Automated Machine Learning for Information Retrieval in Scientific Articles

Hojjat Rakhshani^1, Bastien Latard^{1,2}, Mathieu Brévilliers^1, Jonathan Weber^1, Julien Lepagnot^1
Germain Forestier^1, Michel Hassenforder^1, Lhassane Idoumghar^1
^{1,2}*Université de Haute-Alsace, IRIMAS UR 7499, F-68100 Mulhouse, France – firstname.lastname@uha.fr*
^{2}*MDPI AG, Basel, Switzerland*

***Abstract*—The amount of scientific conferences and journal articles continues to increase and new approaches are required to support users in finding relevant publications. This study investigates to what extent a new machine learning (ML) pipeline may preferentially identify links between similar scientific articles. The characteristics of intersections and unions of keywords, contextualized keywords (i.e., synsets) and neighbors are computed and used to train a ML model. Automated machine learning (AutoML) is then applied to ease the search for a new pipeline. Extensive experiments demonstrated that a newly designed ML model is able to achieve an accuracy of 0.90 on a dataset of approximately 120,000 article pairs. These results suggest that application of ML for proposing new recommendation systems could have in the long term a positive impact in the literature.**

***Index Terms*—Word sense disambiguation, AutoML, semantic similarity, information retrieval, evolutionary algorithms.**

I. Introduction

The explosive growth in the number of scientific articles presents a challenge to the search for relevant literature and might hinder the access to the right papers [1]. In these circumstances, most researchers have to commit their time to search for recent findings which is labor-intensive. Scientific recommender systems (SRS) benefit the researchers to be well aware of most recent works in their community with the crucial aim of saving their precious time [2], [3]. They do so by considering necessary information extracted from an input source. This might include profiles of users, keywords, citation analysis, a single paper, online behaviors, and so on. The main idea has become very popular since the introduction of the famous "*Customers who bought this item also bought...*" from Amazon, or other Netflix movie recommendation systems.

Co-occurrence-based, collaborative filtering and content-based approaches are among the main groups of recommendation algorithms applied to scientific literature. In the first case, the goal is to find relatedness between articles, sometimes from textual co-occurrences, often relying on the citations graph—assuming that two articles having references in common are somehow related. An illustrative example is described in [4], where a citations graph is used to identify the similarity between two articles. The main drawback of this approach is that a citation might be generic (or even unrelated) or from a negative sentiment (e.g., "*Authors made the wrong assumption that...*"). Alternatively, collaborative filtering approaches have emerged [5], [6]. In these methods, users are the center of interest and suggestions are usually based either on their readings or ratings. Consequently, only accessed or rated articles can be suggested to a user. Matsatsinis [5] tackled this problem by forcing the users to set up their profile first in order to get any related suggestions. Meanwhile, these approaches suffer from the cold start problem (i.e., a lack of data when launching the algorithm or when a new article is added), or from low rating participation. Finally, the last group incorporates also article metadata such as abstracts, titles, and keywords. These content-based recommendation systems are the most used approaches in the community; based on Beel's survey [7]. In the case of SRS, the content is represented by text and information retrieval (IR) methods are applied to find the most relevant piece of information. Among many techniques, classical retrieval models (e.g., Boolean, TFIDF, vector models) and enhanced probabilistic models (e.g., BM25) are widely applied in the literature [8]. Although the general-purpose probabilistic models have been successfully embedded in SRS [7], they are often considered as "black boxes" from which retrieval details or suggestion failures are difficult or even impossible to understand. Therefore, improving or correcting recommendations obtained from these approaches is a tedious task, and recurrent errors tend to happen.

To the best of our knowledge, there is no work on applying machine learning methods for IR of SRS which is the main contribution of this paper. The proposed approach resolves ambiguous keywords, expand those and creates features for potential use in ML methods. To be effective in practice, several AutoML algorithms are used to automatically tune and design a new model for the problem at hand. We propose to measure the similarity of scientific articles based on two pre-processing steps, namely categorization and data augmentation. Those respectively disambiguate keywords by finding common categories among the potential senses, and augment the data by retrieving every related neighbor. As the features given to the adopted methods are directly inherited from these two steps, they can be enhanced, modified, or adapted if critical errors are detected. This also provides the possibility of showing to the users how and why articles are suggested as related articles. Therefore, a user can judge whether or not a recommendation is correct before accessing it.

