A Graph Neural Network Approach for Evaluating Correctness of Groups of Duplicates

Michele De Bonis[1](✉), Filippo Minutella[3], Fabrizio Falchi[1], and Paolo Manghi[1,2]

[1] Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo" - ISTI (CNR), Pisa, Italy
{michele.debonis,paolo.manghi,fabrizio.falchi}@isti.cnr.it
[2] OpenAIRE AMKE, Pisa, Italy
paolo.manghi@openaire.eu
[3] TIM S.p.A., Pisa, Italy
filippo.minutella@telecomitalia.it

Abstract. Unlabeled entity deduplication is a relevant task already studied in the recent literature. Most methods can be traced back to the following workflow: entity blocking phase, in-block pairwise comparisons between entities to draw similarity relations, closure of the resulting meshes to create groups of duplicate entities, and merging group entities to remove disambiguation. Such methods are effective but still not good enough whenever a very low false positive rate is required. In this paper, we present an approach for evaluating the correctness of "groups of duplicates", which can be used to measure the group's accuracy hence its likelihood of false-positiveness. Our novel approach is based on a Graph Neural Network that exploits and combines the concept of Graph Attention and Long Short Term Memory (LSTM). The accuracy of the proposed approach is verified in the context of Author Name Disambiguation applied to a curated dataset obtained as a subset of the OpenAIRE Graph that includes PubMed publications with at least one ORCID identifier.

Keywords: entity deduplication · correctness · graph neural networks · author name disambiguation

1 Introduction

Entity deduplication (or disambiguation) refers to the process of identifying duplicates within a given collection of entity metadata descriptions. The primary objective of this process is to group the equivalent entities into distinct groups of duplicates, thereby increasing the data quality and saving storage space. The deduplication process is particularly relevant for providers who curate collections that must be indexed and made available for user search.

Before the advent of machine learning, the most popular approaches were based on a three-stage workflow:

Blocking preliminary blocking stage to group "potentially equivalent" entities to limit the number of comparisons;

Similarity match pairwise comparisons stage inside each block to check for equivalence and to draw similarity relationships between equivalent entities;

Deduplication identification of groups of duplicates (equivalent entities) by closing the meshes in the graph resulting from the previous phase.

This type of approach persists nowadays when the entities bear a well-described set of metadata attributes and the collection curator demands strong control and explainability over the results [1–3]. On the other hand, evaluating the quality (e.g. via metrics) of the groups of duplicates when these are label-less becomes as challenging as inefficient. Measuring quality is essential, to refine or improve the results, or to provide a level of confidence to the consumers of the deduplicated collection. Given the nature of the problem, where the main objective is to have a low number of wrong groups of duplicates, it is important to have an evaluation measure able to give a score to a group independently from the others. In this way, the data quality can be increased by excluding from the final graph all the groups with a low score by cutting their similarity relations.

In this paper, we address this problem by exploiting a Graph Neural Network (GNN) approach, relying on a twofold intuition: (*i*) the similarity match stage described above generates a graph where nodes represent the entities and relationships indicate the equivalence between two nodes; and (*ii*) the deduplication stage generates a set of distinct graphs, whose nodes have no relationships with nodes of other graphs. In the last few years, many different architectures involving deep learning and graphs have been proposed, with GNN methods becoming very popular in the research community. Typically, such methods encode the information in every node of the graph through a feature-extraction algorithm and subsequently generate node embeddings by encoding information about the node's topology via message passing with other nodes in the neighborhood. GNN methods have been proven effective in node and graph classification, where node embeddings are merged to represent the whole graph. Most popular examples of GNNs are the Graph Convolutional Network (GCN) [4], the Graph Attention Network (GAT) [5], Graph Isomorphism Network (GIN) [6] and Graphormer [7], which brings the concept of NLP transformers on Graphs.

