The VLDB Journal (2024) 33:981–1011
https://doi.org/10.1007/s00778-023-00822-z

SPECIAL ISSUE PAPER

ALFA: active learning for graph neural network-based semantic schema alignment

Venkata Vamsikrishna Meduri[1] · Abdul Quamar[2] · Chuan Lei[3] · Xiao Qin[3] · Berthold Reinwald[1]

Received: 30 January 2023 / Revised: 20 July 2023 / Accepted: 17 October 2023 / Published online: 21 November 2023

Abstract
Semantic schema alignment aims to match elements across a pair of schemas based on their semantic representation. It is a key primitive for data integration that facilitates the creation of a common data fabric across heterogeneous data sources. Deep learning approaches such as graph representation learning have shown promise for effective alignment of semantically rich schemas, often captured as ontologies. Most of these approaches are supervised and require large amounts of labeled training data, which is expensive in terms of cost and manual labor. Active learning (AL) techniques can alleviate this issue by intelligently choosing the data to be labeled utilizing a human-in-the-loop approach, while minimizing the amount of labeled training data required. However, existing active learning techniques are limited in their ability to utilize the rich semantic information from underlying schemas. Therefore, they cannot drive effective and efficient sample selection for human labeling that is necessary to scale to larger datasets. In this paper, we propose ALFA, an active learning framework to overcome these limitations. ALFA exploits the schema element properties as well as the relationships between schema elements (structure) to drive a novel ontology-aware sample selection and label propagation algorithm for training highly accurate alignment models. We propose semantic blocking to scale to larger datasets without compromising model quality. Our experimental results across three real-world datasets show that (1) ALFA leads to a substantial reduction (27–82%) in the cost of human labeling, (2) semantic blocking reduces label skew up to 40× without adversely affecting model quality and scales AL to large datasets, and (3) sample selection achieves comparable schema matching quality (90% F1-score) to models trained on the entire set of available training data. We also show that ALFA outperforms the state-of-the-art ontology alignment system, BERTMap, in terms of (1) 10× shorter time per AL iteration and (2) requiring half of the AL iterations to achieve the highest convergent F1-score.

Keywords Semantic schema alignment · Active learning · Data integration

1 Introduction

Semantic schema alignment that finds matching elements across a pair of schemas based on their semantic representation forms a key step towards data integration. The semantic representation of the schema elements, often captured as an ontology, associates these elements to entities in the real world by capturing their properties and structural relationships w.r.t. the other elements in the schema. Figure 1 shows an example semantic schema alignment between two schemas, *CMT* and *Conference* represented as ontologies. The green arrows show schema alignments as matching element pairs across the two schemas, e.g., *Author* ↔ *Participant*, *Document* ↔ *Contribution*, etc.

In the context of semantic schema alignment which matches and integrates schemas represented as ontology

A. Quamar, C. Lei, X. Qin: Work done while at IBM Research.

✉ Venkata Vamsikrishna Meduri
vamsi.meduri@ibm.com

Abdul Quamar
abdulquamar@google.com

Chuan Lei
chuanlei@amazon.com

Xiao Qin
drxqin@amazon.com

Berthold Reinwald
reinwald@us.ibm.com

1 IBM Research - Almaden, San Jose, USA

2 Google, Mountain View, USA

3 AWS AI Research and Education, San Jose, USA

graphs, prior art can broadly be categorized into rule discovery and machine learning (ML)-based solutions. Earlier works on rule discovery for ontology alignment such as AML [18] and LogMap [13] predominantly rely on the lexical similarity between the concepts. Their capability to capture the ontology structure is limited to concept hierarchies. During the rule discovery, these methods require a significant amount of human effort or manual intervention.

ML-based solutions such as graph neural network (GNN)-based techniques have been shown to be effective for semantic schema alignment [5, 25, 32] as they can capture the semantic representation of the schema elements such as their properties, descriptions and relationships to other schema elements in the vector space. However, most GNN-based techniques are supervised and require labeled data to train effective models for schema alignment. Providing labeled data for training entails a significant amount of effort from subject matter experts (SMEs), which is very expensive w.r.t. both cost and manual labor.

With recent advancements in large language models, ontology alignment solutions such as BERTMap [27] try to learn contextual representations based on fine-tuning BERT [17] on ontological text for alignment. BERTMap can support both unsupervised and semi-supervised settings by leveraging synonym and antonym discovery heuristics to generate training labels required to fine-tune language models. The generated training data needs to be diverse and representative of the underlying alignment task to train effective models. This would involve SMEs to manually look at the schemas and identify matching and non-matching pairs of entities across two schemas and provide them as positive and negative samples for model training. The problem gets further exacerbated with multiple large schemas to be integrated into a unified semantic schema.

Active learning (AL) alleviates this problem with a human-in-the-loop approach that provides labeled data incrementally and on-demand to train a model. The goal is to get the highest return in terms of model performance while minimizing the amount of manual labeling effort. AL pipelines typically employ (1) sample selection techniques to choose representative and informative samples for human labeling, (2) label propagation as an optional optimization to propagate the training labels obtained from the human to other unlabeled samples which are similar to the labeled samples, and (3) blocking also as an optional optimization to prune away non-ambiguous samples of data and scale the process of sample selection to large datasets.

Sample selection methods (e.g., entropy-based sample selection [2, 43, 60] and Query-by-Committee [40]) mostly rely on model performance to drive sample selection. Importance weighted sampling [7, 38] selects samples that minimize the sampling bias and are representative of the true underlying data distribution. Other techniques such as

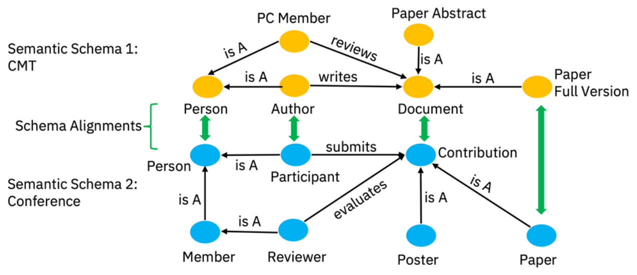


Fig. 1 Schema alignment of *CMT* and *Conference*

gradient [57] and error-based sample selection [24] are computationally expensive and hence fail to scale to large datasets while maintaining interactive sample selection times. There also exist graph-aware sample selection techniques for link prediction between two graph nodes [6, 47]. These graph-aware techniques mostly rely on aggregating structural properties such as degree and centrality sum. However, they do not exploit the semantics of the relationships between the nodes in the graph for sample selection. Similar to sample selection, label propagation [52] and blocking [69, 70] techniques are either model dependent or based on string similarity heuristics, which are devoid of any meaningful semantics capable of relating schema elements to real-world entities and relationships.

In this paper, we propose ALFA, a novel active learning framework to address the aforementioned limitations of existing AL techniques for semantic schema alignment. The key idea of ALFA is to exploit the rich semantic information from the underlying schemas to drive the process of AL. We use a GNN to capture the semantic representation of the elements that includes properties such as names, descriptions as well as relationships with other elements in the schema. We propose a novel ontology-aware sample selection algorithm that minimizes human labeling cost by choosing samples of schema elements across a pair of schemas based on their likelihood of being mis-classified by the GNN model. To further reduce human effort in labeling training data, we develop a novel ontology-aware label propagation algorithm that utilizes human-labeled schema element pairs and propagates their labels to semantically similar pairs of schema elements. Finally, to scale ALFA to large schemas and to handle the issue of class imbalance (label skew) in the labeled training data, we propose a semantic blocking technique that prunes away pairs of schema elements that are unlikely matches based on their semantic representation. To the best of our knowledge, ALFA is the first to address the problem of AL for GNN-based semantic schema alignment where schemas are represented as ontologies.

We conduct an extensive evaluation of our proposed solution on three real-world datasets against representative baselines. Specifically, we choose a state-of-the-art GNN-based schema alignment model [25], multiple AL baselines and an end-to-end semi-supervised ontology alignment solution, BERTMap [27]. Note that we purposely utilize a lightweight GNN model (i.e., RGCN [62]) in ALFA with a modest learning rate as well as a limited number of layers and training epochs. This ensures that the user waiting time is are low when ALFA is used with a human oracle in the AL loop.

Our experimental evaluations on three real-world datasets show that (1) ALFA leads to a substantial reduction (27–82%) in the cost of human labeling, (2) semantic blocking reduces label skew up to semantic blocking reduces label skew up to 40× without adversely affecting model quality and scales AL to large datasets, and (3) sample selection achieves comparable schema matching quality (90% F1-score) to models trained on the entire set of available training data. We show that ALFA outperforms a state-of-the-art ontology alignment system, BERTMap [27], in terms of (1) 10× shorter time per AL iteration and (2) requiring half of the AL iterations to achieve the highest convergent F1-score.

Our main contributions are outlined as follows:

- An end-to-end active learning framework for GNN-based semantic schema alignment.
- A novel ontology-aware sample selection algorithm for human labeling that exploits the semantic representation of the schema elements to minimize human labeling cost.
- An efficient ontology-aware label propagation algorithm that propagates labels based on their semantic representation to further reduce the cost of labeling training data.
- An effective semantic blocking method that prunes likely mis-matches between schema elements to scale to larger schemas, to reduce the sample selection latency without sacrificing the model quality.
- Extensive experimental evaluation of ALFA against state-of-the-art baselines over real-world data sets.

2 Related work

2.1 Schema matching

Relational schema matching Schema alignment or matching is a critical step in data integration for downstream tasks such as entity matching. For simple relational schemas, columns are aligned by domain experts manually [35, 41, 72] or semi-automatically [28]. Heuristic-based approaches for schema matching [4, 54, 65] use column similarity information such as data type, structure (if schema is a graph), linguistic and constraint similarity, and other transformation-based standardization techniques that learn rules or regular expressions. These approaches lack generalizability when the *test* pairs do not adhere to the same patterns as the *training* pairs, and cannot scale to large ontology graphs. Gal et al. [64] and Shraga et al. [21] are more recent works that combine deep learning with heuristic-based approaches to improve generalization to new schemas. However, all these approaches are still confined to the relational representation of the schema.

Logical reasoning-based schema matching Semantic schema (ontology) matching solutions such as AML [18, 71] and LogMap [13, 30] rely on lexical matching between the schema elements combined with simple hierarchical information in the ontology to discover matches. LogMap derives matching rules in Horn Logic from the discovered matches and it requires logical inconsistency discovery mechanism and human assistance to iteratively repair and discover new rules. Recently, MEDTO[25] and BERTMap [27] utilize a supervised GNN or language models for schema matching, outperforming AML and LogMap. However, these approaches either require a lot of training data (for MEDTO) or fine-tuning effort (for semi-supervised BERTMap). Supervised GNNs have also been applied to the task of knowledge graph alignment [74] but they too are plagued by the need for diverse and large training sets.

Active learning for schema matching Cate et al. [66] alleviate the need for a lot of training data during schema matching using active learning. However, they require matching and non-matching relational instances as training data to learn rules for schema matching. Such instance data is unavailable for several real-world ontologies [44, 45] thus making [66] inapplicable to semantic schema matching.

2.2 Active learning

Generic active learning Generic AL techniques include entropy-based selection [2, 43, 60], Query by Committee (QBC) [40], importance weighted sampling [7, 38], gradient-based [57] and error-based selection [24]. Among these selection strategies, gradient and error-based techniques re-train the classifier on each unlabeled pair to compute their *ambiguity* (i.e., the likelihood of being mis-classified), thereby rendering them not scalable in practical settings. Additionally, these AL techniques, if used out-of-the-box, will be ineffective in capturing the underlying semantic information available in the ontology graphs. The reason is that most of the generic AL strategies select the ambiguous concept pairs simply based on the model performance and are ontology-agnostic.

Active learning for GNNs Cai et al. [9], Gao et al. [23] use the GNNs trained from the initial AL iterations to approximate the final embedding space. However, these approaches may suffer from low approximation accuracy since the graph models obtained from the initial training rounds may not be accurate due to the insufficient training labels at the beginning. Wu et al. [73] and Zhang et al. [76] propose a decoupled GCN for approximating the final embedding space. The idea is to construct a simplified GCN model by removing the trainable parameters from its message passing phase. The approximated node embeddings are generated by iteratively aggregating their k-hops neighbors without any node-wise transformations. However, it only works for the GCN model and cannot be easily extended to a wide range of graph embedding methods. ALFA, on the other hand, is extensible and not dependent on a specific graph embedding approach. Also, most existing AL solutions for GNNs mainly focus on node classification over a single graph input.

Active learning for link prediction AL techniques have also been developed for link prediction over social networks and knowledge graphs (KGs). AL for link prediction [6] selects those pairs which have the highest structural significance in the graph. To avoid generating the pool of all possible node pairs, Cesa Bianchi et al. [12] partition the graph into several spanning trees and only query the labels for cross-tree edges. Ostapuk et al. [47] maximize the diversity or stratification of the selected pairs in a KG by clustering the triples and by selecting representative uncertain triples from diverse clusters or strata. Cheng et al. [15] assume a strict 1:1 correspondence between users from two social media platforms, called the *anchor node* assumption. So if a pair of users is a match, neither of these users can have a match with any other user profile across these platforms. Such an anchor node assumption does not hold in semantic schema (ontology) matching. Overall, a majority of these graph-aware AL techniques for link prediction mostly relies on aggregate structural properties such as degree or centrality sum without exploiting the semantics (or meaning) of the relationships represented by the graph edges for deriving the appropriate network representation.

