
For each occupation code, French NEA provides a list of job titles (total 11,081). Since all occupations were not covered by our dataset, provided examples were added, which led to a dataset of ~1,558,800 examples.

Our dataset is unbalanced: gathered data distribution largely differs from ROME occupation codes’ distribution within the nomenclature (see Fig. 1).

**Figure 1: ROME level 1 distribution in dataset vs nomenclature**

Among each level 1 class, occupation codes are also not equally represented (see Fig. 2). The mean number of examples per code is 2930, the most represented occupation code has 43,767 examples and the less represented one 7.

**Figure 2: ROME occupation codes samples' size distribution**

****

We decided not to re-balance our dataset, since scraped job offers advertisements (i.e. what is to be classified) are likely to be imbalanced in a not so different way form training data – even though we don’t claim our dataset to be representative.

***Text preparation***

Even if data appears to be of good quality (employers seems to check quite carefully job offers’ information before publishing them), some text processing is required to normalize titles and reduce vocabulary size.

Job offer titles go through the following process:

* Texts are standardized (which includes conversion to lowercases, removal of punctuation and numeric characters)
* Stop-words and parasite job related words (such as contract type or working hours) are removed. Location informations such as country, region or district names are also removed. (City names are not removed).
* Titles are lemmatised with Morphalou lemmatizer [1] (only available for French language). We also lemmatize female job nouns to male equivalent (eg. “actress” becomes “actor”), since French language tends to decline all job nouns by gender. Finally, we expand abbreviations and acronyms, using ~150 job related acronyms or abbreviations extracted from gathered data and NEA sources.
* For each title we keep only one occurrence of each word, since repetitions are mostly due to lemmatisation (eg. “Actress/ Actor” becomes “actor actor” which is transformed into “actor”).

***Vocabulary insights***

The vocabulary size decreases at each processing step. From 31,991 unique tokens in raw data, vocabulary is narrowed to 22,019 words after text processing. After processing, empty titles are removed from the dataset and the mean number of tokens per title is 2.7.

More than half words in the vocabulary (54.9%) occur in a unique occupation code in our dataset. Apart from code ROME specificity, it can also be due to misspelled words that are used once in the whole dataset and which are not dealt with for now.

***Selected model and results***

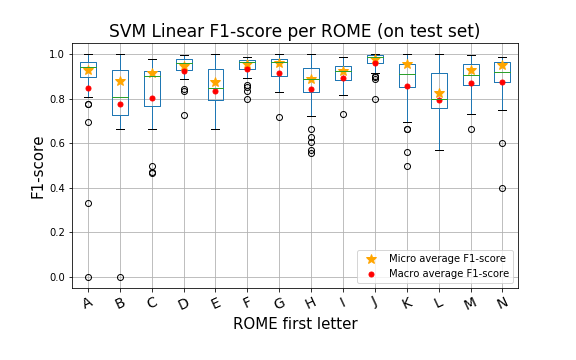
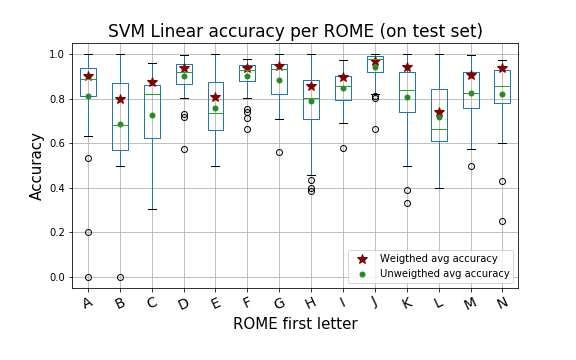
After testing a Text Similarity algorithm which was described in the first ESSNet report [2] and which was not very efficient, we looked at Machine Learning algorithms. As SVM algorithms were providing the best results on a similar problem of classifying job offers from their full text [3], we chose to focus on this algorithm.

Data is split into a train and a test set (80%/20%) and titles are transformed into bag of words vectors. Then, a linear SVM classifier is trained (with hyper-parameter C = 1) in Python, using the well-known Scikit-learn library [4]. This multi-label classifier is trained to classify 532 codes, but we compute scores at different occupation levels (see Tab. 2). At the less precise level (level 1) accuracy and F1-score reach 0.96, which of the most satisfying considering that unweighted F1-score is 0.94. On ROME codes, F1-score is 0.92, but the unweighted F1-score tends to significantly drop (-0.10 from level 1 to level 3).