In light of these observations, we propose a custom model capable of processing groups of duplicates to evaluate their correctness regarding a percentage indicator. The model is then applied in a real-case scenario of scholarly communication to duplicate author names. More specifically, it is trained in a supervised way using a custom dataset of ORCID-provided authors coming from PubMed[1] article metadata records collected by the OpenAIRE Graph[2] [8–10]. To define the model, a preliminary analysis of known Graph Neural Networks has been carried out and a 3-layered GAT was identified as the most promising. The model has been customized in two ways. Firstly, by adding edge weights that reflect the similarity match between two nodes and a node betweenness centrality [11]

[1] https://pubmed.ncbi.nlm.nih.gov.
[2] https://graph.openaire.eu.

measure that reflects the pivotal role of a node in a graph in terms of shortest paths. Secondly, by adding a further LSTM layer [12], before the classifier for the prediction. A sigmoid function is subsequently applied to the classifier output to transform the result into a percentage measure of correctness. Once the model has been trained over labeled data, it can be used to classify groups with no available labels (i.e. the majority of those resulting from a real-case scenario) as the nature of groups of duplicates depends on the algorithm used for the deduplication and therefore remains the same. Our experiments have shown the approach to be effective with both big and small groups with an accuracy of circa 90%.

The paper is organized as follows: Sect. 2 describes the current status of the research on the topic; Sect. 3 describes the methodology used to conduct the research; Sect. 4 provides a description of the developed architecture together with the experimental results; Sect. 5 discusses the obtained results; Sect. 6 concludes the paper providing some hints for the future directions of this research.

2 State of the Art

The literature does not address the problem of assessing the quality of groups of duplicate entities. However, since the groups created after a deduplication process correspond to clusters of equivalent data (entities), *clustering evaluation metrics* may be considered a valuable solution. Two classes of metrics exist: "extrinsic measures" when the ground truth label is required, and "intrinsic measures" when the ground truth label is not required. Known metrics in these fields are the *Rand Index*, the *Mutual Information*, the *V-measure*, and the *Fowlkes-Mallow score* when speaking about intrinsic measure, while the *Silhouette Coefficient*, the *Calinski-Harabasz Index*, and the *Davies-Bouldin Index* when speaking about extrinsic measure. In order to use such metrics for the evaluation it is important to think of the group of duplicates as a set of points in an n-dimensional space, and in some cases to define a measure of distance between such points. The evaluation of deduplication by means of the metrics described above does not allow the evaluation of each group independently from the others, as the final score provided by the formulas of each metric is either inefficient to be computed or descriptive of the whole deduplication.

The evaluation of a deduplication result can be sometimes performed by heavily relying on persistent identifiers of entities [13] (e.g. the DOI) but it is not guaranteed that the measure is trusty, as the persistent identifier for the evaluation is not often available for every entity in the collection of a real-case scenario.

The graph classification problem has been studied in literature and surveys on this topic summarise several methods [14,15]. The methods described in the surveys classify molecules and proteins in a supervised fashion by giving acceptable accuracy ranges. Nonetheless, such methods are not directly applicable to groups of duplicates because of the different and particular nature of such groups, having a dense or sparse distribution of relations that is not directly indicative of the correctness of the group.

As claimed by the authors, the best approach for classifying a graph is Graphormer [7], which brings the transformer concept into Graph Neural Networks. Nonetheless, the nature of transformers makes such architectures' training and inference process extremely slow or feasible only when a high computation power is available, which is not the case in most scenarios.

3 Methodology

The research of this paper has been conducted by following three steps: (*i*) the preparation of the dataset to be used as training, validation, and testing set; (*ii*) the preliminary experiments on base Graph Neural Network architecture to highlight the advantages and the disadvantages of each model; and (*iii*) the implementation, training, and validation of the final model architecture to be used for the classification of groups of duplicates.

The dataset preparation has been performed by mimicking a real-case scenario when a standard framework for deduplication has been applied. In this research we used FDup [16], a framework for efficient data deduplication using decision tree-based matching. The FDup framework delivers a full deduplication workflow in a single easy-to-use software based on Apache Spark Hadoop, where developers can customize the blocking and the similarity matching via an intuitive configuration file. In particular, the similarity matching function is engineered as a decision tree that drives the comparisons of the fields of two records as branches of predicates and allows early-exit strategies to save computation time.

