Deep Unordered Composition for Multi-label Classification Applied to Skill Prediction

Yann Duperis[1,2], Adrian-Gabriel Chifu[1], Bernard Espinasse[1], Sébastien Fournier[1] and Arthur Kuehn[2]

[1] *LIS UMR CNRS 7020, Aix-Marseille Université, France*
[2] *MOBEN & ROOSTER, France*

Abstract
Today, many recruitment processes are digitalized. Job offers are posted on job boards and candidates apply by submitting their resumes. To select an appropriate candidate for a job, recruiters rely mostly on the evaluation of the professional skills of the individual. However, researches have shown that individuals tend to omit some skills from their professional profile. A human recruiter, knowledgeable in a given activity sector, is often able to fill the gaps and infer the missing skills. In this paper our aim is to support this human recruiter by automatically inferring theses missing skills, a non-trivial task. To solve this task, first we propose a method to tackle the skill prediction problem by transforming it from a multi-label classification task it to a binary classification task. Then we implement this method with a deep learning model inspired by the Deep Unordered Composition approach. Two different variants of this model, one with the Deep Averaging Network architecture and the other with the Set-Transformer architecture, are evaluated on an open IT resumes data set, and the results are promising.

Keywords
Job recommender systems, Neural networks, Natural language processing

1. Introduction

Nowadays, most recruitment processes are partially digitalized. In our broad research works, we focus on two specific twin phases of the recruitment process: the sourcing and screening steps. The sourcing step consists in searching new candidates for a given job. The screening step consists in filtering the applications for a published job offer. Both steps rely on the evaluation of the adequacy of the pair Candidate/Job. Often, this evaluation is performed manually by a recruiter more or less knowledgeable in the related activity sector. Automated systems, like *job recommender systems*, have arisen in the last decades to assist both phases of the recruitment process, thus allowing the recruiters to focus on activities where they can have a significant added value.

Numerous research works have been, and still are, conducted and published on this matter. A key factor in the success of such endeavor is the appropriate modeling of the professional needs expressed in a job offer, written in natural language, and of the professional abilities of a candidate, also expressed in natural language in digitised documents like resumes. Different approaches exist for such modeling, from the most explicit, with *ontologies* [1], to *Machine*

CIRCLE (Joint Conference of the Information Retrieval Communities in Europe) 2022
yann.duperis@lis-lab.fr (Y. Duperis)

Learning based black boxes [2]. Most of the works presented in this literature are not specialized for a given activity sector or focus on the IT activity sector. Hence, most existing solutions do not yield appropriate results for the activity sector of our industrial partner : the process industry sector. Like other industry sectors, this one exhibits two interesting features:

- A significant specialization of practices, skills, equipment and tools;
- The emergence of a shared sub-language with its own lexicon, concepts and documents structure.

Another constraint expressed by our industrial partner is the need to identify candidates for specialized or expert-level jobs. Hence, our research project aims at producing a candidate recommendation system for expert-level jobs in specific activity sectors. Since human recruiters and most automated systems rely on an individual's skills to evaluate their adequacy for a given job, our own recommendation system should do the same. The novelty of our research project is to create a recommendation system working with specialized and meaningful skills for a given activity sector (i.e. not skills like *Team spirit*, *Management*[1] or *Microsoft Office*).

Researches have shown that these skills are not always declared by candidates the way an automated system (or a recruiter) expects them or even not declared at all [3]. Hence, some pre-processing of candidates professional profiles is required to detect these skills. Although some approaches rely on text pre-processing strategies to extract skills and normalize them to a canonical form [4], some approaches use the information contained in a profile to predict the missing or misspelled skills. This task is referred to as *skill prediction* or *skills inference* and can be assimilated to a *multi-label classification* task.

We hypothesize that the text written in a professional profile is a good predictor of the skills of an individual. Related work presented in section 2.2 introduces recent proposals of methods to build job recommender systems based entirely or partly on the processing of natural language documents - i.e. professional profiles and job offers. Most of the proposed approaches rely on *syntactic composition functions* implemented in neural networks architectures like *Convolution Neural Network*, *Long Short Term Memory* or *Transformer* based architectures [5]. However, models based on *unordered composition functions*, like the *Deep Averaging Network* (*DAN* [6]), have demonstrated the ability to rival syntactic approaches on some natural language processing tasks.

In this paper, we present a method to predict the skills of an individual from their professional profile (resume or professional networking platform profile). Our contributions are:

- A method to tackle the skill prediction problem by transforming it from a *multi-label classification* task it to a *binary classification* task. The core idea is that the model predicts whether a given couple $(profile, skill)$ is likely to be realistic. Unlike the *One versus Rest* approach to multi-label classification, the proposed method does not require the training of one model per skill but requires one inference for every skill prediction;
- A deep learning modular architecture that performs this binary classification. The module responsible for the production of profile text hidden representation is inspired by the

[1]Though the ability to manage a team or a process is a real skill, we see two problems in its usage for recommendation: as a soft skill it can hardly be measured and a statistically significant amount of profiles declare possessing it.

Deep Unordered Composition approach [6]. We evaluated two different implementations compatible with this approach: DAN [6] and the Set-Transformer [7].

This paper have the following structure. In Section 2 we present related works concerning skill prediction and job/profile recommendation. In Section 3 we introduce a formal definition of the skill prediction problem, by formally defining profile and skill prediction. In Section 4 we present our method to tackle the skill prediction problem by transforming it from a multi-label classification task to a binary classification task, and two different deep learning-based implementations of this method, with the DAN architecture and with the Set-Transformer architecture. Section 5 is dedicated to experiments of our method on an open IT resumes data set, which results are discussed in section 6. Finally, we conclude in Section 7 by presenting some perspectives of this research.