Active learning for EM using MLPs Existing work on entity matching (EM) [33, 39] applied active learning to fully-connected feed-forward neural networks such as the multi-layer perceptrons (MLP). Concretely, Meduri et al. [39] represent a pair of tuples to be matched as a vector of similarities computed over the attribute pairs in those tuples. These feature vectors are then fed to an MLP with a sigmoid function as the output layer. If the sigmoid function emits a probability close to 0.5, this indicates that the MLP is unsure about the label of this pair of tuples, which should be assigned to an oracle for labeling. This approach is not attribute-invariant and hence cannot be applied in domain adaptation scenarios where an EM model trained on one label-abundant domain needs to be adapted to a label-scarce domain using transfer learning.

Kasai et al. [33] avoid using pre-determined similarity functions to represent tuple pairs. Sequence models such as bi-directional GRUs along with fastText embeddings [8] are used to learn attribute representations, and they are made attribute-invariant by aggregating individual attribute embeddings to generate tuple-level embeddings. Such feature representations can also be generated by auto-encoders and LSTMs [67].

2.3 Blocking

Blocking is a generic optimization technique commonly used to reduce the search space in entity matching and other related tasks. Jaccard similarity-based blocking has been extensively used in the EM literature [39]. It prunes the obvious non-matching entity pairs in the search space by employing a Jaccard similarity threshold. Rule-based blocking for active learning [50, 51] employs conjunctive blocking predicates that are explicitly defined by a human and are incorporated into the matching rules learned in disjunctive normal form (rule DNFs). Deep learning (DL)-based blocking techniques [67] have been proposed to overcome the need for human-defined blocking predicates. They divide the blocking task into two steps—(a) representation learning and (b) vector pairing. Representation learning is used to learn embeddings for the tuples to be aligned in EM tasks. Vector pairing applies locality sensitive hashing (LSH), nearest neighbor techniques with a top-k parameter or similarity functions such as cosine or Jaccard with a similarity threshold. However, DL-based solutions such as AutoBlock [77] require labeled pairs to learn embedding vectors for blocking. To overcome this limitation, Jain et al. [29] use an ensemble of pre-trained transformer models with nearest neighbor indexes from the FAISS library [31] that are iteratively refined by oracle-supplied labels.

In ALFA, we leverage a semantic blocking strategy for ontology alignment that relies on a pre-trained language model to generate embeddings and subsequently applies clustering using Euclidean distance as the nearest neighbor detector to find post-blocking pairs. Our blocking is currently utilized as a pre-processing step outside the AL loop. Integrating blocking into ALFA's AL pipeline is left for future work.

3 Preliminaries and system overview

In this section, we first describe how semantic schema alignment differs from relational schema alignment and entity alignment in knowledge graphs. We then introduce a GNN-based supervised model for schema alignment where the schemas are represented as ontologies. Lastly, we briefly describe the active learning techniques and terminologies followed by ALFA system overview.

3.1 Relational versus semantic schema alignment

Relational schema alignment focuses on aligning the structure of relational database schemas. A relational schema contains the metadata of tables and columns which specifies the column names, their data types and constraints such as functional dependencies, check constraints, primary and foreign key constraints connecting the tables at the metadata level. The goal of relational schema alignment is to map corresponding tables and columns from different databases.

Semantic schema alignment goes beyond the relational schema alignment and focuses on aligning the underlying meaning and semantics of data. It involves understanding the real-world entities, their attributes, and the relationships between them, which are often represented as an ontology. The correspondence between an ontology and its schema counterpart is not always direct. Depending on how an ontology is created, a concept in the ontology could refer to a column in the relational schema, or occasionally be mapped to a table with its columns appearing as the data properties of this concept [25]. The semantic information, such as connected concepts and the labeled edges between them, in the form of neighborhood information from an ontology, is crucial to semantic schema alignment or ontology alignment [13, 18, 25].

3.2 Entity alignment in knowledge graph versus semantic schema alignment

Entity alignment in knowledge graph (KG) (a.k.a KG alignment) aims to find equivalence relations between entities in different knowledge graphs which semantically represent the same real-world object. The entities in a KG are often represented as triples in a semi-structured format. The key distinction between semantic schema (or an ontology) and KG is that an ontology concept represents a generalization of entities stored in a KG. For example, the semantic schema shown in Fig. 1 refers to entities such as *Author*, *Document*, *Reviewer* as abstract/general concepts whereas KG contains the actual names or titles of these concepts. In other words, semantic schema matching is a meta-level matching task whereas entity alignment in KG can be seen as an instance-level matching task.

Zhang et al. [75] survey entity alignment in KG using representation learning and highlight translation-based and GNN-based embedding models both of which capture neighborhood of an entity either in a triple or at a broader level in the knowledge graph. As mentioned in [75], most KG alignment methods and the corresponding benchmark datasets make the anchor node assumption (a.k.a the *bijection* problem). Specifically, the ground truth in entity alignment datasets often has a 1:1 correspondence where an entity from one knowledge graph only matches to a single entity from

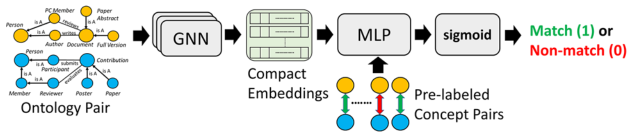


Fig. 2 GNN-based semantic schema alignment

another KG. In this work, we neither assume 1:1 correspondence between schemas nor observe the *bijection* problem in the benchmark datasets used for semantic schema matching.

Moreover, an ontology is generally well-curated and corresponds to the schemas of relational databases. On the other hand, KGs are often extracted from semi-structured data such as web pages and unstructured data sources, and they are generally incomplete and are affected by the open-world assumption. In this work, we assume that the ontology is complete and accurate. We match ontologies for the purpose of creating a unified ontology that can serve as a unified semantic schema.

3.3 GNN-based semantic schema alignment

Fig. 2 shows the architecture of a GNN-based semantic schema alignment model (e.g., MEDTO[25]) to find matching elements across two input schemas. It utilizes a GNN to generate a low-dimensional representation (compact embedding) for each node (i.e., a schema element) in the two input schemas. It then uses a classifier such as a Multi-Layer Perceptron (MLP) with a sigmoid output layer to classify a given pair of elements across the two input schemas as a match or non-match. The GNN-based semantic schema alignment model takes the two ontology graphs representing the schemas to be matched, a set of initial feature vectors for each schema element (concept) in the ontology graph and a training set of labeled matching and non-matching ontology concept pairs as input. The initial set of feature vectors is typically generated from the schema element properties such as their label names and descriptions using a pre-trained language model.

We first pre-process the labels and the textual description (if available) of the schema elements. We tokenize them using the NLTK word tokenizer [42]. We remove the stop-words, special characters such as punctuation and arithmetic symbols from the tokens. Then, we concatenate the pre-processed label and description tokens separated by a whitespace and feed the resulting text into a pre-trained language model. In this paper we choose Universal Sentence Encoder (USE) [11], which generates the embeddings as the initial semantic representations of schema elements.

The GNN model takes the schema element properties as well as their structure (relationships with other schema elements) into account while generating a semantically rich representation of each schema element as a compact embedding. The alignment model also distinguishes between the different kinds of relationships among the schema elements such as *is-A* or hierarchical relationships, unions, and other functional relationships such as *writes* and *reviews* as shown in Fig. 1. A GNN typically has a graph encoder-decoder architecture [25, 34, 54, 62] that learns the compact embeddings for each node in a graph based on its local neighboring nodes, the labeled edges connecting the node to its neighbors and the initial semantic representations of the node. RGCN [62] is a representative relational graph convolutional network that captures both type and direction of an edge into a node's local neighborhood during the node embedding generation and has shown promising results for link prediction on knowledge graphs. Hence we choose RGCN to propagate information from neighboring nodes to a target node while generating its embedding. The objective is to allow nodes with similar neighborhoods to be closer to each other in the embedding space. For details about how the message-passing framework is designed in RGCNs to generate the node embeddings, we refer the reader to [62].

Finally, we use a multi-layer perceptron (MLP) and a sigmoid function to classify pairs of concept nodes as match or mis-match for ontology alignment. To train this classifier, labeled pairs of concepts along with their expected labels are fed to the model as input. For the matching probability at the sigmoid function to match the expected class label, we use the binary cross entropy as the loss function, which helps in backpropagating the training loss through both the MLP and the GNN layers. This results in adjusting the weights of both the MLP as well as the GNN, thereby refining not only the prediction accuracy of the MLP, but also enhancing the node representativeness of the compact embeddings. This is because we are simultaneously training both the GNN and the MLP based on a single loss function in a stacked manner. After several epochs, GNN training converges when the expected labels match the predicted labels, and we also get refined compact embeddings for the nodes in the ontology. During prediction, given an unlabeled pair of concepts, RGCN takes the initial USE embeddings of each concept node as input and generates the compact embeddings for each concept based on its ontology neighborhood. Next, the MLP takes the embeddings for the pair of concepts as input and predicts the matching probability for the pair at the (sigmoid) layer. The pair is deemed as a match if the matching probability is above 0.5.

Learning such GNN-based schema alignment model in a supervised manner requires a large amount of labeled training data consisting of matching and non-matching pairs of schema elements. During the training process, the losses based on the classification labels are back-propagated to learn the appropriate node embeddings of the schema graphs. During the prediction phase, the model generates the embeddings and uses them as input to the classifier to identify matching or non-matching pairs of schema elements. Active Learning (AL) can significantly reduce the amount of labeled data required to train such models.

3.4 Generic active learning framework

A general AL framework enables an iterative human-in-the-loop process wherein a model is iteratively trained on data labeled by a human or oracle at a given cost. The iterative training process stops when the desired matching quality of the model is achieved or when the labeling budget is exhausted. The key components of an AL framework are described briefly below. Note that sample selection is a mandatory component in an AL framework whereas label propagation and blocking are optimization techniques that can be optionally deployed.

3.4.1 Sample selection

At the heart of a typical active learning framework lies a smart sample selection technique that chooses informative samples from the underlying data distribution for human (or oracle) labeling in each AL iteration. The target is to learn an effective model with the minimum amount of labeled training data in the fewest possible AL iterations. This is achieved by choosing samples which will majorly influence the model based on one or more factors such as their representativeness of the underlying data distribution, associated uncertainty of model prediction, expected effect on model learning, etc.

3.4.2 Label propagation

To further optimize the return on investment and to reduce the cost of human labeling, label propagation can be used to propagate the training labels obtained from the human (or oracle) in each AL iteration to other unlabeled training data based on the similarity of the unlabeled data to the oracle-labeled data. A variety of different techniques and similarity metrics can be used for label propagation based on the type of training data being used and the model being trained. The choice of this metric and its effective implementation affect the quality of label propagation and hence have a direct bearing on the performance of the model being trained. Label propagation is alternatively termed as *mapping extension* [27], *weak supervision* [52] and *label spreading* [14] in prior art.

3.4.3 Blocking

Unlike sample selection and label propagation which are applied in each AL iteration, blocking is a pruning step that is

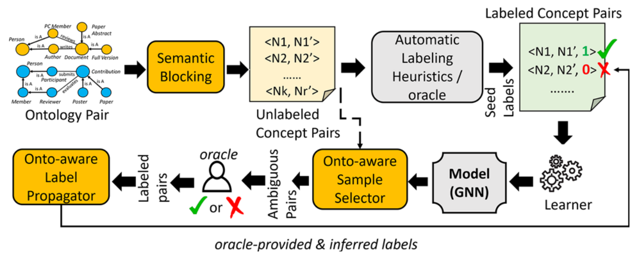


Fig. 3 ALFA: system architecture

typically applied once before commencing active learning.[1] To scale the process of sample selection to large datasets, blocking techniques are used to prune away obvious non-ambiguous samples from the majority class which is typically the non-matching (or the negative label) class. This results in a reduced search space of candidate samples depending on the level of aggression with which blocking is applied. Additionally, blocking is also used to control the class imbalance (or label skew) to train effective models efficiently. Blocking helps achieve interactive sample selection times over large datasets, making the AL pipeline suitable for the inclusion of a human-in-the-loop to perform the labeling task. On the other hand, blocking is also prone to pruning away the ambiguous samples from the minority class (i.e., the positive label class containing all the matching pairs) which could have benefited from human labeling. The trade-off thus is between scalability and the desired classification quality of the model.

3.5 ALFA system overview

Give a pair of semantic schemas (ontologies) O_L and O_R, a human oracle H, a supervised GNN-based semantic schema alignment model M, and a labeling budget B, our goal is to design an active learning framework that queries H for the minimum number of informative training labels L such that $|L| \leq B$ and the re-trained version of M predicts the equivalent schema element pairs across O_L and O_R with a high accuracy. Figure 3 shows the architecture of ALFA, our proposed AL framework, that exploits the rich semantic information of the underlying schemas represented as ontologies combined with model confidence for smart sample selection, label propagation and blocking. Collectively, they ensure effective use of the labeling budget by learning a high-quality schema alignment model in fewer AL iterations.

ALFA consumes an ontology pair to be matched and generates the Cartesian Product of all possible schema element (concept) pairs as the pool of unlabeled examples, which can be quite extensive for large ontologies. A *semantic blocking algorithm* prunes the obvious non-matches to reduce the search space for sample selection and class imbalance (Sect. 4.3). A small seed set, typically 0.1–0.3% of the post-blocking pairs, is manually labeled (with the assistance of automatic labeling heuristics or labeling functions [55] if necessary) and fed to a learner to train an initial GNN-based schema alignment model. The actual sizes of the seed label sets are between 5 and 40 across all our experimental datasets. This bootstrapping operation is required because we use a supervised GNN model that needs to be initialized before applying AL.