The code is available on GitHub[3] and it is written in Python by using the Deep Graph Library[4] (DGL), a Pytorch-based library which implements fast and memory-efficient message passing primitives for training Graph Neural Networks. All the models have been trained using an *NVIDIA GeForce RTX 3060 Laptop GPU.*

4 Results

4.1 Dataset Preparation

Since this is aimed to be classified as a supervised way of training a Graph Neural Network for graph classification, the main objective of the research is to have a proper dataset to test the goodness of the findings. We decided to use Author Names Disambiguation as an example, and we extracted only Authors with an ORCID identifier provided by PubMed records in the OpenAIRE Graph. The identifier will be used as a label to determine the correctness of a group.

The first step of the dataset preparation consists of extracting the authors from PubMed records. Every author extracted from a publication comprises personal fields and fields inherited by the respective publication. Such fields are:

[3] https://github.com/miconis/dedup-groups-evaluator.
[4] https://www.dgl.ai.

– full name (i.e. *"Surname, Name"* or *"Surname, N."*)
– the co-authors list (i.e. the list of authors belonging to the same publication as a list of strings)
– the abstract of the publication

Since the deduplication algorithm needs meaningful information for the comparison, the abstract of the publications has been processed with the Latent Dirichlet Allocation (LDA) [17]. To this aim, the abstracts of the publications have been tokenized (i.e. transformed into vectors of words excluding the stop-words), and vectorized (by means of a Bag of Word model) using the Dewey Decimal Classification [18] as a dictionary. Various models of LDA have been trained over the vectorized abstracts by varying the number of topics. The perplexity score over the testing set has been fined-tuned to reach the optimal number of topics for the collection, which resulted in 15, as depicted in Fig. 1.

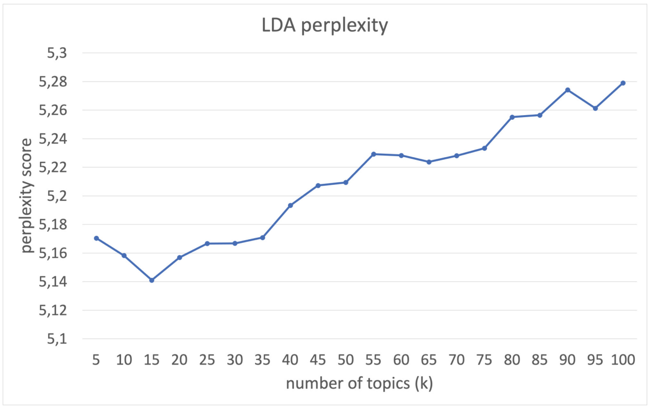

Fig. 1. LDA perplexity score varying the number of topics

Once the optimal LDA model in terms of perplexity is obtained, every abstract has been processed to produce a 15-sized topic vector assigned to the publication's author names to describe authors in terms of the topics they touched upon. The collection of enriched author names is deduplicated at this stage by applying FDup. The framework has been configured as follows:

– preliminary Last Name First Initial (LNFI) blocking stage to identify potentially equivalent authors as authors sharing the surname and the first letter of the first name; in particular, authors having the same surname and the same initial letter of the name are considered potentially equivalent and therefore processed by the similarity matching function (i.e. *"Sandra Smith"* and *"Steven Smith"* will end up in the same block as they share the same blocking key - *"smiths"*);

– pairwise similarity matching based on comparing the co-authors lists and LDA topic vectors. The similarity on the co-authors' lists is measured by counting the number of common names among the lists (i.e. number of similar names), while the cosine similarity measures the similarity on the topic vectors. Note that the threshold of the co-authors similarity has been set empirically to 2, while the threshold of the topic vectors similarity has been set to 0.5 after a False Positive - False Negative analysis varying the threshold on all the possible comparisons. The FDup decision tree used for this purpose is depicted in Fig. 2.

