Studo Jobs: Enriching Data With Predicted Job Labels

Markus Reiter-Haas
Moshbit GmbH
Graz, Austria
markus.reiter-haas@studo.co

Valentin Slawicek
Moshbit GmbH
Graz, Austria
valentin.slawicek@studo.co

Emanuel Lacic
Know-Center
Graz, Austria
elacic@know-center.at

ABSTRACT

In this paper, we present the Studo Jobs platform in which we tackle the problem of automatically assigning labels to new job advertisements. For that purpose we perform an exhaustive comparison study of state-of-the-art classifiers to be used for label prediction in the job domain. Our findings suggest that in most cases an SVM based approach using stochastic gradient descent performs best on the textual content of job advertisements in terms of Accuracy, F_1-measure and AUC. Consequently, we plan to use the best performing classifier for each label which is relevant to the Studo Jobs platform in order to automatically enrich the job advertisement data. We believe that our work is of interest for both researchers and practitioners in the area of automatic labeling and enriching text-based data.

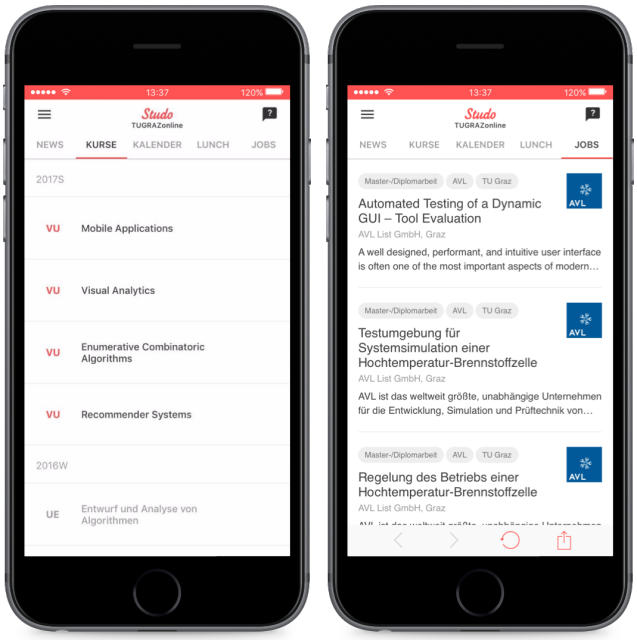
KEYWORDS

job platform; data enrichment; label prediction; comparative study;

1 INTRODUCTION

The nature of the job market is a highly competitive one and finding a new job is not an easy decision as it usually depends on many factors like salary, job description or geographical location. This has led to the recent rise of business-oriented social networks like LinkedIn[1] or XING[2]. Users of such networks organize and look after their profile by describing their skills, interests and previous work experiences. But finding relevant jobs for users using such carefully structured content is actually a non-trivial task to perform [1]. Tackling the same problem gets even more difficult for university students, as they normally have only some or no relevant work experience at all. This has become a real issue for students as they get more aware that having a degree does not automatically guarantee them their desired job after graduation. For instance, the recent study of [10] reports that one third of graduates in the U.S. were employed in positions that do not require a university degree. Moreover, the authors report that 23.5% of employed graduates in 2013 are not only underemployed but also work in positions with a below-than-average salary.

In 2016 we launched the Studo[3] mobile application with the initial aim to provide constant guidance and support to Austrian students in their everyday life. As seen in Figure 1a, Studo integrates several university-relevant services (e.g. course management, mail, calendar) but also enriches the student's daily life by providing relevant news articles. With such a feature-set a student is not only better informed but also encouraged to connect and collaborate with other peers from the same community. Moreover, the ever increasing popularity of the application at Austrian universities[4] has shown that students clearly need additional guidance throughout their studies. Thus, one of the main goals of Studo is to better prepare the students for the job market they need to face after graduation.



(a) (b)

Figure 1: Screenshot of (a) the course overview in the Studo mobile application and, (b) the Studo Jobs platform.