II. Related Works

In the recent years, a number of surveys have been presented to provide a review of related works on ML based recommendation systems [9], [10], [11], [12]. Accordingly, we can see that there are very few works which well shape the application of ML for SRS. In [13], authors introduced a collaborative filtering method based on deep neural networks. They trained network using normalized user-rating and normalized item-rating vectors. Ebesu and Fang [14] proposed a flexible encoder-decoder architecture capable of incorporating author meta-data for context-aware citation recommendation. Experimental results on the large-scale CiteSeer dataset show that the proposed method has a significant improvement over its competitive baselines. In another work [15], a collaborative model based on an extension of the stacked denoising autoencoder is proposed. The authors show that the new model is able to jointly learn users and items' latent factors from both side information and the rating matrix by conducting some experiments on two datasets containing ratings from users on movies and books. Cai et al. [16] put forward a generative adversarial network based heterogeneous bibliographic network representation to design a personalized citation recommendation system. Interestingly, Wang et al. [17] proposed to use the convolutional neural networks for the editor article recommendation task. Moreover, Bansal et al. [18] incorporated recurrent neural networks to represent text items for collaborative filtering method, considering the task of scientific article recommendation. Altogether, these success stories of ML for SRS motivated us to investigate the efficiency of a new ML pipeline model for IR of scientific articles.

III. Materials and Methods

In the context of ML-based IR, the main focus should be on building a high quality ML pipeline. This is due to the fact that there is no universal approach and new ML pipelines have to be constructed for each new data set [19]. The pipeline is a linear sequence of machine learning process (usually data cleaning, feature selection, and modeling) that transforms an input vector $\mathbf{x} \in \mathbb{X}$ into a target value $\mathbf{y} \in \mathbb{Y}$. This task can be based on the structure of the pipeline and the choice of the ML algorithms and their hyperparameters.

Definition 1 (Pipeline Creation Problem): *Given a set of algorithms $\mathcal{A}$ and their associated hyperparameters Λ, a training set $\mathcal{D}_{train}$ and a validation set $\mathcal{D}_{valid}$ such that $\mathcal{D}_{train} \cap \mathcal{D}_{valid} = \varnothing$, the pipeline creation problem (PCP) can be defined as a joint algorithm and hyperparameter selection minimization problem using a loss function $\mathcal{L}$ [19]:*

$$g^*, A^*, \lambda^* \in \underset{g\in G, A\in\mathcal{A}^+, \lambda\in\Lambda}{arg\,min} \; \mathcal{L}\left(\underset{g,A,\lambda}{\mathcal{P}}(\mathcal{D}_{train}), \mathcal{D}_{valid}\right) \quad (1)$$

Here, g is a directed acyclic graph (DAG) that denotes the structure of the pipeline $\underset{g,A,\lambda}{\mathcal{P}}$. In g, the nodes (consisting of the selected algorithm A and its associated hyperparameters λ) represent an arbitrary machine learning process and edges represent the flow of an input. The performance of the configuration (g, A, λ) should be evaluated using the validation set $\mathcal{D}_{valid}$.

In this work, the pipeline structure g is supposed to have a fixed shape which eliminates the complexity of creating a DAG graph. Figure 1 introduces the general workflow of our pipeline. The first node exploits the keywords of the article and finds common categories among those (Section III-A). This categorization also validates the context of the keywords. After this disambiguation phase, data is augmented (Section III-B) by adding semantic neighbors (i.e., synonyms, hypernyms, hyponyms, etc.). From these two steps, different types of variables can be used to connect two scientific articles. Indeed, common keywords or neighbors can be shared among those, in different proportions. Thereafter, a set of five state-of-the-art optimization techniques are used to automatically build a superior ML model for the problem at hand. Here, different hyperparameter optimization (HPO), neural architecture search (NAS), and combined algorithm selection and hyperparameter optimization (CASH) problem formulations are used. Accordingly, we will have an ensemble of five ML models where only the best one is allowed to exist among those alternatives.

The computational nodes are fed with connectivity details obtained from the aforementioned two steps (see Section III-C) and trained to determine whether a pair of articles is similar or dissimilar. Given that no ground truth is available, the similarity is based on the assumption that articles belonging to journals along the same scope (or from the same journal) are similar. Figure 2 is a simplified illustration of the proposed approach. Articles go through the same categorization process and their degree of connections (see Sec. III-C) are used to predict their similarity.