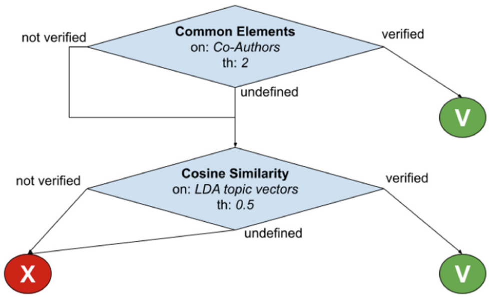

Fig. 2. FDup decision tree used for authors' disambiguation

The result of the deduplication creates groups of authors sharing at least 2 co-authors and/or having a cosine similarity of their topic vectors greater than 0.5. Such groups have been processed and prepared to be the proper training set for the Graph Neural Network. To this aim, they have been labeled and manually classified into positive groups (i.e. all the authors in the group have the same ORCID), or negative groups (i.e. the group has authors with different ORCIDs). Subsequently, 2-sized groups have been removed (i.e. they are pairs), and the dataset has been balanced to have the same number of positive and negative samples. Statistics on the groups are reported in Table 1. The dataset used for this research is available on Zenodo.org [19]. Note that the total number of positive and negative groups has been balanced, but the dataset reflects the common situation in real-case scenarios where the number of wrong groups increases with the size of the groups.

4.2 Preliminary Experiments

In order to provide meaningful features to the Neural Networks, the abstract associated with each author has been further processed via a pre-trained BERT Sentence Embedding model called *bert-base-multilingual-cased*. BERT [20] is a method of language representation able to extract high-quality language features from text data, guaranteeing that similar texts produce similar embeddings.

Table 1. Training dataset statistics

	positive	negative
global	25,450	25,450
groups of 3	12,291	6,699
groups of 4 to 10	11,882	12,107
groups of more than 10	1,277	6,644
total	50,900	

Once the BERT embedding for each graph node has been created, the dataset has been divided into training, validation, and testing set with a ratio of 60%, 20%, and 20%. The idea is to exploit the message passing to update node embeddings with topology information coming from the neighborhood and consequently apply a readout (e.g. aggregation of node embeddings) to have the final graph embedding classified (as usual in graph classification tasks). We decided to test 3 base architectures using the most popular GNN layers. Such architectures are described below:

- GCN3: a 3-layered Graph Convolutional Network;
- GAT3: a 3-layered Graph Attention Network;
- Graphormer: a 6-layered graph transformer with Spacial Encoding and Degree Encoding, as described in the original paper [7].

In every case, the network is finalized by a Linear transformation layer inputted to a sigmoid function to obtain the value as a percentage to be used as a score for evaluating a group. Each architecture has been trained and tested to stop the training process once the overfitting condition was verified (i.e. the loss on the testing set increasing for more than 20 epochs). We chose the best model for each architecture by taking the one from the epoch with the lower loss. Subsequently, models have been evaluated by measuring their performances on the validation set.

Results depicted in Table 2 showed the GAT3 model to be the most promising approach for this kind of activity, confirming the outcomes of the literature claiming that putting attention on neighborhoods' features brings better results.

Table 2. Preliminary experiments on base Graph Neural Networks

model	Acc	TPR	TNR	FPR	FNR	Precision	F1-Score
Graphormer	75.91	85.02	66.56	33.43	14.97	72.29	78.14
GCN3	78.76	81.63	75.81	24.18	18.36	77.59	79.59
GAT3	81.73	87.16	76.17	23.82	12.83	78.96	82.86

4.3 Final Model Architecture

Once the best base model has been pointed out, the final architecture to be used for the purpose of this research has been developed. The intuition behind the approach is in the intrinsic characteristics that make a group of duplicates wrong.