Current work. In this paper, we present our work-in-progress of the newest extension to Studo - the Studo Jobs platform. As seen in Figure 1b, students can browse related job advertisements in order to gather relevant working experience even before they graduate. In our case, these job advertisements typically describe the candidate's job role, required skills, the expected educational background as well as the company description, but only in an unstructured free-text format. In the context of job recommendations, having such unstructured data can be problematic as students already struggle with having little job experience [13]. To overcome this limitation, in this work we focus on enriching the job advertisement data by automatically generating and assigning labels (i.e., categories) to which an advertised job belongs. The benefits of such data enrichment are twofold. First, students can more easily navigate through the available job offers (e.g., by filtering out irrelevant categories). Second, by correctly enriching the job advertisement data we hope to increase the performance of future job recommendations (e.g., by

[1] http://linkedin.com
[2] http://xing.com
[3] https://studo.co/
[4] As of June 2017 the 30,000 monthly active users have on average 100 applications starts per month

performing clustering like in [9]). As such, we perform an extensive algorithmic comparison study on how to predict suitable labels for a particular job advertisement. We believe that our findings can support both developers and researchers on how to enrich their data and potentially improve the recommendation performance.

2 RELATED WORK

Most work which is related to assigning labels to job advertisements come from the research on multi-label classification, an emerging machine learning paradigm which tries to assign a set of labels to a document [17]. In an extensive literature review [20], previous multi-label learning has been divided into two main groups, i.e., algorithm adaptation and problem transformation methods. Algorithm adaptation methods adapt popular learning techniques to deal with multi-label data directly, while problem transformation methods transform the multi-label classification problem into either one or more single-label classification problems. In our work we build upon the later, i.e., explore on how to construct several single-label classifiers in order to assign relevant labels to job advertisements. We base our decision as other work have shown that binary relevance is a suitably method to tackle the problem of multi-label classification (e.g., [16]). Moreover, the author of [18] have shown that task of predicting the job sector (i.e., category or label as in our case) can be done more accurately than its title or education requirement.

3 METHODOLOGY

Similar to the work of [18], we train multiple binary classifier on the basis of features contained in the text (i.e. terms). Thus, for each label l we define a parameter vector for class c: $\theta_c = \{\theta_{c1}, \theta_{c1}, ..., \theta_{cn}\}$, where n is the size of the vocabulary in the corresponding training set. The values of vector θ_c are the calculated TF-IDF values as denoted by Equation 1 and 2, where TF(t,j) is the term count within the job advertisement and DF(t) is the number of job advertisements in which that particular term occurs.

$$\text{TF-IDF(t,j)} = \text{TF(t,j)} \times \text{IDF(t)} \tag{1}$$

$$\text{IDF}(t) = log\frac{1+n}{1+\text{DF}(t)} + 1 \tag{2}$$

For our comparison study we performed experiments on different job labels using several classification algorithms. As a baseline, we first explored three well-known algorithms from the literature. Specifically, we looked into: (1) the Naive Bayes algorithm which assumes pairwise independence of the input features [19], (2) a Classification And Regression Tree (CART), where at each node one input is tested and depending on the results the left or right subbranch is traversed [2] and, (3) a Random Forest ensemble approach, where each tree votes for a particular class [4].

Next we experimented with AdaBoost, a boosting algorithm which does adaptive weight adjusting of incorrectly classified instances [7]. Another approach was a linear model using Logistic Regression which assumes that the posterior probability of a class is equal to a logistic sigmoid function acting on a linear function [3]. Support Vector Machines (SVM) is another algorithm that has been shown to perform well with text classification. As such, we used two SVM methods: (1) a Linear SVM which tries to fit a hyperplane with the maximum soft margin, thus allowing for a minimal number of errors [6] and, (2) a SVM-SGD approach, where the stochastic gradient descent optimization method is applied on the SVM [21].


```
{
"id" : "123456",
"jobTitle" : "Junior Java Developer",
"text" : "Ihre Aufgaben: Erhebung der Anforderungen fuer die...",
"labels" : ["Software"]
},
{
"id" : "456789",
"jobTitle" : "Marketing, Sales & Event Manager",
"text" : "Rocken−motivieren−begeistern! Schoen, dass Du...",
"labels" : ["Marketing", "Catering", "Graphics, Design"]
}
```


Listing 1: Example of two crawled job advertisements (text was shortened for readability) in JSON format. In our experiments we only used the *text* of a given job in order to predict the best suited labels.