A. Categorization

This approach disambiguates article keywords using the lexical knowledge database BabelNet [20], assuming that the more keywords within the same article sharing categories in common, the higher the probability that the category is one contribution of the article. An enhanced n-gram approach splits multiple-word keywords when no data is found in the BabelNet ontology, and sub-keywords are exploited. In other words, BabelNet indexes are queried for every (sub-)keyword, and each request may return a list of *synsets*. [1]

Synsets have various metadata attached, such as categories, domains, neighbors, or translations obtained from Wikipedia, WordNet, and other knowledge databases, and these are merged into BabelNet. Domains are the highest level in the semantic tree, whereas categories (mainly obtained from Wikipedia) are much more specific[2]. Then, three co-occurrences matrices—namely, categories per keywords, sub-keywords, and synsets—are created to avoid categories being selected when they are over-represented within a non-significant pool of synsets (e.g., when two synsets from the same keyword share the same category). Then, the Hadamard product of these three matrices is computed, and categories appearing in at least two different synsets from two distinct keywords are retained as potentially representative for the article.

[1]A *synset* can be seen as a contextualized word with a specific sense or meaning, which will not have different meanings for homonyms.

[2]There are only 26 domains, but around 290,000 categories.

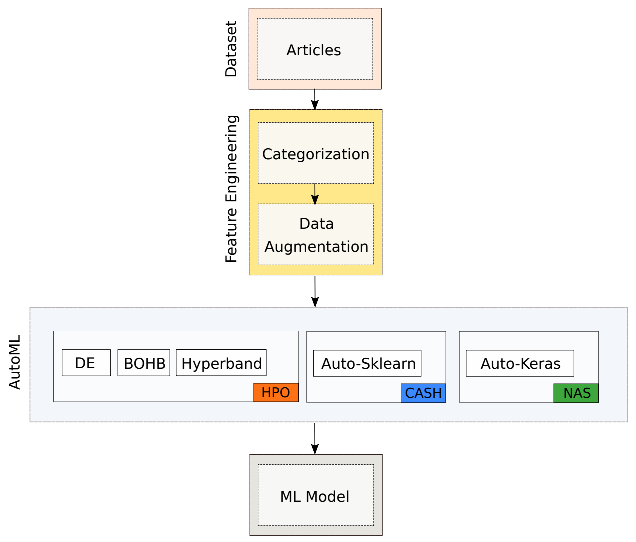

Fig. 1: Illustration of the adopted ML pipeline. First, the input data is used for feature extraction task. Then, DE, BOHB, Hyperband, Auto-Sklearn and Auto-Keras methods are applied to find a superior ML model. Accordingly, we will have five output models and the one with highest accuracy will be selected for the problem at hand.

Finally, articles are categorized and keywords are disambiguated, as illustrated in Figure 3. In this example, the article had three keywords (i.e., "AIDS", "HIV", and "Cervical cancer"). BabelNet returned, respectively, 1, 3, and 1 synset(s) while searching for these. Given that the synset from "HIV" shared categories with synsets from both other keywords, the approach validated their related synsets sharing it and the article was categorized. A weight of two (i.e., the number of keywords sharing it) was allocated to these two categories. This weight will be used for the intersection confidence (Section III-C).

B. Data Augmentation

After categorizing articles, data is augmented, and, more precisely, semantic neighbors are retrieved. Neighbors are obtained from semantic relationships obtained from Wikipedia, WordNet [21] and other sources merged into BabelNet. Hypernyms, hyponyms, homonyms and other semantically related words (i.e., other semantic relationships) are embedded into neighbors in BabelNet. However, synsets are too massively connected within the ontology tree, and some generic synsets may have a huge number of neighbors. For instance, the synset "Artificial Intelligence"[3] has 1210 neighbors (from 2212 edges) involving 5483 categories. Retaining only expected categories (i.e., the article's categories) helps to focus on more targeted neighbors and avoid retrieving generic (or unrelated) neighbors. Considering the previous example, the "Artificial intelligence" synset has eight categories in BabelNet (e.g., "Artificial intelligence", "Computational neuroscience", "Cybernetics", and "Emerging technologies") obtained from Wikipedia. Then, filtering out all neighbors which do not belong to any of its eight synset categories removes 1127 unrelated neighbors, and only 83 neighbors are finally selected.

[3]https://babelnet.org/synset?word=bn:00002150n

C. Article Connections

After these two steps, article keywords are disambiguated—at least the ones sharing categories in common—and neighbors matching categories are retrieved. Several semantic relationships can be obtained from these augmented data. Indeed, intersections and unions between the different sets may contribute to a new similarity metric, taking into account all of the following variables. Let K_x be the sets of keywords, S_x the synsets, and N_x the neighbors of the article A_x.

Keyword (K) relationships: The intersection and union cardinality of *original* keywords from both articles A_1 and A_2 (respectively, $|K_1 \bigcap K_2|$, $|K_1 \bigcup K_2|$). In other words, all keywords are taken into account, even those without synsets.