The first source of errors is in the clustering key that defines initial blocks to limit the number of comparisons. In other words, each node in the graph must include in its encoding also some sort of encoding for the field used for the blocking. Since the strategy adopted for the purpose of this experiment is to apply the LNFI on the authors' names, the encoding of each node should include also an encoding for author names. The type of encoding used for this purpose is a Bag of Letters-like method, a simplifying representation that imitates the most common Bag of Words representation used in natural language processing and information retrieval. Each author name is coded in a 55-sized feature vector in which each element indicates the frequency of a specific letter in the name (the size of 55 indicates the number of characters in the alphabet used as a dictionary). Such kind of encoding is sufficient to achieve good results since it guarantees a good representation of typos, which may be present in author names but are still coded in similar vectors (in case of typing errors leading to letters swapped in positions, the encoding is exactly the same). An example of how an author name is encoded is shown in Fig. 3. To better describe the differences among the names in the group, each edge has been weighted with the Jaro-Winkler distance between the two names. This way each edge is normalized with the degree of similarity of the names of its nodes.

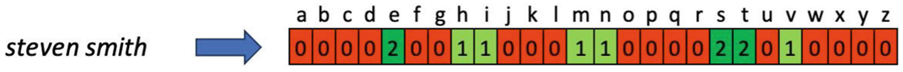

Fig. 3. Simplified example of the author name encoding

The second source of error lies in one (or more) nodes leading to the creation of bigger groups. Such nodes have been identified as "bridges" because they are usually poorly described nodes (with a missing first name, and missing co-authors) matching with nodes belonging to different groups for their intrinsic characteristics, creating bigger groups putting together different entities. Examples of bridges are depicted in Fig. 4, where authors with missing first names matched with authors with two different names resulting in the creation of a big wrong group of duplicates after the closure of the meshes.

In order to emphasize such nodes, we developed a centrality encoder that gives a higher weight to nodes which can be potentially a bridge. For this purpose, we used the betweenness centrality measure, which detects the influence a node has over the flow of information in a graph. It is often used to find nodes that serve as a bridge from one part of a graph to another because it measures how

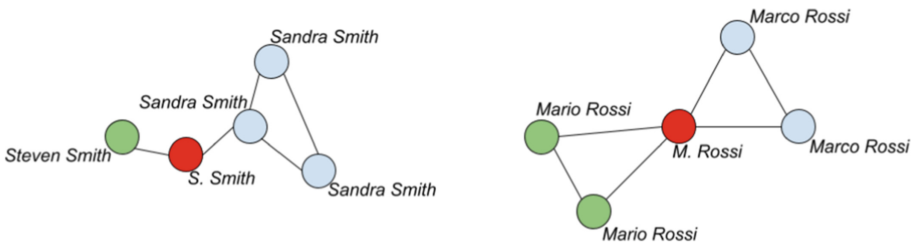

Fig. 4. Example of bridges in a disambiguation graph

many times, in proportion, the path needs to pass from a specific node to reach another.

To complete the model, we added an LSTM to process the output of the GAT convolutional layers. This intuition comes from the fact that groups in the dataset are not of the same length and a smaller group may be flattened by passing through a high number of convolutional layers. On the contrary, bigger groups must pass through more layers for better representation. In this economy, the concatenation of the outcome of every GAT convolutional layer is inputted to the LSTM which will be able to learn to which extent to consider the results of the first layers of the network (meaningful for small groups) combining them with results of the last layer of the network (meaningful for big groups). Following the previous description, our final architecture is depicted in Fig. 5.

Table 3 reports the results obtained for the testing set using the newly created model, dividing them on the nature of the block to better describe how the model behaves. It is shown as the model has an accuracy of about 90% on each class of groups.

Table 3. Experiments on the final architecture

model	Acc	TPR	TNR	FPR	FNR	Precision	F1-Score
GAT3NamesEdgesCentrality	89.87	93.03	86.62	13.37	6.96	87.71	90.29
(in groups of 3)	88.56	95.05	76.75	23.24	4.94	88.14	91.46
(in groups of 4 to 10)	88.77	91.48	85.98	14.01	8.59	87.08	89.22
(in groups of more than 10)	96.25	88.64	97.81	2.18	11.35	89.29	88.97

5 Discussion

Usually, a deduplication process ends up with a series of groups with different sizes: smaller groups are the most probable while bigger groups are less. Conversely, the number of wrong groups among bigger groups is higher because