2. Related work

Although most recruitment and professional activity related information retrieval systems make use of skills, only a few research works focused specifically on the skill prediction (or inference) problem, we present them in section 2.1. However, the literature on jobs/profiles recommender systems provide valuable insights on the usage of professional profiles data for information retrieval purposes and the section 2.2 will introduce the most recent methods using deep learning to extract useful features from profiles text.

2.1. Skill prediction

To predict the skills of an individual, it is first required to define the concept of skill. Different domains have different definitions. Information retrieval approaches usually rely on an extensive paradigm where skills are modeled as elements of *ad hoc* sets. In this paradigm, a skill is a property of an individual (either binary - the individual possesses the skill - or associated to a proficiency score) useful to model their professional abilities. Here, the set of supported skills is often called a dictionary and is built with data mining approaches [3, 8]. Other approaches to skill modeling integrates more knowledge through the usage of ontologies [1, 9] (with subsumption relationships between skills). Even these approaches still consider skills like labels associated with a profile. There is no intensive definition nor rich semantic relationships between skills or other domain concepts, although relations are sometimes defined between jobs and commonly required skills (like in the ESCO ontology [9]). Such knowledge has been implicitly encoded in a latent space - along with job titles - with graph embedding techniques in [10].

In [3], the authors present the construction of the "Skills and Expertise" *falksonomy* in use at LinkedIn.com. After the introduction of this section in the profiles, the authors noticed a lack of declarations and developed a skills recommendation system to help users fill this new section. The best results were obtained by a *Naive Bayes classifier* using the following features: industry sector, company, function, title and group.

The authors of [11] propose a joint prediction factor graph to predict the skills of individuals in a professional social network by using relationships like: same company, same university or same job title. This approach was evaluated on the 10 most frequent skills, we cannot evaluate its performances in our case.

In [12], the authors propose an explainable *Convolution Neural Network* (*CNN*) model for *high-level skills* prediction in the IT domain. The introduced concept of high-level skill is a composition of low-level skills. Based on the provided examples - *css*, *html*, *php*, etc. for low-level skills and *front-end developer*, *web developer*, etc. for high level skills - we interpret high-level skills as job titles. Although the tackled problem is different, the presented work shows that interesting professional information can be inferred by a deep learning model operating on a profile text.

2.2. Job/Profile recommendation

In [13], the authors propose a co-teaching neural network consisting in a relation matching component inspired by the *Relational Graph Convolution Network* (*R-GCN* [14]) approach and a text matching component relying on a pre-trained *BERT* model [15] for sentence encoding. The document encoding is produced with an hierarchical architecture similar to the *HIBERT* model [16]. The relation matching component also relies on shared keywords with a high *TF-IDF* score as a relation between two objects.

[17] also relies on an hierarchical architecture to build documents representations. The authors use a *Bidirectional Long Short Term Memory* network (*BiLSTM*) model to contextually encode the words in the profile and job sentences. Different attention layers applied either to the profile or to the job extracts latent representations of the offered and requested abilities.

CNN can also be used to extract representations from the texts of profiles and job offers. In [18], the authors propose a deep siamese network based on CNN to maximize the similarity between related profiles and job offers, thus allowing the twin models to learn how to build representations of the text contained in these documents.

The related work presented in this section demonstrates the ability of deep learning models to capture meaningful information in the text of a professional profile. The power of *Transformer* based architectures [5] (like BERT [15]) encouraged us to experiment an unordered version of it, the *Set-Transformer* architecture [7] for one of our model's implementation.

3. Formal definition of skill prediction problem

In this section we define the main concepts used throughout this paper. Section 3.1 introduces the concept of *Skill*, section 3.2 introduces the concept of *Profile* and finally section 3.3 defines the skill definition problem tackled in this paper.

Throughout this paper, the following notations are used:

Symbol	Definition
E	a set of elements $\{e_1, e_2, ..., e_{\|E\|}\}$
$\|E\|$	the cardinality of the set E
e	an element of E
E_k	a subset of E related to a specific element k (element of another set K) with $E_k \subset E$
N^a, θ or θ^b	a constant number
$\mathbf{v}$	a vector
$\mathbf{T}$	a tensor with $rank(\mathbf{T}) > 1$

3.1. Skill definition

We define a *Skill* as a property of an individual that allows the prediction of their ability to perform specific tasks. The understanding of what is a skill, how it is developed and how it manifests itself during the task execution is out of the scope of our scientific work. We consider it in the information retrieval domain. Thus, we can define a skill as a binary property of an individual - the person possesses it or not. With this definition, a skill can be assimilated to a tag attached to a professional profile.

Contrary to a simple *keyword* that might be a mere statistical artifact, a skill is usually meant to refer to an implicit concept, which extension can vary. Indeed, when a recruiter writes a job offer or when candidates write their resumes, the skills are often written, in specific parts of the documents, as short *surface forms* referring to implicit concepts (especially for abstract skills like *Management*, *Machine Learning* or *Information security*). Thus, skills can't be considered like mere keywords and the canonization of the myriad surface forms to a normalized form associated to the concept is far from trivial [4].

We define S the set of all skills ($s \in S$) supported by our information system.

3.2. Profile definition

We define a (professional) *Profile* as a digitized document describing the professional life of an individual. Such documents are not standardized and vary from a multi-page, mostly textual, PDF resume to a *de facto* bag of keywords on professional networking platforms like LinkedIn.com. Even with different formats and contents, most of these documents aim at fulfilling a common function: advertising the capacity of an individual to perform adequately some economic activity[2].