Finally, we experimented with three different neural network approaches. We first trained a Multilayer Perceptron (MLP) which consists of an input layer that is provided with the input vector θ_c, followed by one hidden layer with a size of 1024 units and two smaller hidden layers with 128 units each. Each of the hidden layers is followed by a batch normalization layer. The next two models are based on the work of [11]. As such, we used a convolutional neural network (CNN) with an embedding layer of 200 units. The embedding layer is followed by a 1-dimensional convolutional layer with 128 filters and a kernel size of 3. This was followed by a global max pooling layer and a dense layer with 128 units. The third model was a multichannel CNN (M-CNN) that connects the embedding layer with three convolutional layers in parallel, each one having 128 filters and a kernel size of 3, 4 and 5 respectively. Every convolutional layer is followed by a max pooling layer which outputs are then afterwards merged together. In all three networks we used rectified linear units as the activation function (i.e., Equation 3). The output layer has always two units and uses a standard softmax activation function (i.e., Equation 4). Each of the hidden layers uses dropout with a rate of 0.2 for regularization. The models also use the Adam optimizer [12] with a learning rate of 0.001. It also needs to be noted that the last two CNN approaches do not utilize a TD-IDF based input vector as the rest. The input is generated by transforming the textual content of a job advertisement into a sequence of word indices. Having a maximum sequence length of 1,000, shorter texts were just padded with a default zero index.

$$f(x) = x^+ = \max(0, x) \tag{3}$$

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{4}$$

4 EXPERIMENTAL SETUP

In order to perform a comparative study we first constructed a training and test set by crawling job advertisement data from leading

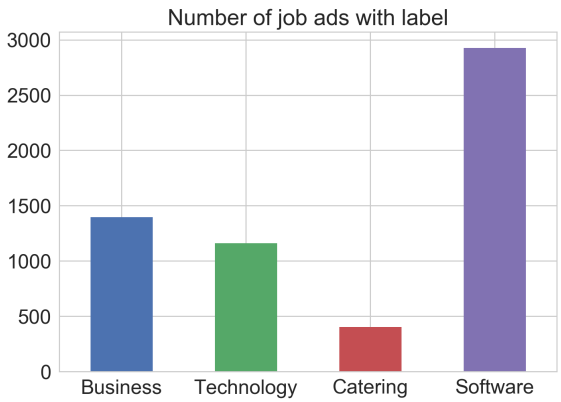


Figure 2: Occurences of the four Studo Jobs labels in the crawled dataset. Management and Marketing are merged together to the Business label.

Austrian job platforms, i.e., stepstone.at, karriere.at and monster.at. We utilized an incremental crawler which was given a manually constructed list of URLs for each job type. If a particular job advertisement was not in the database a new entry was added, otherwise it was enriched with a new label. The crawler then iterated over a finite number of pages. As seen in Listing 1, an extracted job entry consists of an id, a title and description in plain text and a list of labels which denote the type of the job.

Dataset. We crawled 5,602 job advertisements in total. On average, a posted job had 1.05 labels assigned to it. In our experiments we focused on four different labels which are mainly used in the Studo Job platform, namely: *Software*, *Catering*, *Technology* and *Business*. As the Studo-specific label *Business* could not be directly crawled, we derived it by combining job advertisements from the type *Management* and *Marketing*.

Evaluation. As the crawled dataset is clearly imbalanced (e.g., as seen in Figure 2 the *Software* label dominates), we further constructed subdatasets for each label to experiment on. Thus, for each label a subset was used containing all the jobs containing that particular label as well as a random sample of the same size containing other labels. Therefore the resulting subdataset had 50% of job advertisments containing the evaluating label and 50% without it. The evaluation was performed using Scikit-learn [14] for the Naive Bayes, CART, Random Forest, AdaBoost, Logistic Regression, Linear SVM and the SVM-SGD approach. For the neural networks we utilized Keras [5]. These models were trained and evaluated using a 10-fold stratified cross-validation on each subdataset respectively. In order to finally quantify the prediction performance, we used a set of well-known information retrieval metrics. In particular, we report the prediction accuracy by means of Accuracy, the F_1-measure and the Area Under the ROC curve (AUC) [15].