Synset (S) relationships: The intersection and union cardinality of disambiguated keywords (i.e., the selected synsets): $|S_1 \bigcap S_2|$, $|S_1 \bigcup S_2|$.

Synset intersection confidence: Given that synsets may come from highly or poorly represented categories from the article categorization, the mean weighted category value of connected synsets can represent the confidence of these in regard to their best category occurrence.

$$conf(S_1, S_2) = \sum_{s \in (S_1 \cap S_2)} \frac{1}{2} * \left(\frac{bestCat(A_1, s)}{|S_1|} + \frac{bestCat(A_2, s)}{|S_2|} \right), \tag{2}$$

where $bestCat(A_i, s)$ returns the highest category weight for s in A_i.

Synset–Neighbor (N) relationships: The cardinality of intersection and union between synsets from the first article and neighbors of the second article: $|S_1 \bigcap N_2|$, $|S_1 \bigcup N_2|$.

Neighbor–Synset relationships: The cardinality of intersection and union between neighbors from the first article (N_1) and synsets of the second article (S_2): $|N_1 \bigcap S_2|$, $|N_1 \bigcup S_2|$.

Neighbor relationships: The cardinality of intersection and union between neighbors from both articles: $|N_1 \bigcap N_2|$, $|N_1 \bigcup N_2|$.

Number of original keywords: The number of **ambiguous** keywords (i.e., authors' or computed keywords) for each article. This data might be used to determine whether the intersection per union ratio is representative of the overall number of keywords: $|K_1|$, $|K_2|$.

Number of keywords with synsets: The number of disambiguated keywords for each article. I.e., the number of keywords from which synsets sharing common categories have been found in the categorization step: $|S_1|$, $|S_2|$.

All of these variables are the features passed to a neural network—a multilayer perceptron (MLP) —and the expected labels will be the journal matching. In other words, the MLP is expected to learn the prediction of a positive similarity classification for two articles within the same (or related) journal(s).

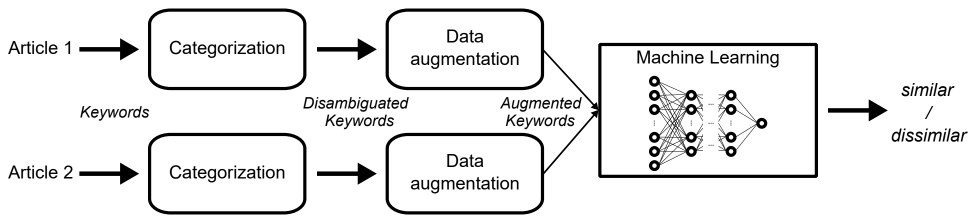

Fig. 2: General workflow of our approach. Article keywords are first categorized, augmented and a neural network learns to predict similarity of article pairs. The validation is finally based on their journal belonging.

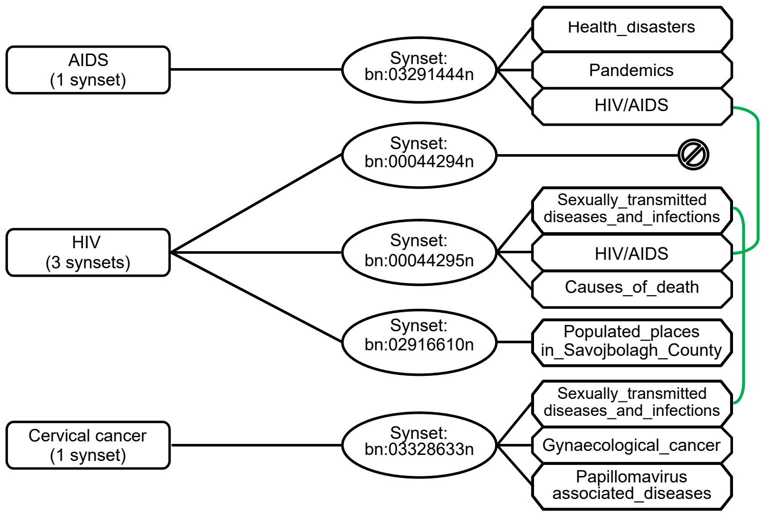

Fig. 3: Keywords' senses are disambiguated by finding common categories shared among them. Here, all three keywords share categories in common.