Most profiles contains information to advertise the individual capacities: i.e. history of past professional experiences, self-assessed skills, education and associated degrees, etc. As we have defined in 3.1, the skills declared in a profile are surface forms of concepts embedded in wide implicit knowledge graphs. Other information contained in a profile are mostly semi-structured text written in natural language.

Our approach focuses on the lexical properties of a profile (the domain-specific terms mentioned), thus we simplify the profile definition by concatenating the professional experiences

[2]Most research works focus on jobs but some aim at modeling economic activity at a higher granularity, like tasks allocation in an organization.

and education declared as a *bag of words* (BOW):

$$P = \{(W_p, S_p)\}\ \forall W_p \subset V,\ \forall S_p \subset S, \tag{1}$$

with W_p the most significant terms of a profile, V a domain vocabulary and $S_p \subset S$ the declared skills.

3.3. Skill prediction definition

The given definition of a profile considers that all information contained in it is declared by the individual which professional abilities are described. Researches have shown that the resulting profile is often incomplete [3].

In this paper we focus on the prediction of an individual's skills using the text contained in their profile. More specifically, we model the text as a *bag of words*. Thus, for a profile p with unknown skills S_p and a set of significant domain terms W_p, our research hypothesis is that the skill prediction problem can be simplified by estimating $P(s \mid W_p)$ with a function $f_{sp}(W_p, s)$.

A skill s is considered a predicted skill of the profile p if $f_{sp}(Wp, s) \geqslant \theta^{sp}$, with θ^{sp} the skill prediction confidence threshold. The aim of the work presented in this paper is to build a deep learning model able to estimate the function f_{sp}.

4. A method to predict skills with deep learning

In this section, we present our method for skill prediction with deep learning. This method relies on our two main contributions. In 4.1, we introduce the first one - i.e the transformation of the initial multi-label classification task to a binary classification task by using the skill to predict as an input. In 4.2 we present the second one - i.e. the deep learning model performing the binary classification with a deep unordered composition approach for the handling of profile text. Finally, Section 4.3 will present how the model is trained.

4.1. From a multi-label classification task to a binary classification task

Predicting all possible skills that an individual might possess, from their textual professional profile, is a multi-label classification task. The number of classes is the number of entries in the skills dictionary. A granular skills taxonomy would then produce a significant amount of classes (possibly thousands). As an example, using the European Skills/Competencies and Occupations ontology (ESCO [9]) as a skills dictionary would require the model to classify examples amongst 13.5K classes[3], most of which are entangled in semantic relationships. In such a scenario, a training example presented to the model would be a profile as input with its declared skills as outputs. Such an output vector is high-dimensional and sparse.

To address the shortcomings of such a multi-label classification task, we shifted to a binary classification task with a few adjustments. In this task, the model is trained to predict the probability that a given profile possesses a given skill. This choice is motivated by the following research hypothesis:

[3]https://ec.europa.eu/esco/portal/skill

Using the skill which ownership is to be inferred as an input allows the model to project it in a latent space, hence allowing it to learn relationships amongst skills and with profiles hidden representation.

A training example presented to the model is composed of a profile and a skill as inputs and a binary label as expected output. The meaning of this label is **0: the individual which profile is provided as input to the model does not possess the skill | 1: the individual possesses the skill**.

Formally, let E the set of all examples and e an example modeled as a tuple ($e \in E$):

$$E = \{(p, s, l_{s_p})\}, \tag{2}$$

with $l_{s_p} \in \{0, 1\}$ the classification label. Positive examples are generated from profiles by using declared skills, S_p. The set of positive examples derived from a profile p, E_p^+, can be defined as:

$$\begin{aligned} E_p^+ &= \{(p, s, 1)\},\ \forall s \in S_p \\ |E_p^+| &= |S_p| \\ E^+ &= \bigcup_{p \in P} E_p^+ \end{aligned} \tag{3}$$

Due to the nature of our data set, i.e. a collection of professional profiles with a set of declared skills, we can only extract positive examples. To overcome this limitation, we perform a random negative sampling step during training. We introduce a training hyper-parameter n_s and, for each positive example, we generate n_s negative examples. The skills used to generate these negative examples are randomly sampled uniformly from the skills not declared in the profile.

If we take $n_s = 1$, the examples set, $E = E^+ \cup E^-$, is balanced for the binary classification task. However, the binary classification task is a proxy for the multi-label classification task. The balance of the data set, in terms of classes (i.e. skills) population, is not guaranteed and can have an impact on the performances of the method.

4.2. Deep unordered composition based model

Inspired by the good performances and lower complexity of deep unordered composition based models on some natural language processing tasks [6] - compared to syntactical ones - we decided to model profile text as a bag of words. Our decision is also motivated by an hypothesis:

Specific terms in a profile are used by recruiters to infer implicit skills. Thus, a deep learning model should be able to do the same.

Our proposed model takes two inputs: the profile text (as a BOW representation) and the skill which ownership by the profile must be inferred. Both these inputs are then embedded by using pre-trained embeddings. Each of these inputs is then projected in a latent space by two parallel modules: respectively the *Profile Encoder* and the *Skill Encoder*. The hidden representations are then merged by a *Combination Module*. The resulting aggregated representation is further processed by the *Combination Encoder* before the final binary classification layer (single neuron with a *sigmoid* activation function). The *Combination Encoder* can be omitted and its impact is evaluated by ablation studies in our experiments. The general architecture of the model is depicted in Figure 1.