5 RESULTS

The overall results of the algorithmic comparison can be seen in Table 1. As each subdataset had a different size of the test set (i.e., due to the imbalanced nature of the original dataset), for each approach we report a weighted average in terms of Accuracy. F_1-measure and AUC. For example, the F_1-measure would be calculated as: $F_1 = \alpha F_1(Business) + \beta F_1(Technology) + \gamma F_1(Catering) + \delta F_1(Software)$, where the weight values contain the percentage of the corresponding label in the dataset (i.e., $\alpha = 0.237$, $\beta = 0.197$, $\gamma = 0.068$ and $\delta = 0.497$). In general, we found strong accuracy performance in all models (e.g., the worst performing Random Forest did have an Accuracy of 0.8017 and F_1 of 0.7932). The best performing approaches were the SVM based ones, where the linear approach had the best AUC and the one using stochastic gradient descent had Accuracy and F_1. Interestingly enough, due to the recent popularization of deep learning approaches our first assumption was that the models based on CNN would perform much better than the SVM based ones. Although still competitive, we assume that a lower accuracy performance was reached because the hyperparameters were not previously tuned enough and early stopping was not used in order to cope with overfitting. As such, we hypothesize that there is still much to gain from such approaches by learning these parameters beforehand (e.g., using a nested cross-validation like in [8]) and incorporating a validation set to stop the model training at the most optimal time.

Algorithm	Accuracy	F_1-measure	AUC
Random Forest	0.8017	0.7932	0.8739
CART	0.8046	0.8053	0.8054
MLP	0.8224	0.8253	0.8923
AdaBoost	0.8331	0.8300	0.8952
Naive Bayes	0.8470	0.8555	0.9069
M-CNN	0.8571	0.8549	0.9177
CNN	0.8597	0.8604	0.9236
Logistic Regression	0.8661	0.8653	0.9335
Linear SVM	0.8707	0.8709	**0.9344**
SVM-SGD	**0.8748**	**0.8751**	0.9329

Table 1: Average performance across all labels. Each label is weighted with the number of samples in its subdataset.

The individual label results in terms of the F_1-measure can be seen in Figure 3. We show only the F_1-measure due to space restrictions and the fact that the Accuracy values, when compared to, are almost identical. The best performance was achieved on the *Software* label using the SVM-SGD approach with an Accuracy of 0.9193 and F_1 of 0.9196. A contributing factor to such performance is possibly the size of the training set which was by far the largest for the *Software* label. An interesting finding is that all of the approaches that were utilized on the *Catering* label, which had the least training data, performed much better than on the *Business* and *Technology* label. Looking at the data, we think that the reason for such a performance difference lies in the broader definition of these label terms. Moreover, the reason that the *Bussiness* label had the worst performance could lie in the fact that its data come from a combination of the crawled *Management* and *Marketing* labels. It should also be noted that the MLP approach outperformed all others when applied to the *Catering* label. This suggests that when the right hyperparameters are picked, an increase in performance could still be gained.

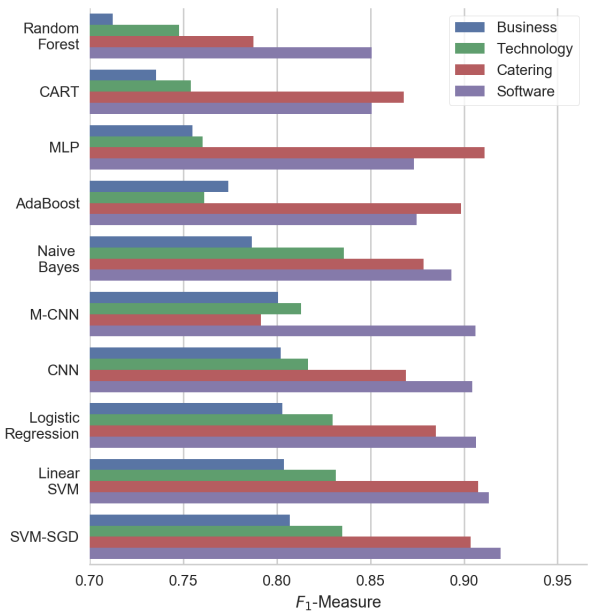


Figure 3: Performance results of the utilized binary classification algorithms in terms of the F_1-measure for all labels relevant to the Studo Jobs platform.