In each AL iteration, an *ontology-aware sample selector* combines the rich semantic information from the input schemas with the model output to choose a batch of *ambiguous* samples for human labeling (Sect. 4.1). The batch size is set based on the #labels the human oracle prefers to label per AL iteration and the maximum #iterations possible with a pre-constrained labeling budget. To further reduce the human labeling effort, we design *ontology-aware label propagation* which identifies concept pairs that are semantically similar to the pairs labeled by the human and infers the labels for such pairs. The pairs labeled by the human and the pairs whose labels are inferred through label propagation are together included as additional training data into the existing training set. The model is re-trained on the cumulative set of labeled concept pairs at the end of each AL iteration.

4 ALFA system design

In this section, we describe the main building blocks of ALFA. We first describe the core component which is our ontology-aware sample selection, followed by our optimizations i.e., ontology-aware label propagation and semantic blocking in detail.

4.1 Ontology-aware sample selection

Our ontology-aware sample selection algorithm chooses *ambiguous* pairs of schema elements that are likely to be mis-classified (i.e., a matching pair being mis-predicted to be non-matching and vice-versa) and passes them for human labeling. The likely mis-predictions are detected based on the labeling disagreement between the trained model and an ontology clustering algorithm that clusters ontology concepts (i.e., schema elements) in the unified ontology graph. The unified ontology graph combines both the input ontologies into a single graph. Note that both the model and the clustering are iteratively updated, thereby resulting in the detection of an updated set of ambiguous samples in each AL iteration. Our sample selector does not explicitly control the class skew or the ratio of matching and non-matching pairs in each AL iteration. The class imbalance issue is

[1] Note that variants of blocking which are applied in each AL iteration exist in the entity matching literature [29].

resolved by our semantic blocking optimization (details are in Sect. 4.3) which is applied before AL commences. However, it was empirically observed that the ambiguous samples comprised concept pairs from both the classes (*matching* and *non-matching*) over several AL iterations.

Figure 4 illustrates our ontology-aware sample selection. As we have earlier mentioned in Sect. 3.3, we use RGCN as the GNN model that is trained over several active learning iterations. In each active learning iteration, the model is trained using the current set of labeled training pairs. The binary cross entropy loss computed at the output of the Multi-layer Perceptron (MLP) classifier based on the model output and the ground truth provided by the labeled training pairs is backpropagated to update the model. The compact node embeddings produced by the RGCN model in each AL iteration are fed to (1) an MLP classifier that predicts the labels of the remaining unlabeled pairs and (2) a K-means clustering algorithm [36, 37] that clusters the ontology concepts from the remaining set of unlabeled pairs, based on the Euclidean distance[2] between their embeddings.

Each candidate unlabeled pair in green dashed circles shown in Fig. 4, is scored using (1) the RGCN model + MLP classifier (shown as the green arrow output in Fig. 4) and (2) the clustering model. If both the ontology concepts in the candidate pair lie in the same cluster, they are likely to represent the same semantic concept and hence can be considered a match. However, if the RGCN+MLP-based alignment model predicts the same pair as a non-match, there is a disagreement implying a likelihood of mis-classification, namely Likely False Negative (LFN). Similarly if a pair of concepts lies across two clusters, they likely represent different semantic concepts and if the RGCN+MLP-based model predicts the same pair as a match, this disagreement implies a likelihood of mis-classification, namely Likely False Positive (LFP). The *disagreement* between the classification and the clustering models in essence quantifies the ambiguity, which makes the pair an ideal candidate for human labeling.

Algorithm 1 shows the details of the ontology-aware sample selection algorithm in each AL iteration. It takes as input (1) the two ontologies (Ont_L and Ont_R) representing the schemas, (2) the remaining pool of unlabeled candidate pairs of nodes across the two ontologies, $P_{remaining}$, to choose the samples from, (3) the *batchSize* indicating the number of samples required to be selected for labeling in an AL iteration, (4) a reference to the GNN-based alignment model, and (5) the number of clusters $n_{cluster}$ for ontology clustering. Line 1 creates a set of input nodes from two input ontologies. Line 2 computes the ontology clusters based on the RGCN model embeddings using K-means clustering and the ontology clusters are updated in each AL iteration. Lines 4–10 compute the disagreement *score* for each remaining pair based on the RGCN-based schema alignment model prediction probability, *PredProb*. If the nodes in a given pair belong to the same ontology cluster (indicating a match as per the ontology clustering), then 1.0—*model.PredProb* reflects the quantum of disagreement. A high model prediction probability indicates a predicted match and vice-versa. Similarly, if the nodes belong to different clusters (indicating a mis-match as per the ontology clustering), the model prediction probability provides the disagreement score. Finally, lines 11–12 sort all the unlabeled pairs based on the disagreement score in a descending order and choose the top-k pairs for human labeling. The algorithm also returns the generated clusters along with the top-k pairs in line 13 that are used as input by our label propagation algorithm.

Algorithm 1 ontoAwareSelection(Ont_L, Ont_R, $P_{remaining}$, $batchSize$, $model$, $n_{cluster}$)

```
1: $P_{sel} = \{\}$, $InputNodes \leftarrow Nodes_{Ont_L} \cup Nodes_{Ont}$
2: $Clusters \leftarrow$ ontologyClustering($InputNodes$, $n_{cluster}$,
   $model.Emb$)
3: $scores \leftarrow []$
4: for $j: 0$ to $|P_{remaining}| - 1$ do
5:     if nodes in a given pair $P_{remaining}[j]$ belong to the same cluster
       then
6:         $scores[j] \leftarrow 1.0 - model.PredProb(P_{remaining}[j])$
7:     else
8:         $scores[j] \leftarrow model.PredProb(P_{remaining}[j])$
9:     end if
10: end for
11: $sortedPairs \leftarrow$ Sort $P_{remaining}$ DESC on $scores$
12: $P_{sel} \leftarrow$ choose top-$k$ pairs with the highest score from $sortedPairs$
    where $k = \min(batchSize, |sortedPairs|)$
13: return $Clusters$, $P_{sel}$
```

Labeling disagreement is also illustrated in Fig. 4. For a given pair of nodes, *Document* from the *CMT* ontology and *Poster* from the *Conference* ontology, the model predicts a low similarity score of 0.25, whereas the ontology clustering algorithm based on Euclidean distance predicts that *Document* and *Poster* are in the same cluster. The pair receives a high ambiguity score of 0.75 (i.e., 1.0—*model.PredProb*) and is sent to the human oracle who labels it as a match. Upon re-training the model, the model adjusts its prediction probability to 0.75 for this pair thereby accurately classifying it.

Note that both ontology clustering and model prediction are based on the RGCN model embeddings. Therefore, the model quality and the cluster quality improve with more AL iterations as the embeddings are refined. Given that clustering is iteratively applied on model embeddings corresponding to concepts belonging to the remaining unlabeled pairs, the produced clusters are non-homogeneous and large in the initial AL iterations and shrink in the later iterations as the remaining unlabeled pairs become fewer.

2 Scikit library uses Euclidean distance by default for K-means clustering, replacing which by other metrics does not bring a significant difference in clustering quality.

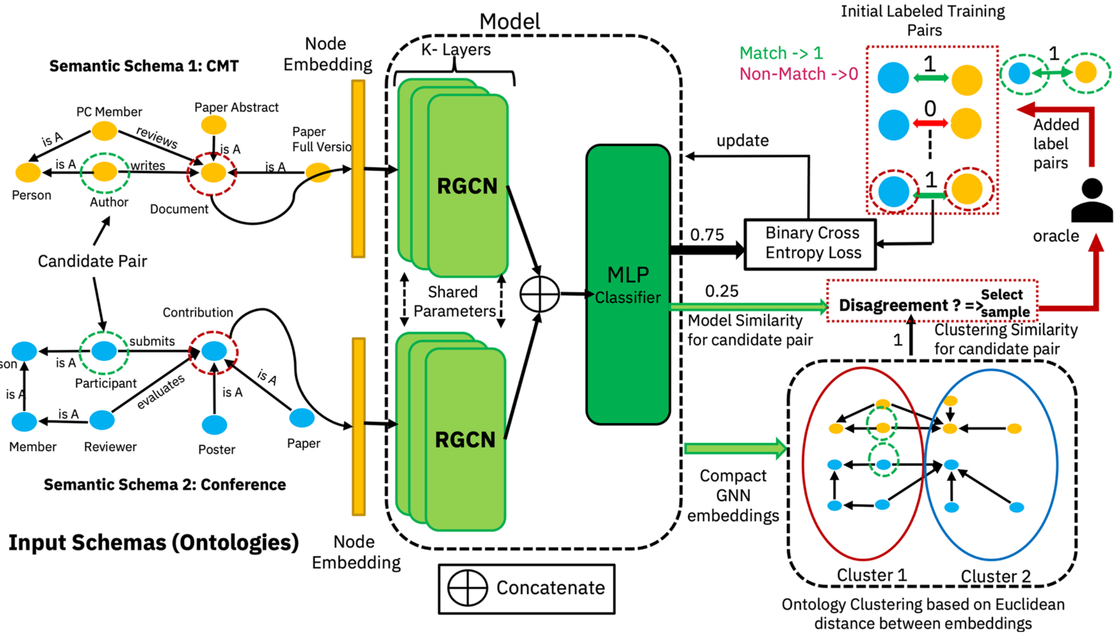


Fig. 4 Ontology-aware sample selection

Another key insight here is that each method tries to capture the real underlying data distribution differently. While the ontology clustering uses the Euclidean distance between the node embeddings as the similarity metric to form clusters of similar nodes, the schema alignment model uses a trained neural network, i.e., an MLP with a sigmoid output layer to determine the similarity between two embeddings. Hence, labeling disagreement between the ontology clustering and the neural network captures the ambiguity in the way the actual distribution is modeled by these two approaches. Thus, a pair with such disagreement is a good candidate for human labeling.

Note that we use the labeling disagreement as a heuristic and find it conceptually similar to the notion of variance computation in query-by-committee (QBC), which was proven to be effective in earlier works [16, 20, 63]. QBC can iteratively reduce the *version space*, i.e., the candidate space of classifiers that can correctly classify the training labels. This iterative reduction in the version space was shown to quickly converge active learning. Additionally, it is worth noting that we do not use a committee of several supervised learning models of the same kind. Instead, our committee comprises an unsupervised clustering algorithm and a supervised GNN model. Clustering employs the Euclidean distance metric and assigns equal weight to each dimension in the GNN-generated embeddings. On the other hand, the MLP is data-driven and learns the appropriate weight for each dimension based on the embeddings and their expected labels. This ensures that both models capture different signals for ontology alignment, which makes ALFA's label disagreement computation informative and effective.

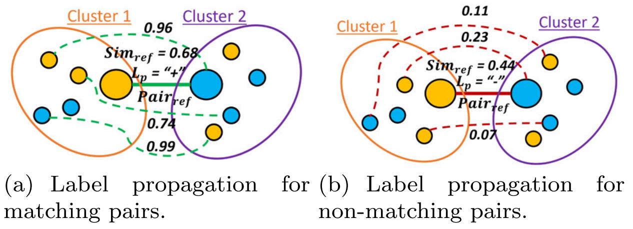


Fig. 5 Ontology-aware label propagation in ALFA

4.2 Ontology-aware label propagation

To further reduce human effort in labeling training data, we propose a novel ontology-aware label propagation algorithm that utilizes the schema element (node) pairs labeled by the human and propagates their labels to semantically similar pairs of schema elements across the two input ontologies.

Figure 5 shows how we propagate the label L_P assigned by a human for a specific pair $Pair_{ref}$ to several other unlabeled samples (concept pairs). We do so by selecting the unlabeled pairs U which are semantically most similar to $Pair_{ref}$ and hence can borrow the same label L_P. For each pair $Pair_{ref}$ (of nodes across the two ontologies) labeled by a human, we first compute the cosine similarity, Sim_{ref}, between the embeddings of the nodes ($Pair_{ref.left}$, $Pair_{ref.right}$) within that pair. We then identify the clusters that each node in the pair $Pair_{ref}$ belongs to, in order to find other nodes that are similar to the nodes in $Pair_{ref}$. We then compute the Cartesian product of all possible pairs of nodes across the two identified node clusters, that belong to two different ontologies provided as input. Note that if both the nodes in the pair belong to the same cluster, the algorithm chooses Cartesian product of all possible cross-ontology pairs within the same cluster. We mark these as the pool of candidate pairs for label propagation. Further, we handle the propagation of *matching* (+) and *non-matching* (-) labels provided by the human to $Pair_{ref}$ as two separate cases.

Case 1: matching pair This case handles the propagation of a *matching* label L_P assigned by a human to $Pair_{ref}$. All pairs within the pool of candidate pairs whose cosine similarity between the node embeddings exceeds Sim_{ref}, are assigned the matching label L_P. The example in Fig. 5a shows $Pair_{ref}$ as the two nodes across Clusters 1 and 2 connected with a solid green line. The qualifying pairs, whose node embedding similarity exceeds Sim_{ref} (0.68), are shown connected with dashed green edges. The intuition here is that if a human labels a node pair as a match, then the candidate pairs similar to the labeled pair (having their nodes belonging to the same clusters as the ones in $Pair_{ref}$) and whose constituent node similarity is greater than that of the labeled pair could borrow the same label.