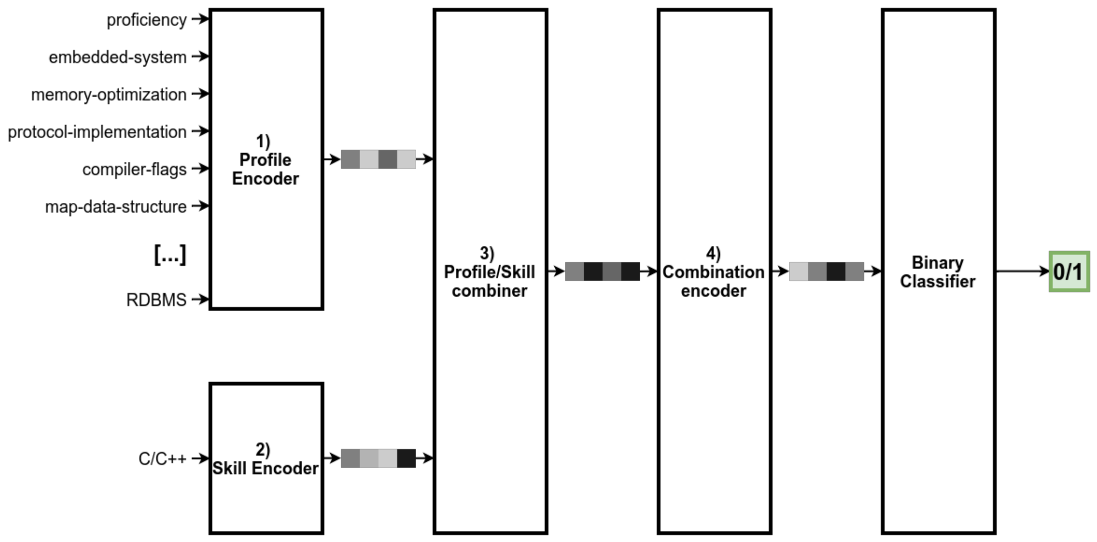

Figure 1: General architecture of the proposed model

Some of these modules have multiple implementations, each with their own variants and hyper-parameters.

4.2.1. Profile encoder

The profile encoder is the module responsible for the encoding of the BOW profile into a single aggregated hidden representation. Figure 2 illustrates this module architecture.

Formally:

$$\boldsymbol{h_p} = PE(W_p,\ T_p), \tag{4}$$

with $\boldsymbol{h_p}$ the profile hidden representation, PE the profile encoder function, W_p the profile terms with the highest scores and T_p the profile terms scores (scored with TF-IDF).

We experimented two integration strategies for the scores:

- **Weighting** - multiplying each term vector by its score and normalizing the vectors before passing them through the next layers;
- **Ignoring** - ignoring the score and only use the terms vectors for profile encoding.

The profile encoder is comprised of multiple layers (see Figure 2):

- $Embedding$ - an embedding layer (trainable or not);
- $Scoring$ - a layer responsible for the aggregation of terms embeddings and scores (it can simply return the terms vectors if the score integration strategy is **Ignoring**);

- TWE - a term-wise encoder (possibly comprised of stacked layers) that further encodes each term in the profile BOW. This layer has two implementations based on deep unordered composition functions and is presented in the next paragraph;
- $Pooling$ - a pooling layer that aggregates the terms latent representations into a single vector;
- $PoolingEnc$ - an optional pooling-encoder (comprised of stacked dense layers) that further projects the pooling to produce a latent representation of the profile [6]. Our experiments evaluates the utility of such a layer for our task.

Since profile text is encoded as a bag of words - hence a **set** of terms - we restricted our choice of layer architectures for the term-wise encoder implementation to the one featuring permutation invariance [7] - i.e. element of an input sequence can be swapped without consequences on the function output. We experimented the following architectures for the implementation of the term-wise encoder layer:

- **Deep Pooling** - Stacked row-wise dense layers that encode each term embedding independently[4];
- **Set Transformer** - Stacked multi-head self-attention blocks without positional embeddings for permutation invariance (implementing the Set Transformer architecture proposed in [7]). We have also experimented the *ISAB* block proposed by the authors to reduce the computational complexity from $o(n^2)$ to $o(nm)$ with m the number of trainable induction vectors used in each *ISAB* block.

To produce a single vector aggregating all the information of the profile, the profile encoder module relies on the usage of a pooling strategy (implemented in the pooling layer). We've experimented the following ones:

- **Average pooling**;
- **Maximum pooling**;
- **Sum pooling with normalization**;
- **Pooling by multi-head attention** - this pooling strategy relies on a multi-head attention block with a trainable vector used as query and the profile terms hidden representations as keys and values. This pooling mechanism has been proposed in [7].

4.2.2. Skill encoder

The proposed method relies on the usage of skills as input to allow the model to learn how to correlate profiles hidden representations with skills hidden representations. The skill encoder module is responsible for the projection of the input skill in a latent space. This module implementation is composed of an embedding layer (trainable or not) and of optional stacked fully connected layers with dropout. The further projection of the skill embedding by stacked layers is commanded by hyper-parameters and is also evaluated by an ablation study in our experiments.

[4]Implemented with the Keras library *TimeDistributed* layer.