**Table 2: SVM linear scores on test set**

|  |  |  |  |
| --- | --- | --- | --- |
| **Occupation level** | **Accuracy** | **F1-score** *Weighted average* | **F1-score** *Macro average* |
| **Level 1** | 0,96 | 0,96 | 0,94 |
| **Level 2** | 0,94 | 0,94 | 0,87 |
| **Level 3** | 0,92 | 0,92 | 0,84 |

When focusing on each code (Fig. 3, Fig. 4), it appears that the classifier is less performant on classes lacking samples per code (eg. B, E and L, for which dataset contains less than 500 samples per ROME code on average (see Fig. 2)).

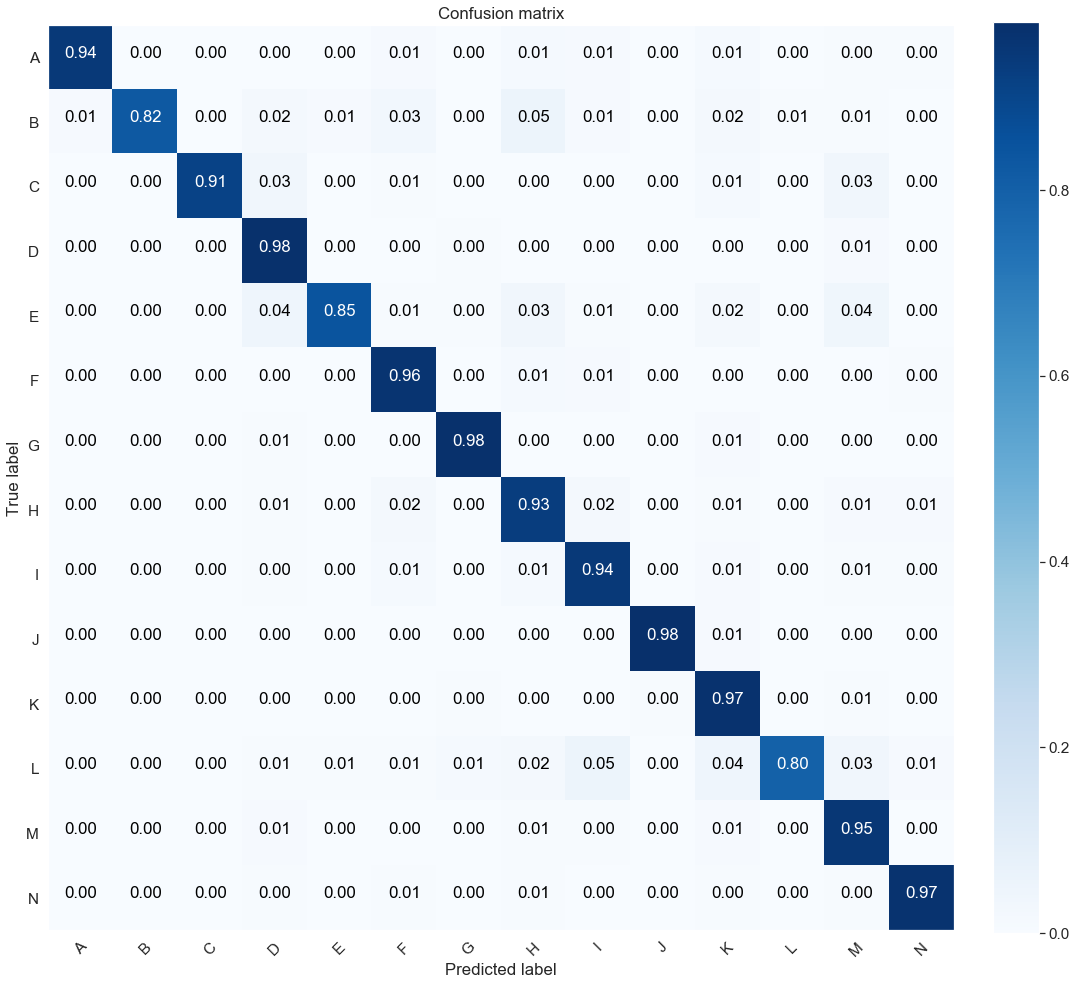
******

**Figure 3: F1-score boxplots**

**Figure 4: Accuracy boxplots**

Confusion matrix (Fig. 5) shows that classes with more examples per occupation code are more wrongly predicted. 41% of misclassified titles (at level 1) are labelled as D, G, F or I (top four mean samples’ sizes classes in dataset) when they contain 23% of nomenclature’s occupation codes.