Case 2: non-matching pair This case handles the propagation of a *non-matching* label L_P assigned by a human to $Pair_{ref}$. All pairs within the pool of candidate pairs whose cosine similarity between the node embeddings is below Sim_{ref}, are assigned a non-matching label L_P. Symmetrically, Fig. 5b shows $Pair_{ref}$ as the two nodes across cluster 1 and 2 connected with a solid red line that is labeled as a non-match by the human. The pairs connected by the red dashed lines having their node embedding similarity below Sim_{ref} (0.44) borrow the negative label from $Pair_{ref}$.

Having determined the methodology for label propagation, the next step is to determine the quantum of label propagation in each AL iteration that would be sufficient to achieve the intended reduction in human labeling effort while also maintaining the desired level of accuracy. ALFA therefore provides a flexible mechanism to control the trade-off between the reduction in human labeling cost and model quality (F1-score) using three different modes of propagation.

Mode 1: unrestricted In this mode of label propagation, the human-provided label for each reference pair, $Pair_{ref}$, is propagated without any restrictions to all eligible concept pairs based on the method described in cases 1 and 2 above. This is the most aggressive form of label propagation and provides the maximum amount of reduction in human labeling effort at the cost of achieving a lower model quality.

Mode 2: conservative In this mode, the human-provided label for $Pair_{ref}$ is propagated more conservatively to top-k unlabeled pairs which are semantically similar[3] to $Pair_{ref}$ and are ranked by their constituent node embedding similarities. For instance, k could be 1, in which case, a *matching* label will be propagated to one additional unlabeled pair that is semantically similar to the pair labeled by the oracle and has the highest cosine similarity between its constituent node embeddings. On the other hand, if the pair labeled by the oracle has a *non-matching* label, it will be propagated to the semantically similar pair with the least constituent node embedding similarity. This mode allows for the most fine-grained control over the amount of label propagation and the value of k could be chosen to suit the available human labeling budget. Note that we set k to 1 in our experiments for conservative mode. This is because the label propagation happens for each reference pair labeled by the oracle, i.e., if 20 pairs are labeled by oracle in an AL iteration, conservative mode infers the labels for 20 more pairs. Propagating to top-3 or top-5 pairs results in $3\times$ to $5\times$ more labels in each AL iteration which was empirically found to be aggressive in nature.

Mode 3: adaptive This mode allows for propagating a human-provided label adaptively to a varying number of unlabeled samples in each AL iteration. The key idea is that label propagation is dependent on the quality of clustering which is done based on the model-generated embeddings. In the initial AL iterations, the model is still not mature and hence label propagation is done less aggressively to avoid sacrificing accuracy by incorrect label propagation. As the model becomes more accurate, the clustering is also more refined and hence the labels are propagated more aggressively without sacrificing on model accuracy. In our current implementation, we propagate the label of $Pair_{ref}$ to top-k unlabeled pairs which are semantically similar to $Pair_{ref}$ and are ranked by their constituent node embedding similarities, but with an additional constraint that k is chosen to be the numerical value of the current AL iteration. Hence, the adaptive mode can be considered as a variant of the conservative mode with a dynamically changing value of k that reflects the increasing confidence in the model as it is refined. This mode adaptively balances the trade-off between the cost of human labeling and model accuracy.

We provide a detailed empirical evaluation of the above mentioned trade-off for these modes of label propagation in Sect. 5.3. By default, we use the conservative mode of label propagation in our end-to-end evaluation of ALFA. We discuss how to choose the label propagation mode in Sect. 5.5.

[3] Semantically similar means that the constituent nodes have the same cluster belongingness as the nodes in the labeled pair. *Ranking* by node similarity covers the constraint that the constituent node similarity should be larger or smaller than the node similarity of the labeled pair depending on *matching* or *non-matching* label being propagated.

Algorithm 2 ontoAwareLabelPropagation($P_{remaining}$, $P_{labeled}$, $Clusters$, $labelPropMode$, $iter$)

```
1: $P_{inferred} = \{\}$
2: for $i$: 0 to $|P_{labeled}| - 1$ do
3:     $P_{shortlisted} = \{\}$
4:     $Pair_{ref} \leftarrow P_{labeled}[i]$
5:     $candPairs \leftarrow genCandidatePairs(Pair_{ref}.left.cluster,$
       $Pair_{ref}.right.cluster, Pair_{ref})$
6:     if $labelPropMode$ == "unrestricted" then
7:         $P_{shortlisted} \leftarrow candPairs$
8:     else if $labelPropMode$ == "conservative" then
9:         $P_{shortlisted} \leftarrow topk(k = 1, candPairs, Pair_{ref})$
10:    else if $labelPropMode$ == "adaptive" then
11:        $P_{shortlisted} \leftarrow topk(k = iter, candPairs, Pair_{ref})$
12:    end if
13:    $P_{shortlisted}.label = Pair_{ref}.label$
14:    $P_{inferred} \leftarrow P_{inferred} \cup P_{shortlisted}$
15: end for
16: return $P_{inferred}$
```

Algorithm 3 genCandidatePairs($lCluster$, $rCluster$, $Pair_{ref}$)

```
1: $candPairs = \{\}$
2: $Sim_{ref} \leftarrow cosine\_sim(Pair_{ref}.left.emb, Pair_{ref}.right.emb)$
3: for $i$: 0 to $|lCluster.nodes|$-1 do
4:     for $j$: 0 to $|rCluster.nodes|$-1 do
5:         $Pair_{cand} \leftarrow (lCluster.nodes[i], rCluster.nodes[j])$
6:         if $Pair_{cand} \neq Pair_{ref}$ && $Pair_{cand}.left.onto \neq$
           $Pair_{cand}.right.onto$ then
7:             $Sim_{cand} \leftarrow cosine\_sim(Pair_{cand}.left.emb,$
               $Pair_{cand}.right.emb)$
8:             if $(Pair_{ref}.label == matching$ && $Sim_{cand} \geq Sim_{ref})$ or
               $(Pair_{ref}.label == nonMatching$ && $Sim_{cand} \leq Sim_{ref})$
               then
9:                 $candPairs \leftarrow candPairs \cup Pair_{cand}$
10:            end if
11:        end if
12:    end for
13: end for
14: return $candPairs$
```

Algorithm 2 describes our proposed ontology-aware label propagation algorithm. It takes as input a set of remaining unlabeled node pairs ($P_{remaining}$), a set of human labeled node pairs in the current AL iteration ($P_{labeled}$), a set of clusters ($Clusters$) produced by the ontology clustering algorithm, the label propagation mode *labelPropMode* and the numerical value of the current AL iteration, $iter$. Lines 2 to 15 iterate over each labeled node pair, $Pair_{ref}$, and compute the set of node pairs, $P_{inferred}$, to which the label of $Pair_{ref}$ is propagated. Lines 6 to 12 determine which node pairs among the candidate pairs, $candPairs$, need to be shortlisted in each AL iteration as $P_{shortlisted}$ for inclusion into the final set of pairs, $P_{inferred}$, with propagated labels. If the mode of label propagation is "unrestricted", all the candidate pairs are shortlisted (line 7). In the case of "conservative" mode, the top-k pairs (where k=1) which are semantically similar to $Pair_{ref}$ and ranked by their constituent embedding similarities are shortlisted (line 8). If the mode is "adaptive", k is set to the numerical value of the AL iteration, $iter$, and the top-k pairs are shortlisted (line 9). Line 13 indicates that each pair among the set of shortlisted pairs, $P_{shortlisted}$, gets the same label as P_{ref}. Line 14 includes $P_{shortlisted}$ into the set of pairs with propagated labels, $P_{inferred}$ which is returned in line 16.

Algorithm 3 describes how we generate the candidate pool of unlabeled pairs for each reference pair, $Pair_{ref}$. The algorithm takes $Pair_{ref}$ and the cluster belongingness of the left and right nodes within $Pair_{ref}$ as input parameters. As mentioned before, it is possible that the left and right clusters are the same. We enumerate each candidate pair, $Pair_{cand}$, across the clusters and check if the left and right nodes within $Pair_{cand}$ belong to different ontologies (line 6). For each cross-ontology pair, we propagate the $matching$ and $non\text{-}matching$ labels separately in line 8 based on the embedding similarities, Sim_{ref} and Sim_{cand}, computed in lines 2 and 7 respectively. We include $Pair_{cand}$ into the set of candidate pairs $candPairs$ in line 9 only if the criteria are met for **Case 1: matching pair** and **Case 2: non-matching pair** as described earlier. Finally the set of candidate pairs is returned in line 14.

4.3 Semantic blocking

We propose a semantic blocking technique that prunes away pairs of schema elements that are unlikely matches based on their semantic representation. This reduces the search space of sample selection thereby allowing ALFA to scale to larger schemas. Additionally, it also reduces label class imbalance between the matching and non-matching pairs thus enabling the training of more accurate alignment models efficiently.

Existing techniques for blocking such as those based on the Jaccard similarity metric [39, 40, 69, 70] are dependent on pure string matching and are unable to fully capture the semantic similarity of the schema elements. As a result, this may lead to a lot of false negatives, namely pruning away a number of matching pairs thereby adversely affecting model accuracy. To overcome this limitation, we propose an unsupervised semantic blocking technique which prunes the obvious non-matching schema elements based on their semantic representation to reduce the number of false negatives. This semantic representation is created using pre-trained language models such as USE [10] or BERT [17] to transform the schema element properties such as names and descriptions into fixed size low-dimensional vectors.

In this section, we discuss two variants of USE-based semantic blocking which we empirically compare against Jaccard-based and BERT-based blocking baselines in Sect. 5.4. BERT-based blocking has been evaluated as a deep learning-based blocking candidate for entity matching [67] and recently used by a state-of-the-art ontology alignment system called BERTMap [27].

USESim In this variant, we compute the cosine similarity sim_{USE} between the USE embeddings of the schema elements in each concept pair. If sim_{USE} is lower than a pre-determined similarity threshold parameter τ_{sim}, the pair is pruned away. Despite parallelizing USESim, it has high latency as it enumerates the entire search space of all possible pairs in the Cartesian Product. Therefore, we propose an efficient blocking variant called USECluster.

USECluster The schema elements in the two input schemas are clustered based on the Euclidean distance between these embeddings. The number of clusters is a parameter that allows the system to achieve a pre-specified target level of blocking in terms of number of post-blocking pairs. The blocking algorithm prunes away all the schema element pairs where the individual elements in the pair lie across different clusters indicating a semantic mismatch.

Algorithm 4 describes our USESim variant of semantic blocking which takes as input the two ontologies representing the schemas Ont_L, Ont_R, and the similarity threshold τ_{sim}. Lines 3 to 9 enumerate the entire search space of Cartesian Product between the left and right ontologies, Ont_L and Ont_R. For each candidate pair, P_{cand}, the USE embedding similarity value, Sim_{USE}, between the constituent nodes is computed in line 5 and is compared against τ_{sim} in line 6. Line 7 adds the candidate pair $Pair_{cand}$ to the set of post-blocking pairs if it qualifies. Finally, line 10 returns the set of post-blocking pairs, $P_{remaining}$.

Algorithm 4 USESim(Ont_L, Ont_R, τ_{sim})

```
1: $P_{cartesian} \leftarrow Ont_L \times Ont_R$
2: $P_{remaining} = \{\}$, $Ont \leftarrow Ont_L \cup Ont_R$
3: for $i$: 0 to $|P_{cartesian}|$-1 do
4:     $Pair_{cand} \leftarrow P_{cartesian}[i]$
5:     $Sim_{USE} \leftarrow cosine\_sim(Pair_{cand}.left.USE,$
       $Pair_{cand}.right.USE)$
6:     if $Sim_{USE} \geq \tau_{sim}$ then
7:         $P_{remaining} \leftarrow P_{remaining} \cup Pair_{cand}$
8:     end if
9: end for
10: return $P_{remaining}$
```

Algorithm 5 describes our USECluster variant of semantic blocking which takes as input the two ontologies representing the schemas Ont_L, Ont_R, and the number of blocking clusters, $blocking_{cluster}$. In line 2, the algorithm uses K-means clustering to cluster the schema elements based on their semantic representation represented in USE embeddings. Lines 3 to 7 enumerate the clusters. The set of cross-ontology schema element pairs within each cluster ($Pairs_i$) is computed in line 5 and is added to the set of post-blocking pairs, $P_{remaining}$ in line 6. Finally, the set of post-blocking pairs, $P_{remaining}$, is returned in line 7.

Algorithm 5 USECluster(Ont_L, Ont_R, $blocking_{cluster}$)

```
1: $P_{remaining} = \{\}$, $Ont \leftarrow Ont_L \cup Ont_R$
2: $Clusters \leftarrow K$-means($Ont$, $blocking_{clust}$, $USE$)
3: for $i$: 0 to $|Clusters|$-1 do
4:     $cluster \leftarrow Clusters[i]$
5:     $Pairs_i \leftarrow cluster.Nodes_{Ont_L} \times cluster.Nodes_{Ont_R}$
6:     $P_{remaining} \leftarrow P_{remaining} \cup Pairs_i$
7: end for
8: return $P_{remaining}$
```

4.4 Putting it all together

Algorithm 6 provides a description of ALFA's overall functionality and evaluation in terms of the ontology-aware sample selection, label propagation and semantic blocking techniques to efficiently utilize the human-labeled samples to train an effective GNN-based schema alignment model. Lines 2 and 3 show how test samples and unlabeled (remaining) samples are created from the post-blocking pairs. Lines 1, 7 and 8 show the invocations to our semantic blocking, sample selection and label propagation, respectively. The active learning framework is evaluated w.r.t. progressive F1-scores where the entire set of post-blocking pairs is treated as a test set. This gets a progressive quality measure for the model learned incrementally in each active learning iteration and also returns the number of labels required before the classifier reaches its convergent F1-score.