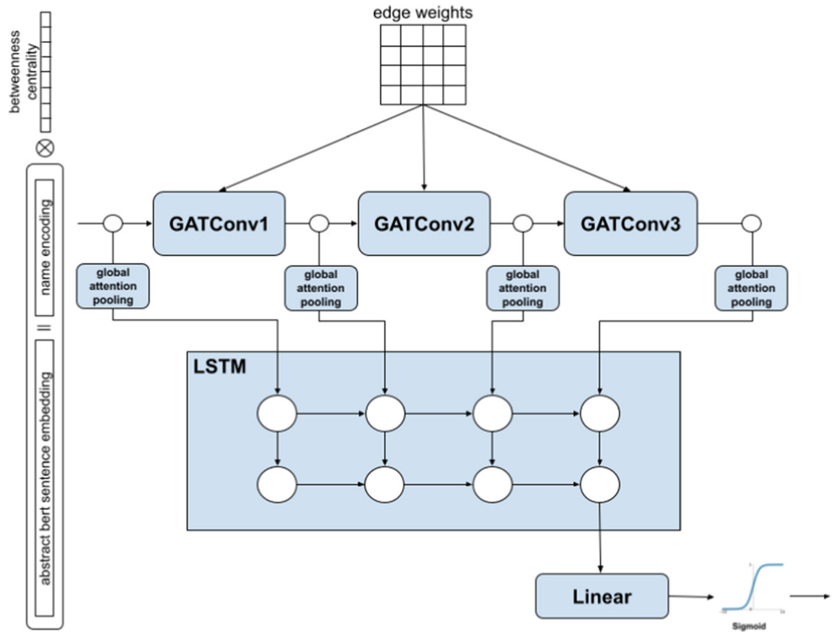

Fig. 5. Final architecture for the classification of a group of duplicates

finding bridges on a more extensive set of nodes is easier. The dataset we created for this research perfectly reflects the environment described above, as table numbers suggest a coherent distribution among wrong and correct groups. Note that the accuracy of the process is not directly comparable to any other architecture of the literature as emerged that the use case was not studied in other research.

The results in terms of accuracy and other metrics depicted in tables for each model architecture tested in this research showed that the main lack of base approach was the misalignment of the accuracies on groups of different sizes. Smaller groups tend to bring down the whole accuracy because the information of the first layers is lost as the other layers of the network process the input. Adding the LSTM at the end of the network allows for overcoming this limitation as it considers meaningful information coming from previous processing steps when needed. In fact, the results of the final model with the LSTM show a balanced accuracy between smaller groups, leading to a higher average accuracy among all the groups in the dataset.

It is important to notice that the accuracy of groups with more than 10 entities is very promising, as such groups are the most difficult to be individuated. The percentages of True Positives, True Negatives, False Positives, and False Negatives fit with the use case, as in this kind of activities is important not to have False Negatives which tend to bring the quality of the data to a lower level.

A fixed threshold has been set on the network output for training. Such threshold identifies a correct group when the output score is greater than 0.5, while identifies a wrong group when the output score is lower than 0.5. A threshold analysis on the scores of the False Negatives and False Positives allows the fine-tuning of such threshold to reduce the number of errors. Increasing the threshold on the final score increases the model's overall accuracy. The outcome of the Graph Neural Network can be consequently used to correct wrong groups (i.e. those with a low score) and to promote correct groups (i.e. those with a high score).

In the end, it is important to mention that the approach is meant to work also for other types of entities since the correctness of a group depends on attributes of the same nature (e.g. titles when the deduplication is performed over publications, legal names when the deduplication is performed over organizations, etc.). The concept of the bridge remains the same in every scenario and does not depend on the type of entity to be deduplicated.

6 Conclusions and Future Works

In this paper, we presented a Graph Neural Network based on the Graph Attention mechanism and the Long Short Term Memory to classify groups of duplicates created by a standard deduplication algorithm. The architecture is provided with a custom encoding for the betweenness centrality of each node and a Bag Of Letters model for the name of the author encoding. The experiments performed on the custom dataset created for this purpose showed acceptable measures of accuracy considering the typical use case of the deduplication activity.