**Figure 5: Normalized confusion matrix**



***Future work***

Apart from testing other ML algorithms, we are confident that our classifier can be improved. As word order matters in NLP, and its importance may be enhanced in short texts, the first improvement would be to take into account the sequence of words instead of using a flatten bag-of-words representation.

A second improvement could be to take advantage of the hierarchical structure of our nomenclature and train multiple models to classify data “level by level”. Our idea is to train one model to classify the first letter of the occupation code. Then 14 models (one for each letter) would determine the two next digits and finally, last models would predict the full occupation code. The main benefit of such a classification pipeline would be to reduce the number of classes to predict at each step, instead of classifying 500+ classes at once.

Finally, although it is computationally more expensive, using offers’ full description when classification from title is not possible or ambiguous is being studied. It would lead to a more rule-based classification and requires a detailed analysis of results at the finest level (level 3) to precisely understand which categories or features are problematic.

***References***

[1] Analyse et traitement informatique de la langue française - UMR 7118 (ATILF) (2019), “Morphalou [Lexique]”, ORTOLANG (Open Resources and TOols for LANGuage)

[2] Swier N. and al. (2018), “Web scraping / Job vacancies Deliverable 2.2 Final Technical Report (SGA-2)”, Eurostat.

[3] Boselli et al. (2018), “Classifying online job advertisements through machine learning”, Future Generation Computer Systems 86, pp. 319-328.

[4] Pedregosa et al. (2011), “Scikit-learn: Machine Learning in Python”, JMLR 12, pp. 2825-2830.

**Country report: Sweden Statistics**

We classify 46 occupations, i.e. two-digit-level of the standard Swedish professions’ code (SSYK), which is developed from the international standard ISCO by Statistics Sweden. Data come from the advertisements of the Swedish Employment Service, which contains the variable ‘title of advertisement’ and ‘SSYK code’.

Following the instruction of text classification from Google (Google developers, 2020), one-gram model is applied and a linear stack of layers neural network model is constructed. Word embedding has also been tested.

Through choosing different sample sizes and compare the results, we come to an optimal sample size considering the training time and the accuracy requirement. Feature selection is very important for efficient training.

The overall accuracy comes up to 72% with 2000 samples per class, which is an improvement comparing with our previous tests.

***Data gathering***

The data are gathered from Swedish Employment Service (SES), containing the variable “title of advertisement” and “SSYK” assigned by SES. In total, there are more than 4 million advertisements from 2007 to now. The distribution of the 46 classes are not even in the dataset, occupations such as requiring university education in the area of education, health and medicine and occupations for sales are many; some occupations have less than 1000 sample, such as manager in financial and bank institutes and special officer.

Two datasets of 1000 and 2000 samples for each class are drawn randomly from the total advertisements by applying the bootstrapping with replacement procedure, see Table 1.

***Data exploration***

In average, the number of words in each advertisement’s title are three to 4.6 words. The length of words in the titles are between one to more than 20 words. Some titles are ambiguous, not containing enough information for classification. This can been seen in words distribution of the clean text corpus, some titles become empty strings after stop words removing and word stemming.

Table 1 shows the two datasets in the test.

Table 1 Training data set description

|  |  |  |
| --- | --- | --- |
| Text | 2000 samples | 1000 samples |
| Number of samples | 92 000 | 46 000 |
| Number of classes | 46 | 46 |
| Number of samples per class | 2 000 | 1 000 |
| Average words per sample | 4.6 | 3 |

A very small sample has been examined manually by the domain expert to see the matching quality of the occupation label. The expert found examples with not enough and ambiguous information in the sample. Otherwise, the SSYK coding in the dataset is good.

***Data preparation***

We cleaned the text corpus, i.e., first set characters to lowercase, tokenizing text into unit delete illegalcharacters and numbers from the text, and then delete **stop words** that do not contribute to the occupation classification; and last **stemming** the words.

After the cleaning procedure, 16 titles in 1000-sample and 38 in 2000-sample dataset become empty strings. The average number of words in both datasets are 2.8, and the variation is between zero words to maximum 10 and 11 words. The number of words’ distributions in the samples are presented in Figure 1; about 90% of samples contain one to five words in both 1000 samples and 2000 samples.