Overall, the SVM-SGD performed best. However, the MLP approach outperformed others for the *Catering* label and the much simpler Naive Bayes had almost the same performance as SVM-SGD for the *Technology* label. This suggests that a diversified model combination could lead to even better performance.

6 CONCLUSION

In this work we presented the Studo Job platforms and showed how we plan to tackle the problem of automatically assigning labels to new job advertisements. For that purpose we performed an extensive comparative study between several state-of-the art text-classification algorithms. Our findings suggest the by utilizing an SVM approach using stochastic gradient descent we can achieve the best performance in terms of Accuracy, F_1-measure and AUC. However, our results revealed that deep learning approaches can also improve the prediction performance, especially with a right hyperparameter setup. As such, for our Studo Jobs platform we will use a combination of those binary classifiers which showed the best performance results.

Limitation and Future Work. As already mentioned, one limitation of our work is that we did not extensively explore the impact of choosing the right hyperparameters for the deep learning approaches. Therefore, we plan to extend the study by finding the optimal hyperparameters for each label that is relevant to the Studo Jobs platform (e.g., by setting up a nested cross-validation). In addition, we also plan to extend our comparison study by including other features besides the textual terms and incorporating methods that adapt algorithms to directly perform multi-label classification. Building on the data enrichment of job advertisements, we further plan to integrate the generated labels and assess their impact on perceived usefulness and navigability to users in a live setting (e.g., by letting users define and store filters to narrow down the search for relevant jobs). Finally, we plan to extend the Studo Job platform with personalized recommendations which leverage the automatically generated job labels. We not only want to investigate which approaches (e.g., content-based, collaborative filtering, etc.) benefit the most from such data, but also on how to incorporate recent label filters as additional time-dependent contextual cues in order to predict the current job interest. For this we also plan to investigate the recently popularized deep learning approaches (e.g., recurrent neural networks) to see if we can predict the future shift in interest of a job type.

Acknowledgments. This work is supported by the Know-Center and ISDS Institute from Graz University of Technology. The authors would also like to thank the AVL company, especially Dr. Markus Tomaschitz, for the support at setting up this research project and giving insights about the job market.

REFERENCES

[1] F. Abel. We know where you should work next summer: job recommendations. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 230–230. ACM, 2015.
[2] Berk. Classification and Regression Trees. *Data Mining with Rattle and R: The Art of Excavating Data for Knowledge Discovery, Use R*, (November):36–350, 2009.
[3] C. M. Bishop. *Pattern Recognition and Machine Learning*, volume 53. 2013.
[4] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
[5] F. Chollet and Others. https://github.com/fchollet/keras.
[6] C. Cortes and V. Vapnik. Support Vector Networks. *Machine Learning*, 20(3):273–297, 1995.
[7] Y. Freund and R. E. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. 139:23–37, 1995.
[8] I. Guyon, A. R. S. A. Alamdari, G. Dror, and J. M. Buhmann. Performance prediction challenge. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pages 1649–1656. IEEE, 2006.
[9] W. Hong, S. Zheng, H. Wang, and J. Shi. A job recommender system based on user clustering. *Journal of Computers*, 8(8):1960–1967, 2013.
[10] J. Jones, J. Schmitt, et al. A college degree is no guarantee. Technical report, Center for Economic and Policy Research (CEPR), 2014.
[11] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.
[12] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. pages 1–15, 2014.
[13] R. Liu, Y. Ouyang, W. Rong, X. Song, C. Tang, and Z. Xiong. Computational Science and Its Applications – ICCSA 2016. 9788:453–467, 2016.
[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
[15] D. M. Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.
[16] J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. *Machine Learning*, 85(3):333–359, 2011.
[17] G. Tsoumakas and I. Katakis. Multi-Label Classification : An Overview. *Int J Data Warehousing and Mining*, 2007:1—-13, 2007.
[18] J. Zavrel, P. Berck, and W. Lavrijssen. Information Extraction by Text Classification: Corpus Mining for Features. *Proceedings of the Second International Conference on Language Resources and Evaluation LREC00*, 2000.
[19] H. Zhang. The Optimality of Naive Bayes. *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference FLAIRS 2004*, 1(2):1 – 6, 2004.
[20] M. L. Zhang and Z. H. Zhou. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837, 2014.
[21] T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. *Proceedings of the twenty-first international conference on Machine learning*, 6:116, 2004.