Algorithm 6 Active Learning for Ontology Mapping (ALFA)

```
Require: seed pairs ($P_{seed}$), #pairs to label in each iteration
    ($batchSize$), left and right ontologies ($< Ont_L, Ont_R >$),
    #clusters for sample selection ($n_{cluster}$), #clusters for blocking
    ($blocking_{cluster}$), label propagation mode ($labelPropMode$), label-
    ing budget ($budget$)
1: $P_{pBlock} \leftarrow$ semanticBlocking($Ont_L$, $Ont_R$, $blocking_{cluster}$)
2: $P_{test} \leftarrow P_{pBlock}$
3: $P_{remaining} \leftarrow P_{postBlock}$ - $P_{seed}$
4: $P_{train} \leftarrow P_{seed}$, $iter \leftarrow 1$
5: while $|P_{train}| \leq budget \wedge |P_{remaining}| > 0$ do
6:     $model \leftarrow$ train($P_{train}$)
7:     $Clusters$, $P_{sel} \leftarrow$ ontoAwareSelection($Ont_L$, $Ont_R$,
       $P_{remaining}$, $batchSize$, $model$, $n_{cluster}$)
8:     $P_{prop} \leftarrow$ ontoAwareLabelPropagation($P_{remaining}$, $P_{sel}$, $Clusters$,
       $labelPropMode$, $iter$)
9:     $P_{sel} \leftarrow P_{sel} \cup P_{prop}$
10:    $P_{remaining} \leftarrow P_{remaining} \setminus P_{sel}$
11:    $P_{train} \leftarrow P_{train} \cup P_{sel}$
12:    $F_{score} \leftarrow$ evaluate($model$, $P_{test}$), $iter$++
13: end while
14: return
```

4.5 Computational complexity of ALFA

We discuss the computational complexity of each component in ALFA below.

Ontology-aware sample selection The time complexity of K-means clustering in each AL iteration is $O(I \cdot n_{cluster} \cdot |P_{remaining}| \cdot d)$, where I is the number of K-means iterations until the convergence of clustering (300 iterations by default in scikit), $n_{cluster}$ is the number of clusters (20 by default in ALFA), $|P_{remaining}|$ is the number of remaining pairs and d is the dimensionality of the RGCN model-generated embeddings (64 by default in ALFA) in each AL iteration. The time complexity of computing the label disagreement and the selection of top-k *ambiguous* pairs using a max-heap and a priority queue is $O(|P_{remaining}| + k \cdot log(k))$. Thus, the time complexity of ontology-aware sample selection in ALFA is $O(I \cdot n_{cluster} \cdot |P_{remaining}| \cdot d + k \cdot log(k))$.

The complexity of sample selection in ALFA is primarily influenced by the number of post-blocking pairs that implies that sample selection takes longer time in the initial AL iterations compared to later iterations. Similarly, the number of clusters and the dimensionality of the embeddings need to be modest for the fast convergence of clustering, and lead to low sample selection latencies and user wait times during AL.

Ontology-aware label propagation If $batchSize$ is the size of an AL batch and $|cluster_{largest}|$ is the size of the largest K-means cluster, the time complexity of the selection of the candidate pairs to which the oracle-assigned labels can potentially be propagated is $O(batchSize \cdot |cluster_{largest}|^2)$. The time complexity of unrestricted mode is $O(batchSize \cdot |cluster_{largest}|^2)$ and conservative mode is $O(batchSize \cdot (|cluster_{largest}|^2 + k \cdot log(k)))$, where k is the top-k elements per $Pair_{ref}$ to which the label is propagated. Lastly, the time complexity of the adaptive mode is $O(batchSize \cdot (|cluster_{largest}|^2 + iter \cdot log(iter)))$, where $iter$ is the numerical value of the AL iteration that is used as the dynamically changing value of k in the adaptive mode.

An ontology-agnostic algorithm would need to search through the entire set of post-blocking pairs to identify candidate unlabeled pairs for label propagation. Our ontology-aware propagation has a lower time complexity than an exhaustive search because its search space is confined to the candidate set of semantically similar unlabeled pairs across the ontology clusters that the concepts in a labeled pair belong to. This is captured by the term $|cluster_{largest}|^2$ that estimates the search space to be, at most, the size of the Cartesian Product of concepts within the largest ontology cluster.

Semantic blocking Among the two blocking variants of ALFA discussed in Sect. 4.3, the complexity of USESim is proportional to the size of the Cartesian product of the number of pairs across ontologies which can be written as $O(|Ont_L| \cdot |Ont_R|)$. If $blocking_{cluster}$ is the number of blocking clusters, the time complexity of USESim is $O(I \cdot blocking_{cluster} \cdot (|Ont_L| + |Ont_R|) \cdot d + \sum_{i=1}^{blocking_{cluster}} |cluster_i|^2))$. Here, d is the dimensionality of the USE embeddings (512 by default in ALFA) which we feed as input.

While USESim is an exhaustive blocking variant that enumerates the entire set of pairs in the Cartesian Product of the ontology concepts, our USECluster variant is not exhaustive and enumerates pairs only within the K-means clusters but not across clusters. Hence, the complexity of USECluster is quadratic in the sizes of the clusters, but not in the sizes of the ontologies.

5 Experimental evaluation

In this section, we evaluate the performance of ALFA with the goal of answering the following questions.

- How effective is our proposed ontology-aware sample selection technique in ALFA against other state-of-the-art AL sample selection techniques, in terms of reduction in the number of labeled samples required to achieve a target model quality (F1-score) and sample selection latency?
- How do the three modes of ontology-aware label propagation influence the trade-off between the reduction in human labeling cost and model quality?
- How does semantic blocking influence label skew (class imbalance) and model quality with varying degrees of blocking, compared to an existing representative blocking technique?

In addition, we also study the effect of varying the number of ontology clusters during sample selection on model quality and conclude the experiments with an end-to-end evaluation of ALFA.

5.1 Experimental setup

5.1.1 Datasets

We use four real-world datasets listed in Table 1. CMT-CONF [44] and HUMAN-MOUSE [45] are publicly available datasets of ontologies from the publication and anatomy domains respectively. BANK-KAFS is a proprietary dataset from the finance (banking) domain. FMA-NCI is a dataset from the large BioMed track of the ontology alignment evaluation initiative (OAEI) [46]. In addition to the size of the actual matching pairs ($|Pairs_{Matches}|$) and the total number of schema element pairs in the Cartesian Product ($|Pairs_{Total}|$), we present the label skew as the ratio between them to establish the challenge involved in matching the dataset ontologies. The ground truth (actual matching pairs) for all the datasets is used to simulate human labeling. Note that the ground truth was curated by Subject Matter Experts (SMEs) for BANK-KAFS. While we evaluate ALFA extensively on the first three datasets, i.e., CMT-CONF, HUMAN-MOUSE and BANK-KAFS, we use the FMA-NCI

Table 1 Dataset details

Dataset	$\|Nodes_{Left}\|$	$\|Nodes_{Right}\|$	$\|Pairs_{Matches}\|$	$\|Pairs_{Total}\|$	$Skew_{Label}$
CMT-CONF	39	77	15	3003	1:200
HUMAN-MOUSE	3298	2737	1516	9 Million	1:5937
BANK-KAFS	2148	7170	394	15.4 Million	1:39,086
FMA-NCI	3720	6488	2686	24.1 Million	1:8986

dataset exclusively for an end-to-end system evaluation of ALFA against BERTMap [26, 27], which employs pre-trained and fine-tuned BERT models [3] to align large biomedical ontologies.

5.1.2 Evaluation metrics

We evaluate ALFA using the following metrics.

Progressive F1-score We evaluate the effect of our proposed AL techniques on model performance in terms of progressive F1-score [19, 22, 39, 68], a popular metric used by the AL community. Progressive F1-score is defined as the evaluation F1-score obtained by using the entire set of candidate pairs available for sample selection as the test set [39]. This is used to evaluate sample selection strategies in the AL and crowdsourcing literature as they progressively query the oracle for a sample of the available candidate pairs to be added to the training set in each labeling iteration. Therefore, progressive F1 is plotted as a function of the number of labels acquired from the human or the AL iterations.

Sample selection latency We use sample selection latency to measure the time taken by our sample selection algorithm to select samples for human labeling.

Label skew We measure the performance of our semantic blocking technique in terms of its effect on label skew (class imbalance) on the training set as % of positive labels out of the total set of labels.

Convergent progressive F1 This is the progressive F1-score that can be achieved by a model during AL upon the exhaustion of all unlabeled pairs [39]. It is possible that the convergent progressive F1 is in practice achieved by the model sooner than AL termination.

5.1.3 Baselines

We compare ALFA against several different baselines for each of our proposed AL techniques.

Sample selection baselines We compare the performance of our ontology-aware sample selection technique against several baselines including entropy-based selection [43, 60], Query by Committee (QBC) [40] and an importance weighted sampling method called OASIS [38] from the generic AL literature. From the link prediction literature [6], we include degree-based and centrality-based selection with and without stratification. The implementation details of the baselines are in Appendix A.1. We also include a random sample selection baseline which randomly selects samples for labeling from the available set of candidate pairs.

Label propagation We compare the effect of the different modes (unrestricted, conservative and adaptive) of our proposed ontology-aware label propagation against two baselines that use pre-trained language models, USE [10] and BERT [17]. While BERT-based propagation is borrowed from BERTMap, USE-based propagation performs an exhaustive search over the entire search space of post-blocking pairs to identify pairs to which labels can be propagated. USE-based label propagation propagates the label of a matching or non-matching reference pair ($Pair_{ref}$) to unlabeled pairs with similar cosine similarities between the USE embeddings of their constituent schema elements. BERTMap uses pre-trained and fine-tuned BERT models in unsupervised and semi-supervised modes respectively for ontology alignment [27]. BERTMap's propagation hierarchically propagates a label from a concept pair to its parent and child pairs in the ontology graph thus leveraging the structural information besides the language model. We also compare ALFA against the vanilla baseline of sample selection with no label propagation.

Semantic blocking We also compare our semantic blocking variants (USESim and USECluster) against three baselines, (1) Jaccard similarity-based blocking that has been extensively used for entity matching [39, 40, 69, 70], (2) BERTMap's candidate selection heuristic that builds a word-level inverted index using BERT's WordPiece tokenizer [27] and reduces the search space to a shortlisted set of concept pairs with overlapping sub-words, and (3) exact nearest neighbor search based on L2-Flat index implemented in the FAISS [31, 49] library. We first train a flat index on all concepts from one of the ontologies which can be referred to as the *base* ontology, i.e., CONF from CMT-CONF, MOUSE from HUMAN-MOUSE and KAFS from BANK-KAFS. We then probe the *base* ontology to find the top-*k* closest *base* concepts for each concept from the *probe* ontology (CMT, HUMAN and BANK being the probe ontologies) to find the nearest *(probe, base)* concept pairs in the L2-distance space. In order to arrive at a pre-determined target number of post-blocking pairs, we adjust the value of "*k*" for top-*k* nearest neighbor detection. Although L2-flat index detects exact nearest neighbors, FAISS optimizes the exhaustive search using C++ vector operations, thus avoiding the need to parallelize the search.

End-to-end evaluation We evaluate our end-to-end implementation of ALFA against BERTMap as it was shown to outperform rule-based ontology alignment baselines such as AML [18, 71] and LogMap [13, 30] which require a significant amount of manual assistance. Although BERTMap does not use active learning, it has a similar three-part architecture as ALFA comprising a blocker, matcher and label propagator and can function in both unsupervised and semi-supervised modes without and with BERT fine-tuning. In our comparison of ALFA against both the variants, we allow BERTMap to utilize GPUs (ALFA does not use GPUs while training) and also access batches of matching and non-matching pairs to fine-tune the BERT model in each AL iteration for a fair comparison.

5.1.4 Configurations and settings

We conducted our experiments on a machine with 2.3 GHz 8-Core Intel Core i9 processor, 64GB RAM, an AMD Radeon Pro 5500 M 8 GB GPU and an Intel UHD Graphics 630 (GT2) GPU with 24 execution units running MacOS. We implemented ALFA using Python 3.9.5. We used PyTorch 1.8.1 as the deep learning platform for the implementation of a GNN-based schema alignment [25]. We used scikit-learn 0.24.2 for implementing K-means clustering. Other generic parameter settings for ALFA are described below.

Model parameters We use the RGCN model from Sect. 3.3 with a learning rate of 0.001, 100 epochs, MLP of one hidden layer, ReLU activation taking 512-dimensional USE embeddings as input to output 64-dimensional embeddings and a classification label.

Seed label set The seed label set is the initial set of labeled pairs used to train the model, which is typically 0.1–0.3% of the entire unlabeled set [39]. The actual sizes of the seed label sets are between 5 and 40 across all our datasets.