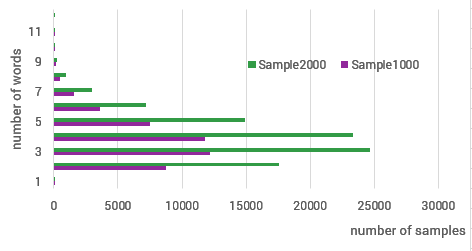


Figure 1 text length distribution

The visualization of the most 100 common words in 1000-sample dataset is shown in Figure 2 below. We choose 1-gram word and most of the words are occupations. The words are mainly occupations that do carry the occupations meaning.

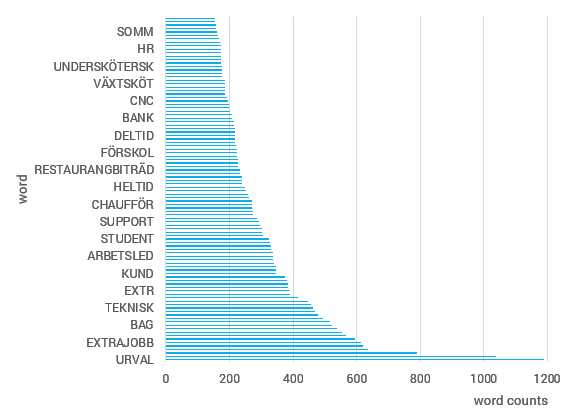


Figure 2 Most important words

***Feature selection***

Feature selection is an automatically way to select the best features for model prediction. According to the rule recommended by Google, ration of “number of samples” and “number of words per sample” when the value is under 1500, n-gram model is preferable considering the accuracy and the computation time. This is our cases, therefore 1-gram model is applied and tf-idf is calculated for representing words. Chi2 is applied to choose the best features from the training text corpus and labels. We examine the scores of the most influencing words in the model; many occupation wordss are included in the most import features, which can be interpreted that Chi2 feature selection is very effective. By feature selection, not only the import words are chosen, the text vectors are in much smaller dimension, 4000 features instead of ~11000 features.

***Build model and tuning the hyper-parameters***

Both datasets are divided into training and evaluation datasets by 67% vs 33%. The neural network model consist of a linear stack of layers. The dimension of the input layer comes from the input data shape, the dimension of the output layer is how many classes there are in the dataset. The intermediate layers are also called hidden layers, of which dimensions of hidden layers are handled automatically. Other parameters specific for multiple classification problem are the activationfunction of the last layer should be “softmax” and the loss function in the model should be “categorical\_crossentropy”.

Figure 3 and 4 show the searching of the best hyper-parameters of layers and features. The two figures show similar structure, layers of two return high accuracy result, the features can be limited to 2500 to 4000. Figure 4 shows that feature 10000 does not improve much comparing with feature 4000. The results show that it is worth to set layers to two and number of features to 4000.

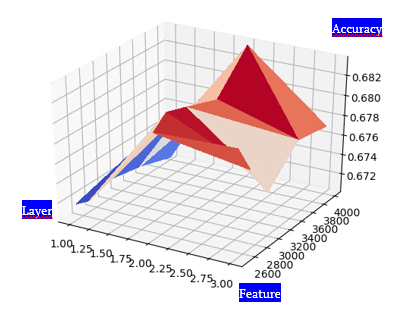


Figure 3 tunning hyper-parameters, 1000 samples per class

Figure 5 shows the accuracy training history of the two datasets with the model contains two layers and 4000 features. The convergence is very fast, after 1 or 2 epochs. By setting epochs to around 10 saves training time. When samples are increased into 2000, the accuracy does improved from 65% to 72%. However, the models have slight overfitting problem, i.e., the training accuracy is higher than the test accuracy.

The accuracy increases not much by epochs, we need to add more words, such as words describing skills and competence. Again, it is a challenge even to access such dataset, that the skills and competence are labelled to occupations’ label.