D. AutoML

AutoML, the end-to-end process of building ML models, can be applied to all components of the pipeline [19]. Our introduced pre-processing methods do not need to be configured and we define the performance of the pipeline as a function of its classification algorithms $\mathcal{A}$ and their hyperparameters Λ. Accordingly, the main part of our AutoML system has a significant overlap with HPO, CASH and NAS. These different problem formulations are characterized based on the complexity of the search space and search strategy.

HPO is equivalent to find an optimal hyperparameters configuration λ^* based on a given ML algorithm A and its associated hyperparameters λ with domain λ'. The search space of HPO is composed of continuous, integer, categorical, and conditional hyperparameters. From Figure 4, we can see how the values of the hyperparameters in support vector classifier (SVC) would first be treated, before the learning process begins[4] . In HPO, ML model selection and hyperparameter optimization are independent which divides the whole search space into subspaces and speeds up the optimization process.

Hyperband [22], BOHB [23] and differential evolution (DE) [24] are three HPO search strategies used in the AutoML component. Hyperband[5] is an extension of SuccessiveHalving algorithm [25] which: allocates a partial budget to a set of configurations, evaluates trained model on the full validation set, discards a set of worst configurations; enhances the budget and repeats until one configuration remains. The algorithm allocates resources to more favourable configurations as the optimization process goes on. This promising search strategy is embarrassingly parallel, but does not take into account the useful information about the fitness landscape of the problem at hand which may lead to slow convergence rate. Hence, we

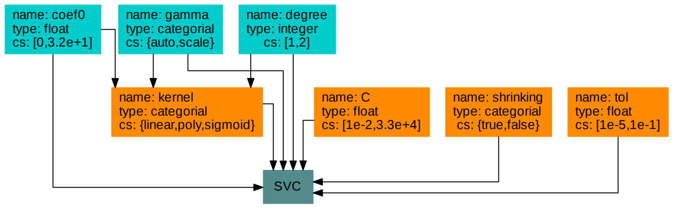

Fig. 4: An illustration of the associated hyperparameters with SVC in Sklearn software package. These hyperparameters must be chosen a priori and cannot be derived via training. The hyper-parameters in SVC have a hierarchy structure, which implies that some of them are conditioned on the value of others (denoted by blue boxes), e.g. degree of the polynomial kernel function will be ignored by all other kernels.

[4]Figure 4 is depicted based on SVC in Sklearn: https://github.com/scikit-learn/scikit-learn

[5]https://github.com/zygmuntz/hyperband

used the BOHB[6] to achieve both strong any time performance and final performance; as outlined in [23]. Moreover, an evolutionary algorithm called as DE [24], is also used to enhance our chance on the very complex search spaces of candidate configurations[7].

Alternatively, CASH formulation takes into account a mixture of the ML algorithms and their hyperparameters. Instead of selecting an algorithm and optimizing its hyperparameters, both steps are applied simultaneously. Needless to say, the search space of CASH formulation is larger because the algorithm selection itself is considered a hyperparameter. We employed Auto-sklearn [26] as the search strategy for CASH problem which is already shown to perform favorably against the state-of-the-art methods in the literature[8].

Finally, we embrace Auto-Keras which is an efficient automating architecture engineering framework for NAS [27], [28]. This approach provides functions to automatically search for architecture and hyperparameters of ML models. It employs a neural network kernel, a tree-structured acquisition function optimization algorithm, and the concept of the Network morphism to efficiently explore the search space[9]. It obtains already superior results over manually designed architectures on some tasks such as image classification and can be a potential candidate for the problem at hand.

IV. Experiments

1) Experimental data: A total of 3,112 articles from four journals (*Symmetry* (725), *Religions* (684), *Viruses* (870), and *Toxins* (833)) were selected to evaluate our approach. All journals were distinct (i.e., not in relation) except *Toxins* and *Viruses*, which have related scopes. Therefore, our approach aims to predict the similarity for pairs of articles from the same journal or for pairs of articles from similar journals. The categorization step found connections among 2,668 unique synsets from 1,389 articles (45%), from which 14,149 unique synsets were added as neighbors. Finally, the dataset contained 119,874 article pairs having at least one intersection as described in Section III-C, involving keywords, synsets, or neighbors.

2) Experimental setup: We use the power of five different optimization approaches to find a superior model for scientific article recommendation. The configuration space for HPO search strategies is composed of *DecisionTreeClassifier*, *KneighborsClassifier*, *RandomForestClassifier* and *MLPClassifier* classifiers which results in total 31 hyperparameters. Moreover, the configuration space of Auto-sklearn is composed of 15 classifiers, 14 feature selectors, and 4 data preprocessors which results in a structured hypothesis space with 110 hyperparameters. Furthermore, Auto-Keras takes into account the highly popular computational modules as the basic components to build a graph-based ML model. These modules are Dense, Dropout, and activation functions (e.g., ReLU). The readers are referred to see the detailed principle about the conducted experiments, the computational resources, the configuration space, and the reported numerical results within the supplementary file made available online[10].