Batch size In all our experiments, the #pairs selected in each AL batch is 1.27% of the entire unlabeled set. This parameter value was arrived at empirically to control the number of AL iterations (80) and thereby keep the overall runtime to less than 1 h upon larger datasets like BANK-KAFS. In actual practice, the batch size would be dependent on the number of samples a human would prefer to provide labels for, in each iteration.

Termination criterion AL iterations could be terminated either when the labeling budget is exhausted or the desired model quality is achieved. In the current implementation, we terminate AL after consuming all the unlabeled data. The human-in-the-loop for labeling is simulated via the available ground truth of matches between the two schemas.

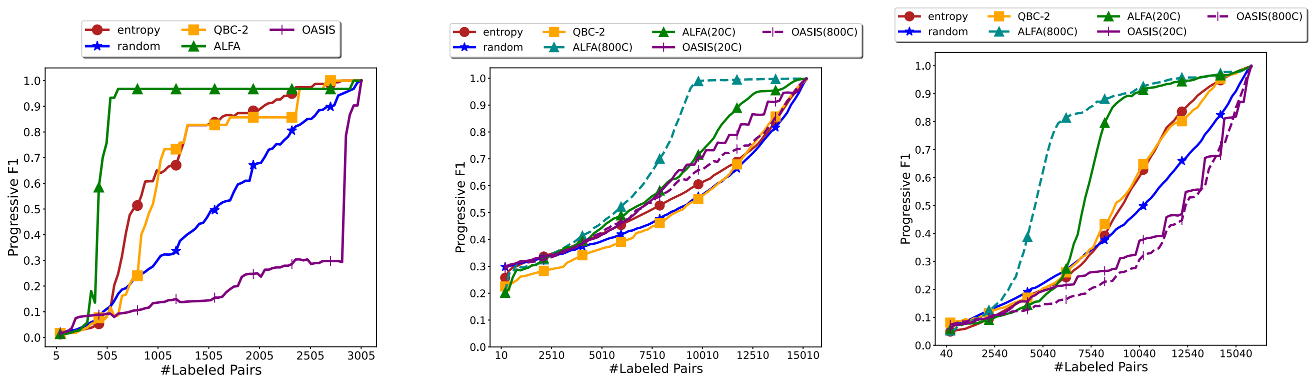
5.2 Evaluation of ontology-aware sample selection

Figures 6 and 7 show that ontology-aware sample selection outperforms generic and link prediction AL baselines w.r.t. prediction quality (progressive F1). We use the entire Cartesian Product of 3003 concept pairs for CMT-CONF, while we reduce the 9 M and 15 M concept pairs from HUMAN-MOUSE and BANK-KAFS respectively to ∼15K candidate pairs by employing negative sampling from a recent work [53] in the ratio 1:9 and 1:39. This samples hard-to-classify negative pairs from the Cartesian Product. We use 20 as the default number of clusters in ALFA(20C), while also including results for 800 clusters on the larger datasets (ALFA(800C)). We use OASIS(20C) and OASIS(800C) as the stratified importance sampling baselines with 20 and 800 strata and QBC-2 with 2 classifiers in the committee. Larger committees such as QBC-10 incur prohibitively long latencies without a significant boost in F1-scores.

Overall, the number of labels required by the best-performing variants of ALFA to achieve a 90% progressive F1-score is 18% (CMT-CONF), 48% (BANK-KAFS) and 73% (HUMAN-MOUSE) of the size of the corresponding set of unlabeled pairs. Compared to ALFA, the next best performing baselines are QBC-2 with a committee of 2 learners and entropy which require 64% of the unlabeled pairs for both CMT-CONF & BANK-KAFS and 92% for HUMAN-MOUSE. Although OASIS is comparable to QBC-2 and entropy on HUMAN-MOUSE, it performs worse than random sampling on CMT-CONF and BANK-KAFS because, OASIS draws samples at random from the most importantly weighted stratum that explains its fluctuating behavior.

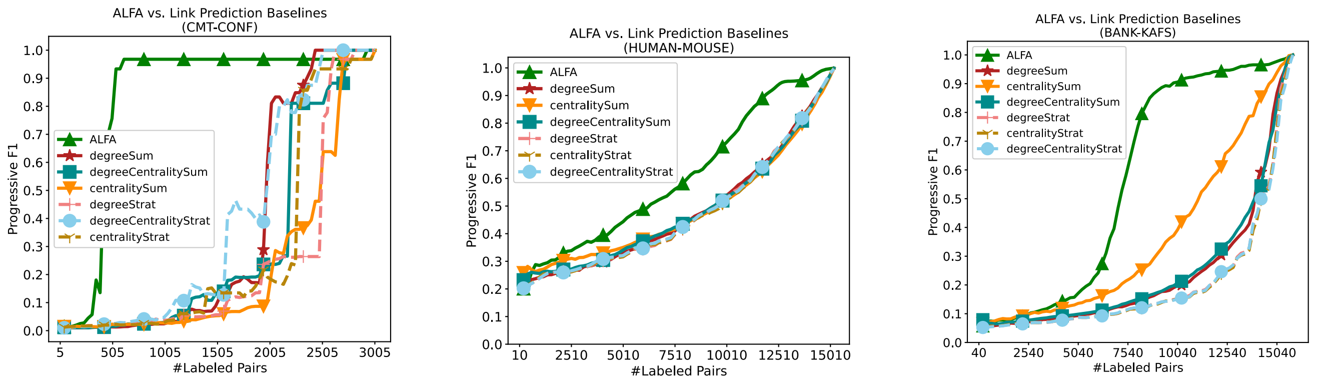
Figure 8 shows the performance of our ontology-aware sample selection technique against the generic AL baselines in terms of sample selection latency. As can be seen, our proposed technique incurs reasonably low latency that is comparable to OASIS and the entropy-based selection baseline. QBC with 2 classifiers (QBC-2) is the most expensive because of training a committee of classifiers in each AL iteration. As expected, random sample selection is the fastest w.r.t. latency but it also yields the lowest F1 scores (Fig. 6). ALFA(800C) using 800 clusters performs better than ALFA(20C) using 20 clusters in terms of F1-scores (Fig. 6) but it also incurs longer sample selection latencies.

Figure 9 shows the latency comparisons with the graph-aware link prediction baselines which also use 20 clusters. We notice that on the CMT-CONF dataset (Fig. 9a), ALFA incurs more latency than the link prediction baselines. However, on the larger datasets such as HUMAN-MOUSE and BANK-KAFS (shown in Fig. 9b, c), the latency of the stratified variants of the link prediction baselines are typically up to 10× higher than their non-stratified counterparts, and our proposed ontology-aware selection in ALFA is 10×-17× faster than the stratified graph-aware link prediction



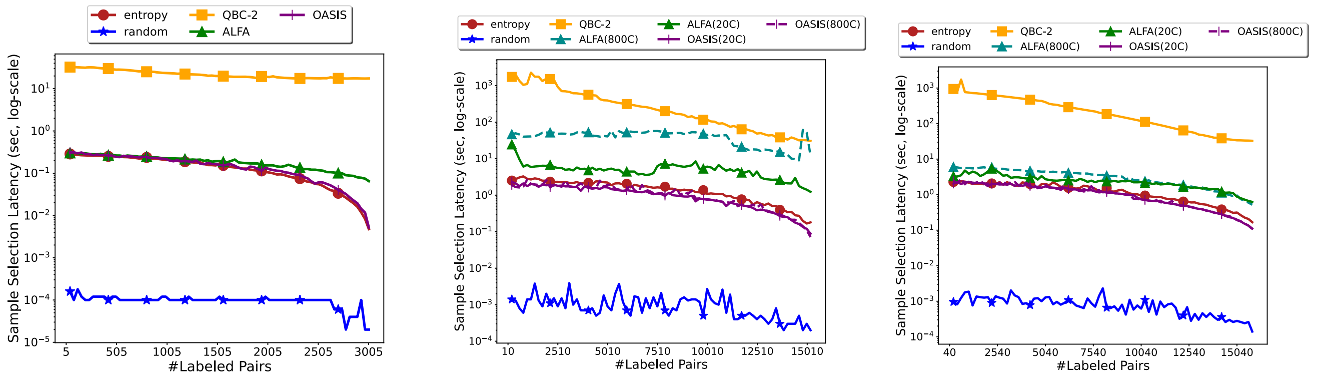
(a) CMT-CONF (b) HUMAN-MOUSE (c) BANK-KAFS

Fig. 6 Ontology-aware sample selection in ALFA versus generic AL baselines w.r.t. progressive F1



(a) CMT-CONF (b) HUMAN-MOUSE (c) BANK-KAFS

Fig. 7 Ontology-aware sample selection in ALFA versus link prediction baselines w.r.t. progressive F1



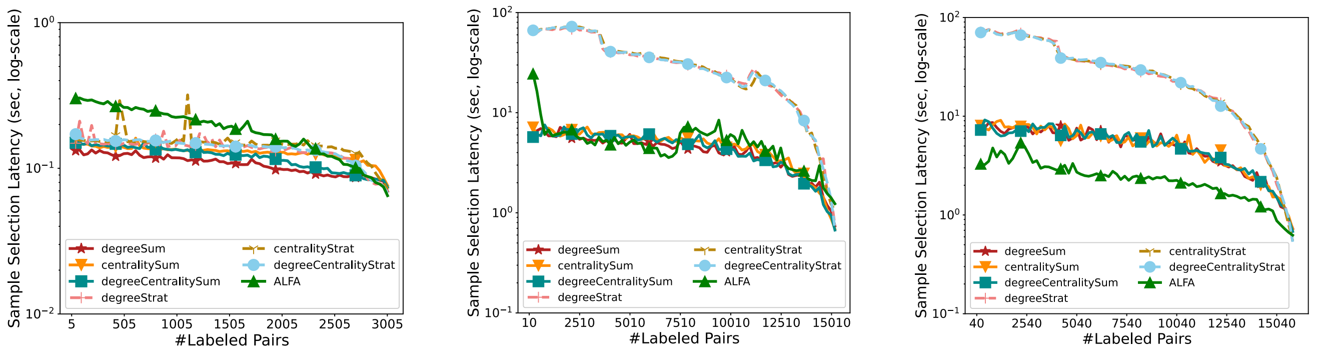
(a) CMT-CONF (b) HUMAN-MOUSE (c) BANK-KAFS

Fig. 8 Ontology-aware sample selection in ALFA versus generic AL baselines w.r.t. latency

baselines. This is because stratified link prediction baselines detect pairs having top-k degree or centrality sum within each cluster which can take longer than detecting the top-k pairs globally using their non-stratified counterparts. However, on smaller datasets like CMT-CONF, the individual clusters are smaller, which reduces the latency difference between the stratified and non-stratified variants while also making them faster than ALFA.

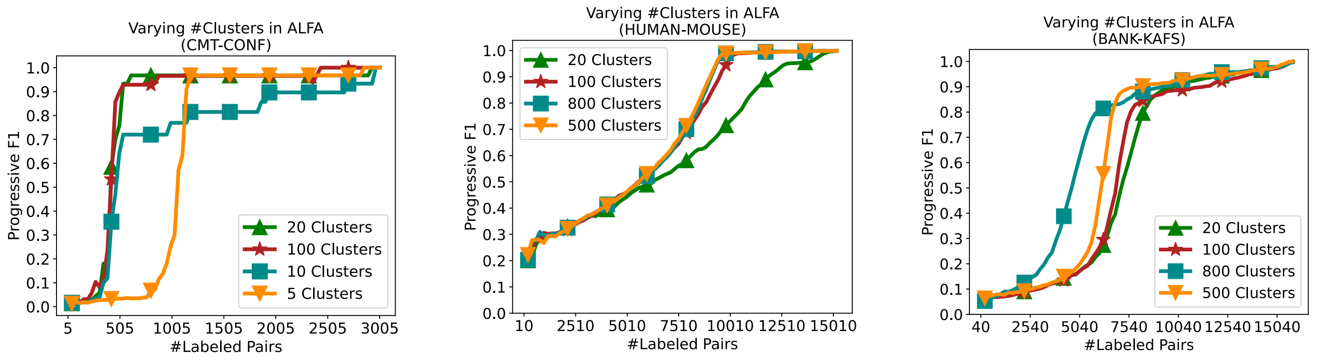
Varying the number of clusters

In Fig. 10, we vary the number of clusters from (5 to 100) for CMT-CONF and 20 to 800 for HUMAN-MOUSE and BANK-KAFS keeping the sizes of the unlabeled pairs in these datasets in perspective. We notice that while fewer than



(a) CMT-CONF (b) HUMAN-MOUSE (c) BANK-KAFS

Fig. 9 Ontology-aware sample selection in ALFA versus link prediction baselines w.r.t. latency



(a) CMT-CONF (b) HUMAN-MOUSE (c) BANK-KAFS

Fig. 10 Varying the number of ontology clusters during ontology-aware sample selection in ALFA

20 clusters lead to a slower convergence to the best possible F1-score, increasing the number of clusters beyond 20 does not bring any significant benefit in convergence speed on the CMT-CONF dataset. On similar lines, 500 clusters for the HUMAN-MOUSE dataset and 800 clusters for the BANK-KAFS dataset are enough to quickly converge to the best possible F1-scores. This indicates the need to use a larger number of clusters as schema size increases, and a requirement to empirically determine the value of #clusters beyond which there is no substantial gain in model quality (F1). We empirically showed the influence of #clusters on the sample selection module evaluated in isolation in Fig. 10. In Sect. 5.5, we will discuss how the number of clusters influences sample selection in the end-to-end implementation of ALFA. In order to automatically determine the value of #clusters beyond which there is no substantial gain in model quality, we implement the ELBOW method [61] using *Silhouette-coefficient* [56].