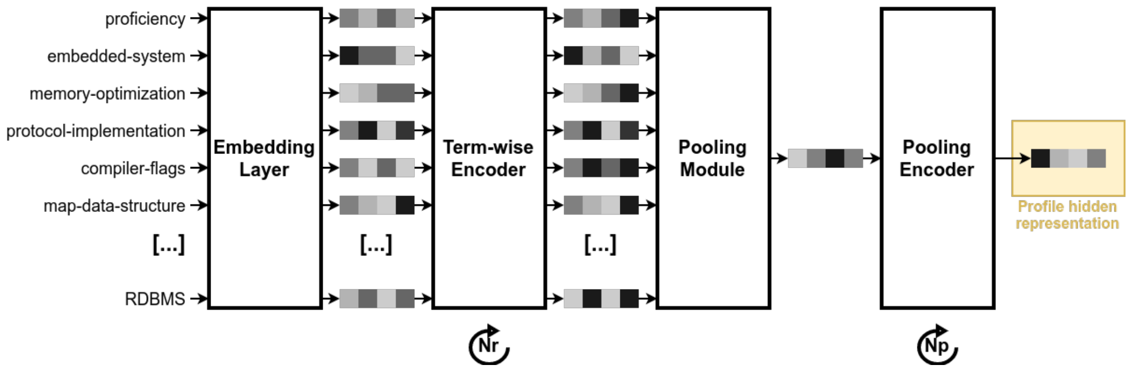

Figure 2: Architecture of the profile encoder module (the Term-wise Encoder can be comprised of N_r stacked layers and the Pooling Encoder can be comprised of N_p stacked layers, N_p can be zero)

4.2.3. Profile/skills Combination module

To evaluate the interaction between a profile and a skill, our model needs a module to perform a combination of both hidden representations. We have only experimented the following strategy:

- **Concatenation** - simple concatenation of the features of both hidden representations into a single vector. This strategy preserves all available information and lets the following layers of the model handle them accordingly to minimize the classification error.

4.2.4. Profile/skills Combination Encoder

The last module of the model projects the profile/skill combination into a latent space that eases the final classification. This module is implemented with stacked fully connected layers with dropout and is optional. Its impact on the model's performances is evaluated by ablation studies in our experiments.

4.3. Model training

4.3.1. Random inputs masking

The variety of profiles can be important and a data set should be huge to cover the lexical diversity of real profiles. Such a diversity can hardly be obtained with a regular data set. So we emulate it by randomly masking some terms in the profile bag of words. This pre-processing step is controlled by an hyper-parameter r_m, the masking ratio. Each profile bag of words used during training undergoes a masking operation when sampled to reduce overfitting. Masked terms are ignored by the profile encoder.

4.3.2. Loss function

We use the *binary cross-entropy* loss to optimize the model, since it is a common choice for binary classification tasks.

4.3.3. Optimizer

Based on several experimental runs, the best optimizer was *Nadam* [19] with a learning rate of **0.01** and exponential decays of **0.9** and **0.999**.

5. Experiments

In this section we introduce the experimental work undertaken to evaluate the proposed method. The conducted experiments aim at answering the following research questions:

- RQ-1: can a Deep Unordered Composition based model capture meaningful information for skill prediction from a professional profile modeled as a set of significant domain terms?
- RQ-2: can domain-specific rare skills be predicted with a machine learning approach?
- RQ-3: is the usage of classes as model inputs in a binary classification task a good proxy for a multi-label classification task?

The section 5.1 presents the selected data set to perform the evaluation, the section 5.2 defines the adequate evaluation metrics for the task at hand and the section 5.3 describes the evaluated models. Finally, section 5.4 shows the results of some of the evaluated models for our task. To the best of our knowledge, no other works proposed a skill prediction method that could be used as a baseline for fair comparison with our method.

5.1. Data set used

We chose to evaluate our approach on a public data set of professional profiles to allow reproducibility and fair comparison of future works. Such a data set is not easy to find, only a few economic actors can collect such data.

We decided to use the data set published in [12] for our work to promote open research. This data set is comprised of **30K** professional profiles in the IT domain[5]. Since the work published in [12] tackled the task of high-level skill prediction - i.e. skills families, job titles or job categories - we had to convert the data set to a format compatible with our own task[6]. This pre-processing step is not perfect and has impacts on the performances discussed in Section 6.

5.2. Evaluation metrics

Although the proposed method transformed the initial multi-label classification task to a binary classification task during model inference and training, we must evaluate it as the former. Hence, we compute the number of false negatives FN, true negatives TN, false positives FP and true positives TP for each class. With these metrics, we can compute Recall, Precision, Specificity, Accuracy and F1-Score.

[5]Although our domain of interest is the process industry sector, this data set allows us to validate the adequacy of our approach for specialized domains.

[6]This adaptation of the data set is available here: https://github.com/yannduperis/circle-2022-it-profiles-dataset

To ease the comparison between different models, we can also compute micro, macro and weighted aggregation of these metrics across all classes (i.e. skills).

For a metric M, we can compute these aggregations with the following formulas:

$$M_{macro} = \sum_{s}^{S} M_s \cdot \frac{1}{|S|}$$
$$M_{weighted} = \sum_{s}^{S} M_s \cdot \frac{|E_s|}{|E|}, \quad (5)$$

with S the set of skills, E the set of all evaluation examples and E_s the set of evaluation examples with the skill s. For the micro version of these metrics, we compute them across all classes.

5.3. Evaluated models

This section presents the evaluated models. The following models are named based on the term-wise encoder used in the profile encoder. Due to the small size of our data set, the evaluated models are small and regularized (with dropout on both attention layers and fully connected layers). For all our models, we fine-tuned the pre-trained skills embeddings, used a profile BOW maximum length of 200 and a masking ratio (r_m) of 30%. We selected these hyper-parameters with prior experiments.

A previous hyper-parameters exploration step guided us to the best embeddings (GloVe 20 dimensions for terms and Word2Vec 10 dimensions for skills), BOW maximum length and masking ratio. We also noticed during this early experiment that taking $n_s > 1$ caused a dramatic drop of the recall.

Except for the DAN model, all models use the *pooling by multi-head attention* strategy.