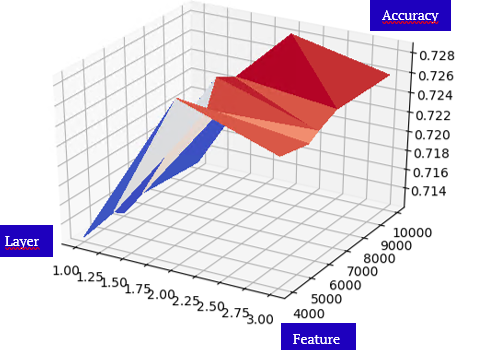


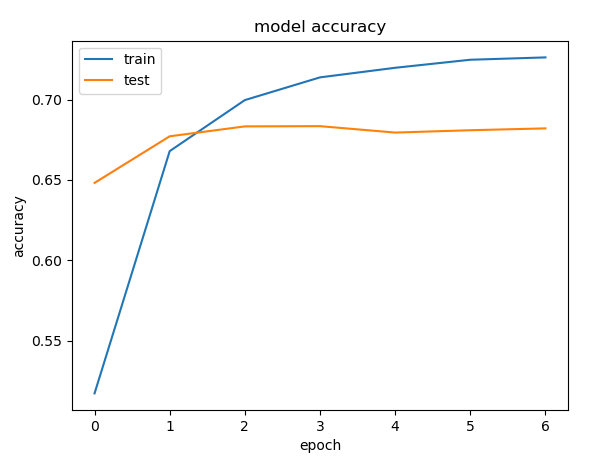
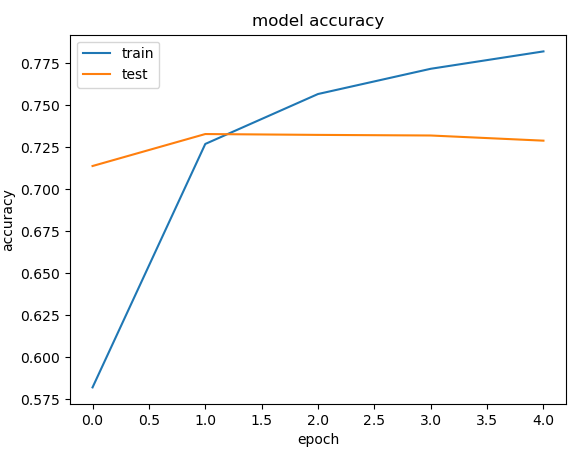
Figure 4 tuning hyper-parameters, 2000 samples pe****r class

Figure 5 Model accuracy in learning process (left: 1000 samples per class; right: 2000 samples per class)

The tests on the two datasets show that the accuracy still has a space for improvement. We increase the samples for each class to 5000. According to the ration between “the number of samples” versus “the number of words per sample”, the model option for the 5000 samples is word embedding. With the word embedding as input, we tested the same 2-layer linear stack neutral network model. We chose 100 dimension as the word embedding, and added a tensorflow embedding layer in the model. The accuracy learning history is shown in figure 6. Through the epochs, the accuracy has been increased, the overall accuracy has been increased to 76%.

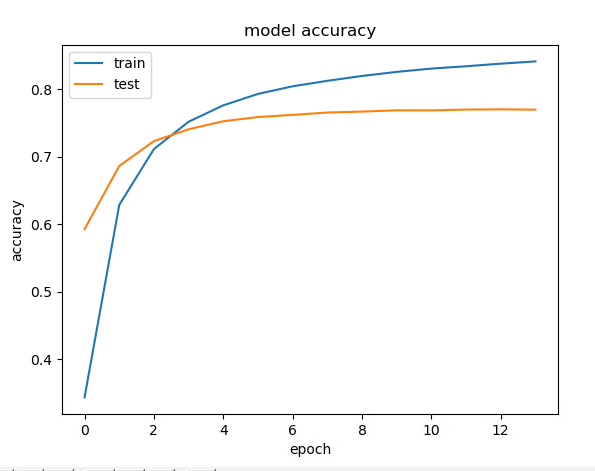


Figure 6 Model accurcy in learing process with word embedding, 5000 samples per class

***Results and future work***

Of the 46 classes, the prediction accuracies are all above 50% except for one class. The correct predictions of every class have the highest scores, shown in figure 6. The overall accuracy of the model has been improved significantly comparing with our previous tests, up to 76% accuracy. Adding chi2 feature selection in the data preparation process, the training becomes more efficiently for the same result. It seems that chi2 feature selection gives the similar effects as the word embedding.

The accuracy has also been improved by increasing the sample size, 1000 samples reaches 65%, 2000 samples reach 72% and 5000 samples come up to 76% accuracy. By considering the trade-off of accuracy, overfitting and efficiency of training, 2000 samples and the 1-gram model should be the best choice. From the data preparation, feature selection, model building and hyper-parameter tuning, the procedure return the accuracy results acceptable.