5.3 Evaluation of ontology-aware label propagation

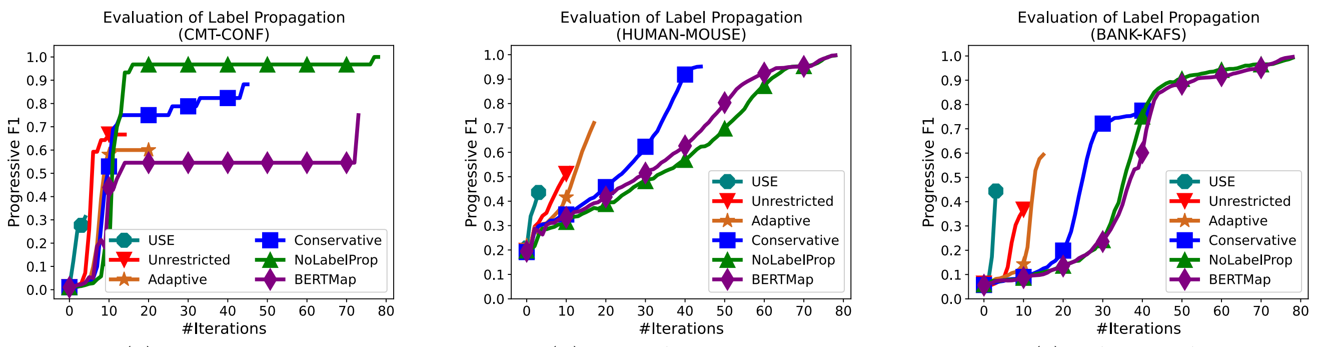
We evaluate our ontology-based label propagation technique in terms of its influence on the trade-off between the reduction in human labeling cost and model quality. We do so by two sets of experiments. Figure 11 shows the first set of experiments capturing the variation of the progressive F1-score with respect to the number of AL iterations. The three modes of label propagation in ALFA are compared against the label propagation baselines discussed in Sect. 5.1.3 when each of them is plugged into ALFA. Among the compared methods, USE-based propagation baseline aggressively exhausts all the unlabeled data in fewest AL iterations and achieves the lowest progressive F1-score. On the other extreme, BERTMap propagates labels conservatively only to the parent and child pairs and achieves the closest progressive F1-score to the baseline that uses no label propagation. The three modes of label propagation in ALFA are flanked by these two extremes while balancing the trade-off between the two dimensions that determine optimality i.e., convergent F1 score and the labeling cost reduction.

Among the three label propagation modes of ALFA, we can notice that unrestricted label propagation exhausts the labels quickly and achieves lower convergent F1-score than the conservative mode. While this pattern is consistent on all the datasets, the distinction between unrestricted and adaptive modes is not pronounced in Fig. 11a. This is because CMT-CONF is a smaller ontology with fewer candidates which qualify for label propagation. BERTMap's hierarchi-

Table 2 Breakdown of label propagation quality on pairs with inferred labels

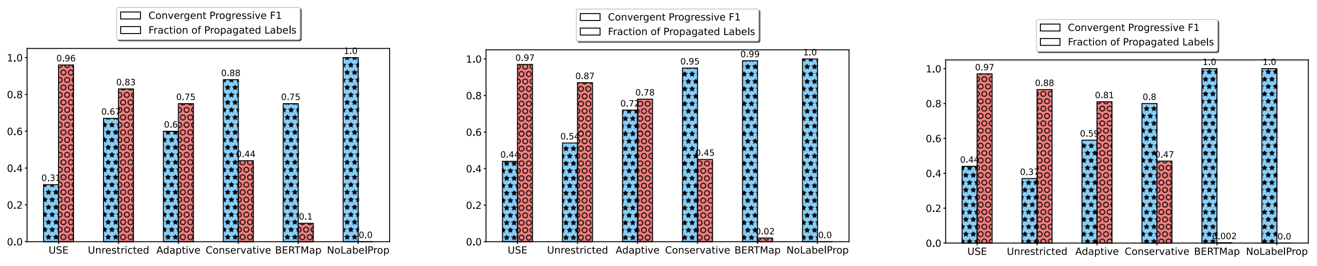
Dataset	Label propagation	#Inferred labels	Inferred precision	Inferred recall	Inferred F1-score	Overall F1-score	Average latency (s)
CMT-CONF	USE	**2,868**	0.13	**1**	0.23	**0.31**	**3.63**
	Unrestricted	2,501	**0**	**0**	**0**	0.67	0.19
	Adaptive	2,263	0.14	0.14	0.14	0.6	0.25
	Conservative	1,318	**1**	**1**	**1**	**0.88**	0.37
	BERTMap	**298**	0.11	0.4	0.17	0.75	**0.009**
HUMAN-MOUSE	USE	**14,479**	0.98	0.85	0.91	**0.44**	**174.78**
	Unrestricted	13,135	**0.71**	**0.74**	**0.73**	0.54	3.88
	Adaptive	11,893	0.75	0.81	0.78	0.72	2.43
	Conservative	6,892	0.81	0.88	0.84	0.95	8.8
	BERTMap	**267**	**0.99**	**0.99**	**0.99**	**0.99**	**0.04**
BANK-KAFS	USE	**15,267**	0.88	0.78	0.83	0.44	**254.87**
	Unrestricted	13,855	**0.16**	**0.02**	**0.04**	**0.37**	1.85
	Adaptive	12,756	0.22	0.22	0.22	0.59	2.79
	Conservative	7,424	0.19	0.21	0.2	0.8	4.82
	BERTMap	**25**	**1**	**1**	**1**	**1**	**0.04**
	NoLabelProp	**0**	N/A	N/A	N/A	**1**	N/A

The color green indicates best scores and color red indicates poor scores



(a) CMT-CONF (b) HUMAN-MOUSE (c) BANK-KAFS

Fig. 11 Evaluation of various degrees of label propagation in ALFA on convergent progressive F1-score



(a) CMT-CONF (b) HUMAN-MOUSE (c) BANK-KAFS

Fig. 12 Trade-off between convergent progressive F1-score and fraction of propagated labels in ALFA

cal propagation performs poorly on CMT-CONF because of incorrectly propagating labels to dissimilar parent and child pairs. BERTMap's convergent F1-score is higher on HUMAN-MOUSE and BANK-KAFS but the extent of label propagation is very limited.

Figure 12 contains the second set of experimental results where we show the trade-off between the reduction in the human cost of labeling and the convergent progressive F1-score achieved by all the approaches for label propagation across three datasets. For each label propagation technique, we show the fraction of propagated labels, i.e., the percentage of the number of unlabeled pairs obtaining labels from label propagation instead of a human oracle. This fraction of concept pairs indicates the savings obtained by not consulting the human oracle for their labels.

Figure 12 shows that USE-based propagation and BERTMap yield the best performance only on one of the dimensions either with the highest convergent F1 or the highest amount of label propagation while performing the worst on the other dimension. ALFA, on the other hand, is found to perform well on both quality (convergent F1-score) and savings in labeling cost (fraction of propagated labels), and it gives the flexibility to choose one of unrestricted, conservative or adaptive modes depending on the end user preference.

The unrestricted mode of label propagation provides a significant reduction in the cost of human labeling ($\sim$80% across all the datasets) while achieving a relatively lower F1-score than the other two modes in ALFA. On the other hand, the conservative mode of label propagation achieves a low amount of reduction in the cost of human labeling ($\sim$45% across all datasets) while achieving high F1-scores. We found that the fraction of correctly propagated labels across both matching and non-matching pairs is $\sim$97% for all the three modes of label propagation in ALFA averaged across all the datasets. However, the fraction of matching labels correctly propagated is $\sim$25% for unrestricted, $\sim$36% for conservative and $\sim$39% for adaptive modes across all datasets. This decline in label propagation accuracy for matching pairs is due to label skew.

Table 2 shows the breakdown of the label propagation quality exclusively based on inferred labels. In terms of the time complexity, BERTMap has the least propagation complexity of $O(1)$ because it takes constant time to propagate a label from a concept to its immediate parents or children in the ontology. This is also reflected in the average inference latency shown for each active learning iteration. Note that the batch size of samples in each AL iteration over which label propagation is done is the same for all the compared approaches. BERTMap achieves the best inferred F1-score on HUMAN-MOUSE and BANK-KAFS datasets but its hierarchical propagation is conservative w.r.t. the number of pairs to which labels are propagated. Hence it saves the least on labeling budget. USE-based propagation, on the other hand, has high inferred F1-scores on HUMAN-MOUSE and BANK-KAFS despite propagating labels to several pairs. The reason is that USE-based propagation performs an exhaustive search over all unlabeled pairs to select candidates to which labels can be propagated. The time complexity of USE-based exhaustive search is $O(batchSize.|Ont_L|.|Ont_R|)$ without blocking, and $O(batchSize.|P_{postBlocking}|)$ when applied on post-blocking pairs. USE incurs the longest inference latency on all the datasets.

In contrast to the costly brute-force search in USE-based propagation for inferred pairs, ALFA's label propagation complexity is $O(batchSize.|cluster_{largest}|^2)$ which is significantly lower because its search space is limited to the candidate pairs across the clusters to which the concepts of a labeled pair belong. Hence, the conservative mode of ALFA is up to 50× faster than USE on inference latency. The search within a limited space of candidate pairs explains the lower F1-scores for ALFA but also underlines the importance of using ontology-aware mechanisms to optimize the search for inferred pairs. ALFA's conservative mode achieves a competitive inferred F1-score of 84% on HUMAN-MOUSE only next to USE, and a perfect inferred F1-score of 1.0 on CMT-CONF. As per complexity analysis in Sect. 4.5, adaptive mode is supposed to take the longest inference latency among the three label propagation modes of ALFA, but Table 2 shows that conservative mode incurs the longest latency. This is due to the variable cluster sizes that are encountered over several AL iterations by conservative mode, whereas adaptive and unrestricted modes terminate earlier after fewer AL iterations. Although the sizes of sampled unlabeled pairs are roughly the same for both BANK-KAFS and HUMAN-MOUSE, we observed a longer inference latency on HUMAN-MOUSE for ALFA because the cluster sizes are more skewed in HUMAN-MOUSE compared to BANK-KAFS.

Table 2 also shows that the overall F1-score for USE is consistently low although the inferred F1-score is high. This is due to the aggressive propagation in USE which exhausts all the labels in the first few iterations and terminates active learning without the GNN model maturing enough. Training the GNN model with more epochs or layers may help combat label skew and improve training accuracy while training upon a large number of inferred labels in a single active learning iteration. However, it is unsuitable for human-in-the-loop active learning as using a heavyweight model further leads to longer user wait time and training latency. We used the entire CMT-CONF dataset which has a skew of 1:200 and employed negative sampling at 1:9 on HUMAN-MOUSE and 1:39 on BANK-KAFS same as Sect. 5.2.

In principle, we could improve the overall F1-score of the USE-based propagation method by spreading the inferred labels across multiple smaller AL batches, allowing the model to train over several AL iterations. In addition, one could implement approximate binary search variants of label propagation for both USE and ALFA to further improve the search efficiency. The single-threaded average inference latency shown in Table 2 can be reduced by parallelizing propagation using multi-threading for all the approaches. These optimization opportunities in ALFA's label propagation are left for future work.

Overall, we recommend using ALFA's conservative propagation if there are several post-blocking pairs. In case that the post-blocking pairs are few and an exhaustive search is affordable, ALFA can interchangeably use its conservative mode or a conservative implementation of USE-based propagation. On the other hand, if the user wants to trade labeling savings for F1-score, and there is an oracle willing to label several pairs, we recommend using BERTMap's hierarchical propagation.

5.4 Evaluation of semantic blocking

Blocking is a pruning technique that aims to filter the obvious non-matching pairs (i.e., the negative labels) and reduce the label skew. Blocking cannot influence false positives or true positives as every pair that is not pruned away by blocking is not guaranteed to be match. It can only incur false negatives by incorrectly pruning the matching pairs. Therefore, we evaluate our proposed semantic blocking in terms of the trade-off between the reduction in quality of the model due to false negatives resulting from blocking, and the reduction in label skew in terms of improvement in the percentage of positive (matching) labels.

Figure 13 shows this trade-off between the percentage of positive labels and false negatives (FNs) while increasing the percentage of blocking clusters. We also plot recall (as a percentage) in the figure to show that the increase in false negatives symmetrically leads to a drop in recall. For uniformity across datasets, we represent the number of blocking clusters as a percentage of the total number of distinct concept nodes across the two ontologies. For example, if we use 200 clusters upon a pair of ontologies containing a total of 1,000 concepts, we denote this setting as *20% blocking clusters*. The reason for using %blocking clusters is to be scale-invariant across datasets by avoiding absolute values. We can observe that the slope of the percentage of positive labels (or the rate at which label skew decreases) is higher than that of FNs thereby showing that the benefits of our semantic blocking outweigh the penalty that it pays in terms of FNs.