ST - Set-Tranformer - These models use the Set-Transformer implementation of the term-wise encoder with 4 attention heads (except for ST-III which has 5 because of the terms embeddings number of dimensions), the ISAB version of the multi-head self-attention block [7] with $m = 25$ and 1 layer. For all models, except ST-VII, the evaluated terms scores integration strategy is *ignoring*.

- ST-I: simple model without profile pooling encoder, skill embedding encoder nor combination encoder;
- ST-II: ST-I with trainable terms embedding layer;
- ST-III: ST-I with pre-trained FastText 50 dimensions embeddings for terms (not trainable);
- ST-IV: ST-I with profile pooling encoder;
- ST-V: ST-IV with profile/skill combination encoder;
- ST-VI: ST-V with skill embedding encoder;
- ST-VII: ST-I with the *weighting* terms scores integration strategy.

DP - Deep-Pooling - These models use the Deep-Pooling implementation of the term-wise encoder with 1 fully connected layer of 20 units. For all models, except DP-IV, the evaluated terms scores integration strategy is *ignoring*.

- DP-I: simple model without profile pooling encoder, skill embedding encoder nor combination encoder ;
- DP-II: DP-I with trainable terms embedding layer;
- DP-III: DP-I with a second fully connected layer of 20 units in the term-wise encoder;
- DP-IV: DP-I with the *weighting* terms scores integration strategy.

DAN - This model relies on an implementation similar to DAN [6] as profile encoder : Deep-Pooling term-wise encoder (1 layer with 20 units), *average* pooling strategy, pooling encoder (1 layer with 20 units) and no skill embedding encoder nor combination encoder.

5.4. Results

We trained the previously introduced models on the training split of our data set (**310K** profile/skill positive examples and the same number of randomly generated negative examples, see section 4.1 for details on the negative sampling procedure) and then evaluated them on the test split of our data set (**44K** positive examples and the same number of randomly generated negative examples, with a fixed random seed for all evaluations).

Table 1 displays the evaluation metrics introduced in section 5.2 for each model. The Figure 3 depicts the distribution of the aforementioned metrics across all classes. We can see that the aggregated metrics of all models are close.

We noticed that a lot of classes had a Precision, Recall and F1-Score of 0. After further analysis of the detailed performances of the model for these classes, we noticed that the model was systematically predicting either 1 or 0 for some of them. We name these classes *Invalid classes* and discuss this phenomenon in the section 6. The number of invalid classes can be a good metric to compare the models. For our best performing model - DP-II - 165 classes over 1113 (14.8%) are affected by this phenomenon (most of them, **157** - 95% - are biased towards negative). Since this phenomenon is easily detectable during evaluation, we know exactly which skills cannot be predicted and then propose results without these classes.

Table 2 displays the evaluation metrics when invalid classes are removed and Figure 4 depicts the distribution of the evaluation metrics across classes for the best performing model (DP-II). Even with the removal of invalid classes, we can still notice a wide distribution of the model performances across classes. The most variable metric is the recall, for which a quarter of all classes has a score lower than 40%, half of the valid classes has a recall score between 40% and 90 %. We can also notice that the precision scores are less variable since the first and third quartile are closer to the median value of the scores. The precision score is between 50% and 80% for 50% of all classes after invalid classes removal.

6. Discussion

In this section we discuss the experimental results and use them to answer the research questions. First we analyze the invalid classes problem in 6.1 and then we conclude on the performances of the method in 6.2.

Model	F1			Recall			Precision			Invalid classes
	micro	macro	weighted	micro	macro	weighted	micro	macro	weighted	
DP-II	**0.84**	**0.50**	**0.66**	0.86	**0.52**	**0.69**	0.81	**0.54**	**0.67**	**165.00**
DP-I	**0.84**	0.49	0.65	**0.87**	0.51	**0.69**	0.81	0.53	0.66	203.00
DP-III	**0.84**	0.49	0.65	**0.87**	0.51	**0.69**	0.81	0.51	0.65	210.00
ST-VI	**0.84**	0.48	0.65	0.86	0.51	0.68	**0.82**	0.52	0.66	202.00
ST-IV	**0.84**	0.48	0.65	0.86	0.50	0.68	**0.82**	0.51	0.65	254.00
DP-IV	**0.84**	0.47	0.64	0.86	0.49	0.68	0.81	0.51	0.65	238.00
ST-VII	0.83	0.47	0.64	0.85	0.49	0.67	**0.82**	0.50	0.64	221.00
ST-I	**0.84**	0.47	0.64	0.86	0.49	0.68	0.81	0.50	0.64	236.00
ST-III	**0.84**	0.46	0.64	0.86	0.48	0.67	**0.82**	0.50	0.64	263.00
ST-II	0.83	0.46	0.64	0.85	0.47	0.66	**0.82**	0.51	0.65	240.00
ST-V	0.83	0.43	0.62	0.84	0.46	0.65	0.81	0.45	0.61	325.00
DAN	0.82	0.41	0.60	0.82	0.44	0.63	**0.82**	0.45	0.61	290.00

Table 1
Model performances.