For HPO-based search strategies, the number of function evaluations is limited to 550, while for Auto-Sklearn and Auto-keras it is considered to be 5 days. To tackle the negative effects of the random initial configurations, each algorithm were run 15 times. In DE, population size is 50, F is 0.9, and CR is 0.5. In Auto-Sklearn, we sampled the hyperparameters space by using functions provided in its library. We limited the maximum training time of each model to one hour and its memory consumption to 3,072 Mb. To evaluate the performance of these methods, we decomposed our dataset into a training set containing 60% of the data (i.e., pairs of articles), a validation set representing 20% of the dataset and a test set containing the remaining 20%.

3) Classifier selection: The statistical results show that DE and Hyperband search strategies obtained 90% of accuracy, followed by BOHB(0.899%), Auto-Sklearn (0.899%), and Auto-Keras (0.895%). Generally speaking, the HPO search algorithms has achieved a better performance and so more analysis are given in Figure 5 to verify this trend. Considering convergence rate, this figure shows that DE has a better performance in comparison with the other HPO algorithms. The numerical results are given in supplementary file[10] (see Appendix II). Generally, these results offer an insight into the application of different search strategies for designing a new ML-based IR systems which can be used in the future works.

V. Results

A. Overall

Experiments described in Section IV highlighted that a random forest classifier is the best classifier for our dataset. Its optimized hyperparameters are obtained as follows: n_estimators=163, criterion=gini, bootstrap=True, max_features=0.96, min_samples_split=2, min_samples_leaf=1. Hence, it has been trained on the training set and evaluated on the test set. Table I provides an overview of the article pairs grouped by their higher types of intersection found within the dataset, the size of the test set for each relationships, and the obtained accuracy for predicted pairs. Elements of $K_1 \bigcap K_2$ are considered as semantically higher than $S_1 \bigcap S_2$ ones, which are higher than elements of $S_1 \bigcap N_2$ and $N_1 \bigcap S_2$, and finally $N_1 \bigcap N_2$ are semantically the lowest type of intersections. Intersections inherited from elements of $S_1 \bigcap N_2$ and $N_1 \bigcap S_2$ are grouped because there is a huge overlap, hence it does not make sense to differentiate them.

An overall accuracy of 0.90 is obtained for the 29,975 pairwise similarity predictions from the test set. As expected, the keywords intersections are the safest ones ($K_1 \bigcap K_2$), thus the easiest to predict (0.99 of accuracy). The same observation is obtained with synsets ones ($S_1 \bigcap S_2$). The accuracy constantly decreases and predictions on intersections involving synsets and neighbors obtain an accuracy of