Accuracy can be further increased by including in the encodings entity attributes used by the deduplication algorithm in charge of performing the pairwise comparisons, as experiments suggested that the source of errors lies in poorly described fields.

The approach described in this paper uses the Author Name Disambiguation as an example use case but it is possible to turn it into a general purpose approach. The "bridge" problem depends not on the entity type being deduplicated but on the 3-stage entity linking paradigm formed by entity blocking, pairwise matching, and closing meshes. To turn the approach into a general purpose, it is sufficient to act on the feature type used to feed the Graph Neural Network in a way that they describe the entity attribute responsible for the equivalence of a pair of entities. The initial ground truth to train the Graph Neural Network with can be created by performing the deduplication on entities with identifiers (e.g. ORCID) to be used for the labeling of groups. Once the network has been trained over the ground truth, it can be used to evaluate the correctness of groups even when they do not contain entities with an identifier.

Acknowledgements. This work was partially funded by the EC H2020 projects OpenAIRE-Nexus (Grant agreement 101017452) and FAIRCORE4EOSC (Grant agreement 101057264).

References

1. Manghi, P., Atzori, C., De Bonis, M., Bardi, A.: Entity deduplication in big data graphs for scholarly communication. Data Technol. Appl. **54**(4), 409–435 (2020)
2. He, Q., Li, Z., Zhang, X.: Data deduplication techniques. In: 2010 International Conference on Future Information Technology and Management Engineering, vol. 1, pp. 430–433. IEEE (2010)
3. Kolb, L., Thor, A., Rahm, E.: Dedoop: efficient deduplication with hadoop. Proc. VLDB Endow. **5**(12), 1878–1881 (2012). https://doi.org/10.14778/2367502.2367527
4. Zhang, S., Tong, H., Xu, J., Maciejewski, R.: Graph convolutional networks: a comprehensive review. Comput. Soc. Netw. **6**(1), 1–23 (2019). https://doi.org/10.1186/s40649-019-0069-y
5. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
6. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? arXiv preprint arXiv:1810.00826 (2018)
7. Ying, C., et al.: Do transformers really perform badly for graph representation? Adv. Neural Inf. Process. Syst. **34**, 28877–28888 (2021)
8. Manghi, P., Houssos, N., Mikulicic, M., Jörg, B.: The data model of the OpenAIRE scientific communication e-infrastructure. In: Dodero, J.M., Palomo-Duarte, M., Karampiperis, P. (eds.) MTSR 2012. CCIS, vol. 343, pp. 168–180. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35233-1_18
9. Manghi, P., et al.: The openaire research graph data model. Zenodo (2019)
10. Manghi, P., et al.: Openaire research graph dump (2022)
11. Ausiello, G., Firmani, D., Laura, L.: The (betweenness) centrality of critical nodes and network cores, pp. 90–95, July 2013
12. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**, 1735–1780 (1997)
13. Vichos, K., et al.: A preliminary assessment of the article deduplication algorithm used for the openaire research graph (2022)
14. Errica, F., Podda, M., Bacciu, D., Micheli, A.: A fair comparison of graph neural networks for graph classification. CoRR abs/1912.09893 (2019). http://arxiv.org/abs/1912.09893
15. Tsuda, K., Saigo, H.: Graph classification, pp. 337–363. Springer, US, Boston, MA (2010). https://doi.org/10.1007/978-1-4419-6045-0_11
16. De Bonis, M., Manghi, P., Atzori, C.: FDup: a framework for general-purpose and efficient entity deduplication of record collections. PeerJ. Comput. Sci. **8**, e1058 (2022)
17. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
18. Scott, M.L., SCOTT, M.L.: Dewey Decimal Classification. Libraries Unlimited (1998)
19. De Bonis, M.: Deduplication groups evaluator data benchmark. https://doi.org/10.5281/zenodo.7997279, June 2023
20. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. CoRR abs/1810.04805 (2018). http://arxiv.org/abs/1810.04805