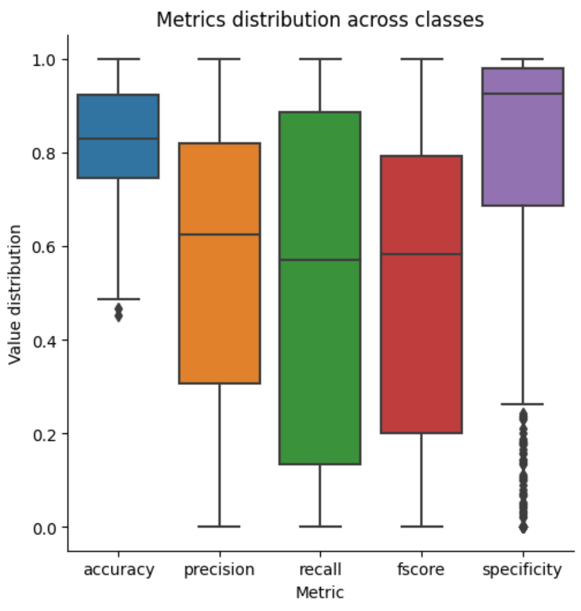

Figure 3: Metrics distribution across classes for the best model (DP-II)

6.1. The invalid classes problem

The aggregated performances of the different model variants are close and, according to the experimental results, the main difference between them is the number of invalid classes. Indeed, this metric varies by a factor of two between the best and the worst performing model. Due to the small size of our data set and the stochastic nature of the negative examples sampling during training, finding a definitive explanation of this phenomenon is not possible, although we can hypothesize some explanations.

Model	F1			Recall			Precision		
	micro	macro	weighted	micro	macro	weighted	micro	macro	weighted
ST-IV	**0.83**	**0.60**	**0.71**	**0.86**	**0.63**	**0.74**	0.80	**0.65**	**0.72**
DAN	**0.83**	0.56	0.70	0.85	0.59	**0.74**	**0.82**	0.61	**0.72**
DP-II	**0.83**	0.58	0.70	**0.86**	0.60	0.73	0.80	0.63	0.71
ST-III	0.82	0.59	0.70	0.85	0.61	0.73	0.79	0.63	0.71
ST-VI	0.82	0.58	0.69	0.85	0.60	0.73	0.79	0.62	0.70
ST-II	0.82	0.57	0.69	0.84	0.59	0.72	0.80	0.63	0.71
DP-IV	0.81	0.58	0.69	0.85	0.61	0.73	0.78	0.63	0.70
DP-III	0.82	0.58	0.69	**0.86**	0.61	0.73	0.78	0.61	0.69
ST-VII	0.82	0.57	0.68	0.84	0.59	0.72	0.79	0.61	0.69
ST-V	0.80	0.58	0.68	0.84	0.62	0.72	0.77	0.61	0.69
ST-I	0.81	0.58	0.68	0.84	0.61	0.72	0.77	0.62	0.69
DP-I	0.81	0.57	0.67	0.85	0.60	0.71	0.77	0.62	0.69

Table 2
Model performances after invalid classes removal.

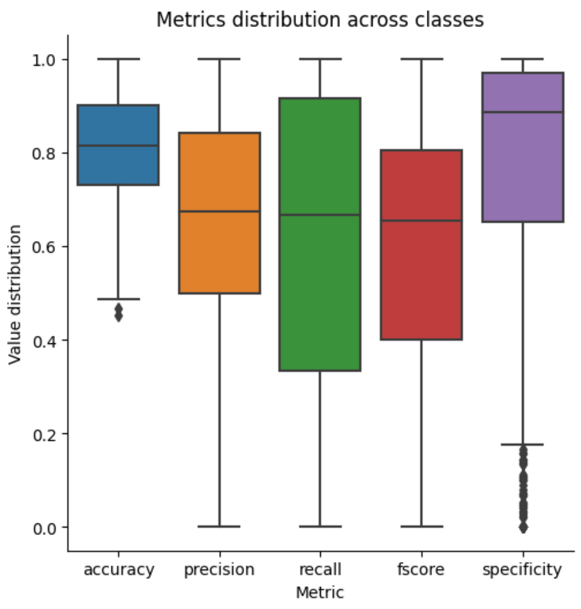

Figure 4: Metrics distribution across classes for the best model (DP-II) after invalid classes removal.

The first is envisioned in section 4.1 and links the asymmetry of classes populations in the data set used to the possibility of oversampling some classes in the negative examples.

A second hypothesis can be formulated by correlating the model complexity (number of layers) with the number of invalid classes. Indeed, we can see that models with more trainable layers tend to have a greater number of invalid classes, thus allowing us to suspect over-fitting of the models on our small data set.

Further work is required to correlate the classes distributions in the training data set to the performances of the model for theses classes and, if a link is established, develop a new negative sampling method. To confirm and evacuate the potential over-fitting of the model, our approach should be applied to a bigger data set. For now, we consider that this shortcoming can be overcome by temporarily removing the unpredictable skills from the dictionary when re-using the model for inference.

6.2. Performances of the skill prediction

The distributions of per-class performances of our best model are wide. A significant amount of skills cannot be predicted reliably. However, for 50% of them (556 skills), our best model achieves a F1-Score of at least 60%. For 25% of the skills (278) the F1-Score is even superior to 80%. We believe these performances to be encouraging for a single model, especially when compared with the small number of skills predictable in related works [12, 11]. Furthermore, the work hereby presented did not focus on the skills dictionary building, hence the performances of the model have been hindered by the lack of pre-processing of the extracted skills. Indeed, the skills dictionary has been built from free-text surface forms without any normalization. This lack of pre-processing left our skills dictionary with duplicates like *xsl*, *xslt* and *xsl/xslt* or *vbscript* and *vb script*. Although we cannot demonstrate the impact of the absence of skills surface-forms canonization, we can hypothesize that:

- The existence of duplicate skills prevents the skills embedding step from adequately capturing the similarities between them ;
- A duplicate skill can be sampled for a negative example while a synonymous skill is used as a positive example for the same profile.

We conclude that the responses to our research questions are:

- **RQ-1** : the architecture of our model is able to capture valuable information for skill prediction for a significant amount of skills;
- **RQ-3** : for a significant amount of skills, our approach to multi-label classification yields encouraging results. Further work is required to correlate similarity between skills hidden representations and the per-class model performances.