[6]https://github.com/automl/HpBandSter

[7]http://www1.icsi.berkeley.edu/~storn/code.html

[8]https://github.com/automl/auto-sklearn

[9]https://github.com/keras-team/autokeras

[10]https://tinyurl.com/rakhshani-wcci2020

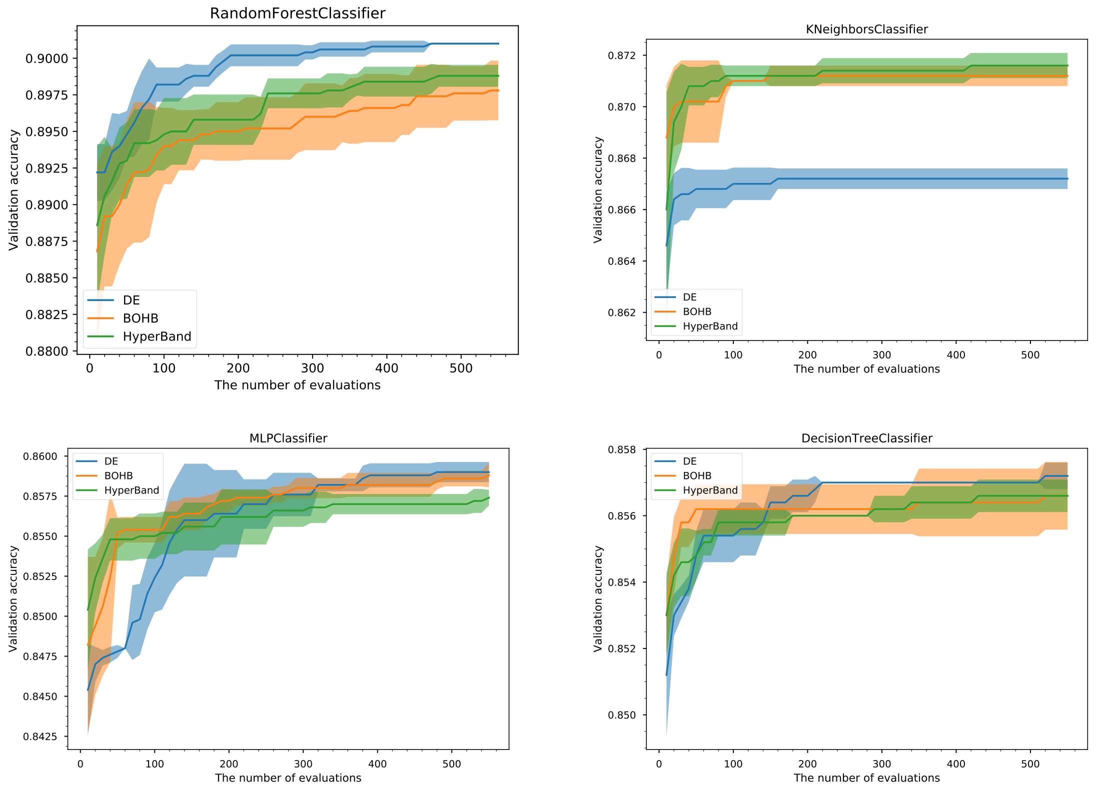

Fig. 5: The convergence plot of the DE, BOHB and Hyperband HPO search strategies over 15 runs. In this figure, the variance of the algorithms are also depicted which indicates how sensitive they are to different random seeds.

TABLE I: Distribution of article pairs according to their type of intersection in the dataset, size of the test set, and the corresponding accuracy.

Number of pairs / type	Total	$\|K_1 \bigcap K_2\|$	$\|S_1 \bigcap S_2\|$	$\|(S_1 \bigcap N_2) \bigcup (N_1 \bigcap S_2)\|$	$\|N_1 \bigcap N_2\|$
in dataset	**119,874**	4,424	4,945	29,829	80,676
in test set	**23,975**	889	981	6,024	16,081
accuracy	**0.9002**	0.9775	0.9786	0.9429	0.8751

0.94 $\Big((S_1 \bigcap N_2) \bigcup (N_1 \bigcap S_2)\Big)$. Finally, the neighbors intersections, which are the furthest semantical relationships obtain an accuracy of 0.87.

Given that neighbors intersections are the most represented pairwise elements within the dataset, their low accuracy significantly affects the overall accuracy. To counter this, we plan to further investigate on the way to improve the classification of pairs exclusively connected by neighbors. An interesting lead would be to use a sigmoid function to return a similarity score normalized between 0 and 1. Therefore, a threshold might be heuristically defined in order to fit with the requirements in terms of precision and number of articles proposed. Then, only pairs with a similarity score above this threshold will be considered as relevant.

B. *Comparison with baseline*

We proposed a baseline to estimate the added value of ML on this task. Accordingly, it computes the similarity score by realizing a weighted sum of Jaccard indexes of the sets of synsets (S_1, S_2), neighbors (N_1, N_2) and synset and neighbors $(S_1, N_2$ and $N_1, S_2)$. Keywords relationships are therefore left away and do not influence the similarity score. Without ML, it obtains a general accuracy of 0.82 on the same test set when no threshold is applied to the similarity score and every pair having a distance is considered as a prediction[11]. In other words, ML improves by 8 points the accuracy of predictions which is probably inherited by the fact that there might be other more subtle relationships between the variables which are hard seen impossible to identify by human.

VI. Conclusion

In this paper, we proposed a new architecture to predict the similarity between pairs of scientific articles. The pre-

[11]A threshold of 0.002 drops the accuracy to 0.55 given that there are many small similarity scores and too many correct predictions are skipped.

processing steps combining categorization and data augmentation built a promising semantic base, which led to the creation of a precise classifier reaching an accuracy of 0.9. In the future, we plan to compare the obtained results with other distance algorithms such as Word Mover Distance or Doc2Vec. To do so, a normalized distance might be returned from the network (i.e., before the activation function). This would provide further insights about the performance of our approach compared to other known approaches.

References

[1] T. Hey and A. Trefethen, "The Data Deluge: An e-Science Perspective," in *Communications Networking & Distributed Systems*. Wiley, 2003, pp. 809–824.