The classification model needs to predict on the detailed occupation level, for which we need to try on even larger sample size.

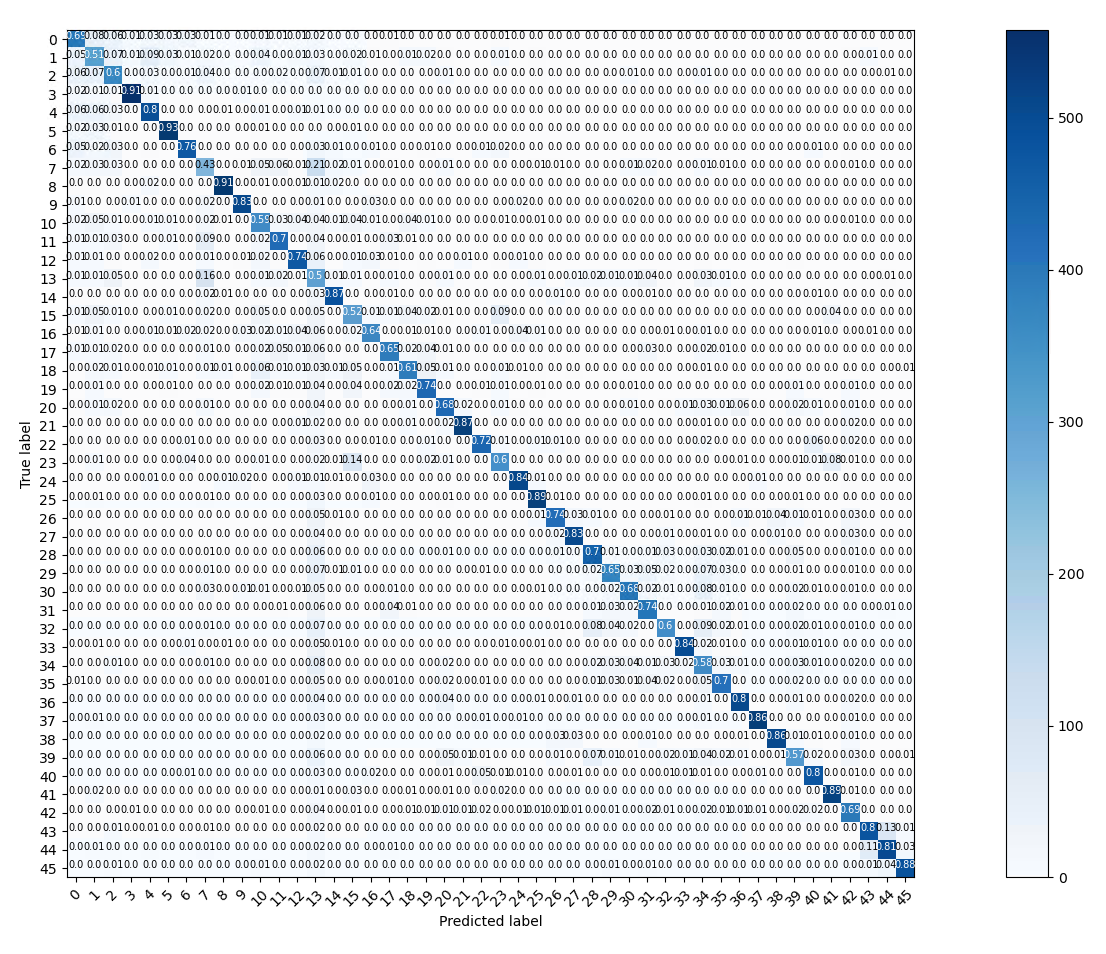


Figure 7 confusion matrix of 46 classes, 2000 samples

Another issue is the metric of accuracy used for model training. There are recall and precision metrics available that we should consider for a better model construction.

***References***

Zheng and Casari, 2018, “Feature Engineering for Machine Learning”, O’Reilly Media, Inc. ISBN: 9781491953242, Released April 2018.

Google developer, 2020, <https://developers.google.com/machine-learning/guides/text-classification>

1. <https://www.tensorflow.org/tutorials/text/word_embeddings?hl=en> [↑](#footnote-ref-1)
2. French NEA, *Pôle emploi*, <https://www.pole-emploi.fr/accueil/> [↑](#footnote-ref-2)