An answer to **RQ-2** can be provided by analyzing per-class performances. To discuss this research question, we provide the performances of the best model on a few rare skills along with the number of examples in the test data set:

- *zookeeper*: 49 negative examples, 6 positive examples - F1-Score 90.9%;
- *windows server 2008*: 32 negative examples, 4 positive examples - F1-Score 88.9%;
- *wan*: 43 negative examples, 4 positive examples - F1-Score 66.7%;
- *visual force*: 32 negative examples, 2 positive examples - F1-Score 100%.

Although the model fails for some rare skills, it performs well for some others. Which is an encouraging results for **RQ-2** as it demonstrates our approach has the potential to predict rare skills accurately.

7. Conclusion

In this paper, we presented a method to tackle skill prediction with a deep learning model using unordered composition functions for the handling of textual data. We also introduced a method for transforming a multi-label classification task to a binary classification task.

To evaluate this method, we proposed an experimental evaluation on a small open data set and showed that this approach has an interesting potential for skill prediction. The proposed evaluation framework allows for a thorough evaluation of the model performances - overall and for specific skills.

The approach to skill prediction introduced in this paper allows for a job recommender system to close the gap between the declared skills of an individual and their actual skills. The proposed deep learning architecture allows for more lightweight models than the syntactical models of the state of art, especially the ones relying on document modeling through an hierarchical architecture - thus allowing faster inferences.

Further work will study the integration of this model in a job recommender system for skill prediction of both profiles and job offers prior to the matching between them. As part of future work, we also plan to enhance this method - by addressing the discussed shortcomings - and to evaluate it on different multi-label text classification tasks similar to the one tackled in this paper - i.e. numerous possible classes with implicit relationships between them and specialized texts to classify.

8. Acknowledgments

This research has been funded by the French government through the Association Nationale Recherche et Technologie (ANRT) under the CIFRE convention 2018/0400 between Aix-Marseille University and MOBEN & ROOSTER[7].

References

[1] E. Tinelli, A. Cascone, M. Ruta, T. Di Noia, E. Di Sciascio, F. M. Donini, I.M.P.A.K.T.: An innovative semantic-based skill management system exploiting standard SQL, in: ICEIS 2009 - 11th International Conference on Enterprise Information Systems, Proceedings, volume AIDSS, 2009, pp. 224–229. doi:10.5220/0002008802240229.

[2] K. Stencel, A. Janusz, K. Ciebiera, D. Ślęzak, S. Stawicki, M. Drewniak, How to Match Jobs and Candidates - A Recruitment Support System Based on Feature Engineering and Advanced Analytics, 2018, pp. 503–514. URL: https://www.researchgate.net/publication/325207539. doi:10.1007/978-3-319-91476-3_42.

[3] M. Bastian, M. Hayes, W. Vaughan, S. Shah, P. Skomoroch, H. J. Kim, S. Uryasev, C. Lloyd, Linkedin skills: large-scale topic extraction and inference, in: RecSys '14, 2014.

[4] F. Javed, P. Hoang, T. Mahoney, M. McNair, Large-scale occupational skills normalization for online recruitment, in: AAAI, 2017.

[7]https://www.mobenrooster.com/

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 30, Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[6] M. Iyyer, V. Manjunatha, J. L. Boyd-Graber, H. Daumé, Deep unordered composition rivals syntactic methods for text classification, in: ACL, 2015.

[7] J. Lee, Y. Lee, J. Kim, A. R. Kosiorek, S. Choi, Y. Teh, Set transformer: A framework for attention-based permutation-invariant neural networks, in: ICML, 2019.

[8] M. Zhao, F. Javed, F. Jacob, M. McNair, Skill: A system for skill identification and normalization, in: AAAI, 2015.

[9] M. le Vrang, A. Papantoniou, E. Pauwels, P. Fannes, D. Vandensteen, J. Smedt, Esco: Boosting job matching in europe with semantic interoperability, Computer 47 (2014) 57–64.

[10] V. S. Dave, B. Zhang, M. Hasan, K. AlJadda, M. Korayem, A combined representation learning approach for better job and skill recommendation, Proceedings of the 27th ACM International Conference on Information and Knowledge Management (2018).

[11] Z. Wang, S. Li, H. Shi, G. Zhou, Skill inference with personal and skill connections, in: COLING, 2014.

[12] F. F. J. Kameni, N. Tsopzé, Skills prediction based on multi-label resume classification using cnn with model predictions explanation, Neural Comput. Appl. 33 (2021) 5069–5087.

[13] S. Bian, X. Chen, W. X. Zhao, K. Zhou, Y. Hou, Y. Song, T. Zhang, J.-R. Wen, Learning to match jobs with resumes from sparse interaction data using multi-view co-teaching network, Proceedings of the 29th ACM International Conference on Information & Knowledge Management (2020).

[14] M. Schlichtkrull, T. Kipf, P. Bloem, R. van den Berg, I. Titov, M. Welling, Modeling relational data with graph convolutional networks, ArXiv abs/1703.06103 (2018).

[15] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, in: NAACL, 2019.

[16] X. Zhang, F. Wei, M. Zhou, Hibert: Document level pre-training of hierarchical bidirectional transformers for document summarization, in: ACL, 2019.

[17] C. Qin, H. Zhu, T. Xu, C. Zhu, L. Jiang, E. Chen, H. Xiong, Enhancing person-job fit for talent recruitment: An ability-aware neural network approach, The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (2018).

[18] S. Maheshwary, H. Misra, Matching resumes to jobs via deep siamese network, Companion Proceedings of the The Web Conference 2018 (2018).

[19] T. Dozat, Incorporating nesterov momentum into adam, 2016.