[2] X. Bai, M. Wang, I. Lee, Z. Yang, X. Kong, and F. Xia, "Scientific paper recommendation: A survey," *IEEE Access*, vol. 7, pp. 9324–9339, 2019.

[3] R. Sharma, D. Gopalani, and Y. Meena, "Concept-based approach for research paper recommendation," in *PReMI*. Springer, 2017, pp. 687–692.

[4] M. Reyhani Hamedani, S.-W. Kim, and D.-J. Kim, "SimCC: A novel method to consider both content and citations for computing similarity of scientific papers," *Information Sciences*, vol. 334-335, pp. 273–292, 2016.

[5] N. F. Matsatsinis, K. Lakiotaki, and P. Delias, "A system based on multiple criteria analysis for scientific paper recommendation," in *PCI*, 2007, pp. 135–149.

[6] S. M. McNee, I. Albert, D. Cosley, P. Gopalkrishnan, S. K. Lam, A. M. Rashid, J. A. Konstan, and J. Riedl, "On the recommending of citations for research papers," in *CSCW*. ACM, 2002, pp. 116–125.

[7] J. Beel, B. Gipp, S. Langer, and C. Breitinger, "Research-paper recommender systems: a literature survey," *International Journal on Digital Libraries*, vol. 17, no. 4, pp. 305–338, 2016.

[8] C. Manning, P. Raghavan, and H. Schütze, "Introduction to information retrieval," *Natural Language Engineering*, vol. 16, no. 1, pp. 100–103, 2010.

[9] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, p. 5, 2019.

[10] A. Nawrocka, A. Kot, and M. Nawrocki, "Application of machine learning in recommendation systems," in *ICCC*. IEEE, 2018, pp. 328–331.

[11] I. Portugal, P. Alencar, and D. Cowan, "The use of machine learning algorithms in recommender systems: A systematic review," *Expert Systems with Applications*, vol. 97, pp. 205–227, 2018.

[21] G. A. Miller, "WordNet: a lexical database for English," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[12] C.-Y. Wu, A. Ahmed, A. Beutel, A. J. Smola, and H. Jing, "Recurrent recommender networks," in *WSDM*. ACM, 2017, pp. 495–503.

[13] H. Lee and J. Lee, "Scalable deep learning-based recommendation systems," *ICT Express*, 2018.

[14] T. Ebesu and Y. Fang, "Neural citation network for context-aware citation recommendation," in *SIGIR*. ACM, 2017, pp. 1093–1096.

[15] X. Dong, L. Yu, Z. Wu, Y. Sun, L. Yuan, and F. Zhang, "A hybrid collaborative filtering model with deep structure for recommender systems," in *AAAI*, 2017.

[16] X. Cai, J. Han, and L. Yang, "Generative adversarial network based heterogeneous bibliographic network representation for personalized citation recommendation," in *AAAI*, 2018.

[17] X. Wang, L. Yu, K. Ren, G. Tao, W. Zhang, Y. Yu, and J. Wang, "Dynamic attention deep model for article recommendation by learning human editors' demonstration," in *SIGKDD*, ser. KDD '17. ACM, 2017, pp. 2051–2059.

[18] T. Bansal, D. Belanger, and A. McCallum, "Ask the gru: Multi-task learning for deep text recommendations," in *RecSys*, ser. 16. ACM, 2016, pp. 107–114.

[19] M. Zöller and M. F. Huber, "Survey on automated machine learning," *CoRR*, vol. abs/1904.12054, 2019.

[20] R. Navigli and S. P. Ponzetto, "BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network," *Artificial Intelligence*, vol. 193, pp. 217–250, 2012.

[22] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *Journal of Machine Learning Research*, vol. 18, no. 185, pp. 1–52, 2018.

[23] S. Falkner, A. Klein, and F. Hutter, "BOHB: Robust and efficient hyperparameter optimization at scale," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 10–15 Jul 2018, pp. 1437–1446.

[24] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[25] K. Jamieson and A. Talwalkar, "Non-stochastic best arm identification and hyperparameter optimization," in *Artificial Intelligence and Statistics*, 2016, pp. 240–248.

[26] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *NIPS*, 2015, pp. 2962–2970.

[27] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *Journal of Machine Learning Research*, vol. 20, no. 55, pp. 1–21, 2019.

[28] H. Jin, Q. Song, and X. Hu, "Auto-keras; an efficient neural architecture search system," in *SIGKDD*, ser. KDD '19. ACM, 2019, pp. 1946–1956.