Table 3 compares our proposed semantic blocking variants, USESim and USECluster, against Jaccard similarity based blocking and BERTMap's inverted index-based blocking baselines. We obtained several candidate entries for post-blocking pairs in Table 3 by varying the similarity thresholds for Jaccard similarity-based blocking between

Table 3 ALFA versus Jaccard, BERTMap and FAISS L2-Flat index (exact nearest neighbor) blocking baselines

Dataset	Jaccard Threshold	#Post-blocking Pairs	%False negatives				
			ALFA (USESim)	ALFA (USECluster)	Jaccard (LabelSim)	BERTMap (InvertedIndex)	FAISS (L2Flat)
CMT-CONF	**0**	**3003**	**0**	**0**	0	0	0
	[0.001–0.01]	**134**	**13.3**	**20**	66.7	20	13.3
	0.1	**69**	**13.3**	**33.3**	66.7	35.3	13.3
	0.2	**40**	**20**	**26.7**	66.7	42.8	20
	0.4	**9**	**60**	**66.7**	73.3	N/A	66.7
HUMAN-MOUSE	**0**	**9 M**	**0**	**0**	0	0	0
	[0.001–0.01]	**57 K**	**11.4**	**17.2**	82.1	5.7	8.7
	0.1	**27 K**	**15.4**	**20.1**	82.6	7.5	10.8
	0.2	**1.4 K**	**30.9**	**28.2**	85.8	N/A	25.9
	0.4	**366**	**N/A**	**N/A**	87.1	N/A	76.1
	0.6	**207**	**N/A**	**N/A**	88.5	N/A	86.5
BANK-KAFS	**0**	**15.4 M**	**0**	**0**	0	0	0
	[0.001–0.01]	**0.84 M**	**1.5**	**28.4**	6.3	5.1	2.1
	0.1	**0.76 M**	**1.8**	**28.7**	6.3	5.1	2.4
	0.2	**0.11 M**	**8.2**	**30.5**	12.7	16.6	7.6
	0.4	**12 K**	**23**	**36**	30.8	48	21
	0.6	**1.6 K**	**39.6**	**43.8**	42	98.5	37.8
	0.8	**280**	**59.2**	**60**	60	N/A	60.1

The color green indicates best scores and color red indicates poor scores

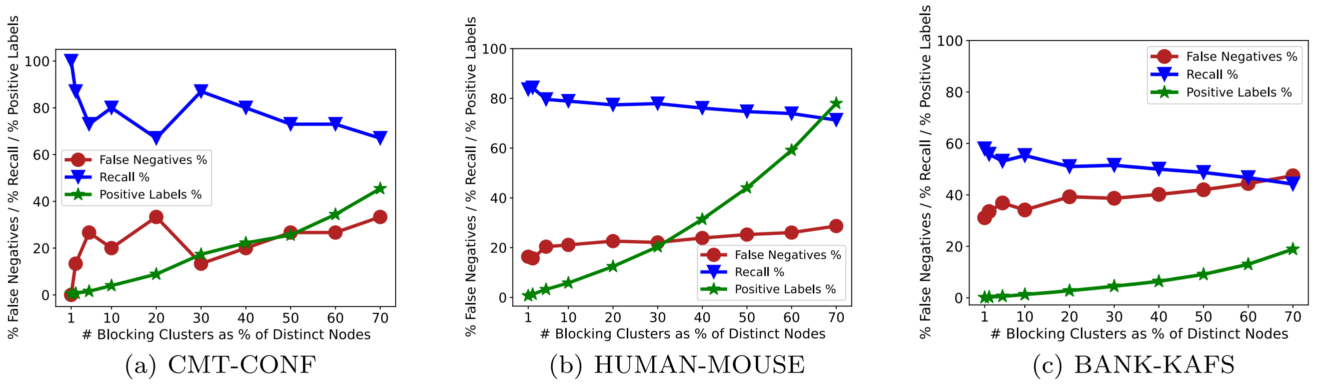


Fig. 13 Trade-off between false negatives (recall) % and positive labels % for various degrees of blocking in ALFA

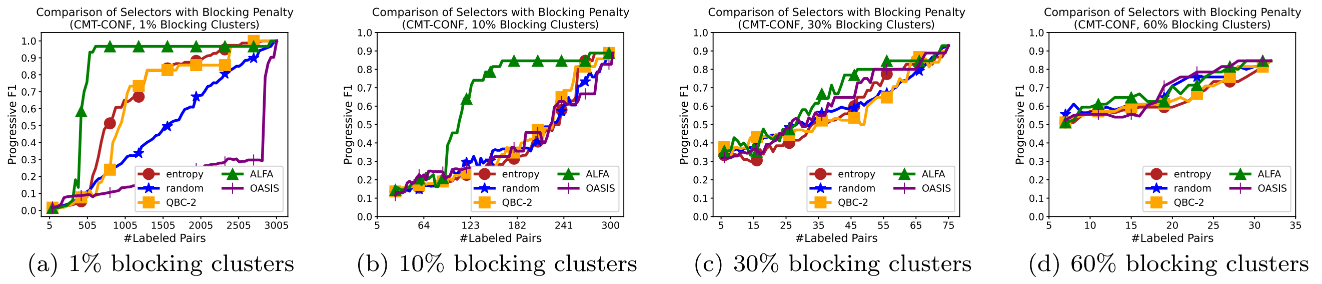


Fig. 14 Evaluation of sample selection at various degrees of blocking in ALFA

0.001 and 0.8. We imposed an additional constraint that the minimum number of post-blocking pairs should have at least one positive sample in them. We thus obtain a minimum of 9 post-blocking pairs for CMT-CONF (at a similarity threshold of 0.4) and ∼200 for HUMAN-MOUSE and BANK-KAFS datasets at maximum thresholds of 0.6 and 0.8, respectively. The parameters of the remaining blocking approaches are adjusted to reach the target number of post-blocking pairs obtained using the Jaccard threshold for each row in the table. N/A entries in the table indicate that no suitable parameter settings are found for the corresponding blocking approaches to reach the pre-specified target pairs when they are typically very few.

Since Jaccard similarity computes the fraction of matching words for a pair of concepts, the correlation between the similarity thresholds used, and the target post-blocking pairs obtained is based on the number of words in the concept names and the total number of concept pairs to be matched in each dataset. Jaccard similarity influences the variation in the post-blocking pairs in a step-wise manner and the distribution of post-blocking pairs across thresholds is skewed. For some intervals such as [0.001–0.01], varying the thresholds would not change the number of post-blocking pairs, and there would be an abrupt reduction in the post-blocking pairs when a slightly higher threshold like 0.1 is used. Due to this step-wise behavior, not all possible fractions (percentages) of the Cartesian Product can serve as target post-blocking pairs. This is because several of those candidates may not be reachable by all the blocking methods. Therefore, we used Jaccard similarity thresholds that can lead to the generation of distinct candidate values of post-blocking pairs in Table 3.

From Table 3, we can observe that the USESim blocking variant of ALFA outperforms the baselines on CMT-CONF and BANK-KAFS datasets with an exception that BERTMap's inverted index-based blocking performs the best on HUMAN-MOUSE dataset. This is because the concept names in HUMAN-MOUSE generally contain several sub-words which were captured effectively by the inverted index. FAISS's L2-flat index performs an exhaustive search for the nearest neighbors similarly to USESim and hence incurs comparable %False Negatives as USESim on all the datasets. Although we do not evaluate other approximate nearest neighbor search methods such as IVF, HNSW from the FAISS library, we refer the reader to Papadakis et al. [48] for an extensive study on how nearest neighbor approaches can be tuned for blocking purposes.

Jaccard-based blocking baseline performed poorly in most cases including the HUMAN-MOUSE dataset. This explains that a naive implementation of word similarity using Jaccard metric cannot capture the word importance as effectively as BERTMap does through its inverse document frequency metric or an advanced language model such as USE. USECluster serves as an approximation to the USESim variant of ALFA which is also reflected in the percentage of false negatives on the CMT-CONF and HUMAN-MOUSE datasets. On the BANK-KAFS dataset, when the target number of post-blocking pairs is high, USECluster uses fewer clusters which do not effectively capture the proximity among the pairs. USECluster is more effective when several clusters are used to obtain fewer target post-blocking pairs. Among all the blocking approaches, Jaccard similarity-based blocking and USESim blocking approaches are parallelized using 48 threads as they enumerate the entire Cartesian Product and they are still ∼15× slower than USECluster, BERTMap and the L2 flat index [49] in FAISS on an average across all the datasets. Although FAISS performs an exhaustive search, it is faster than USESim because of the way the vector operations are optimized in the C++ implementation of the FAISS library. Note that our USESim and USECluster are natively implemented in Python. We use USECluster as the default blocking mechanism in ALFA which can be interchangeably used with FAISS.

Finally, Fig. 14 shows the effect of blocking on model quality. We plot the variation of the progressive F1-score for different sample selection techniques upon the CMT-CONF dataset for different degrees of blocking. We use CMT-CONF dataset for this experiment as it has only ∼3K pairs in total, and it is possible to train a competent, lightweight (i.e., with limited number of layers and few training epochs) classifier with the default AL batch sizes in a few AL iterations even under the presence of label skew. On the remaining datasets, at low blocking degrees, the label skew coupled with the size of the dataset prevents our GNN model from successfully converging to a low training loss.

The results as expected, show that the higher the degree of blocking, the poorer is the model performance. ALFA does better than all the baselines in terms of model quality (Progressive F1-score) for blocking up to 30% which can be attributed to the fewer false negatives incurred by semantic blocking as compared to the other techniques. Blocking beyond 30% blocking clusters (the number of clusters are plotted as a percentage of distinct nodes as explained earlier), leads to poorer performance in terms of model quality with no significant difference among various sample selectors. This can be attributed to the increase in false negatives as we increase the degree of blocking. The reduction in skew is from 1:200 to 1:5 for CMT-CONF i.e., a 40× improvement in label skew is observed without significantly affecting model quality. This factor of improvement in label skew also holds for the remaining datasets.

Our observation in Fig. 14 is consistent with our claim that there are few false negatives at low blocking degrees where *if a model can successfully be trained to completion*, ends up getting a high convergent F1-score. At higher blocking degrees, the convergent F1-score drops for the model regardless of the example selector used, due to a large number of false negatives.

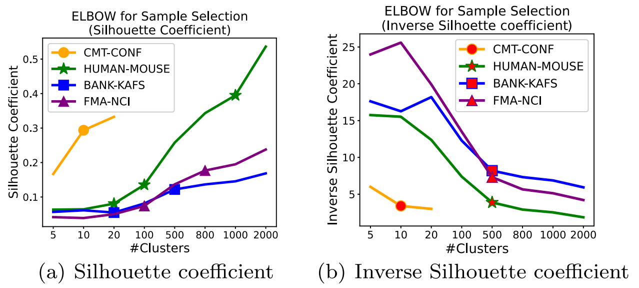


Fig. 15 ELBOW: detect #clusters for sample selection

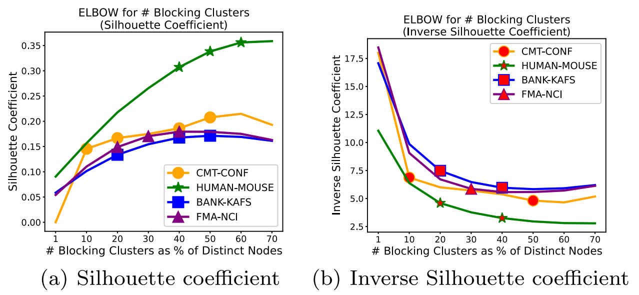


Fig. 16 ELBOW: detect #clusters for blocking

5.5 ALFA: end-to-end system evaluation

Having evaluated our sample selection, label propagation and blocking techniques, we now provide a brief summary on the usability of ALFA in terms of the different parameter settings and their effects on ALFA's performance.

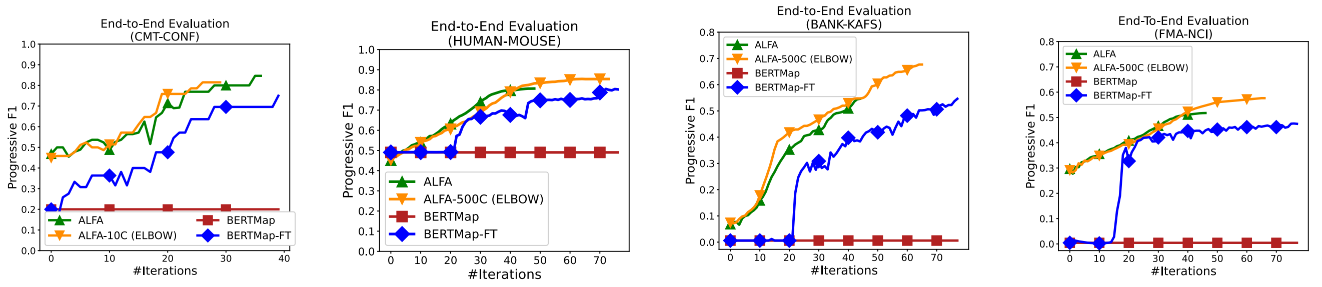
Choosing number of clusters. Ontology clustering is a critical step for both sample selection and label propagation in ALFA. In the real deployment of ALFA, automatically choosing the optimal number of clusters can be done using ELBOW method [61] and *Silhouette-coefficient* [56]. Automatic detection of the optimal number of clusters is crucial for sample selection in ALFA, as the disagreement between the clustering and classification (GNN) models happens in each active learning iteration and determines the quality of samples selected for human labeling.

Silhouette coefficient [59] is defined as $\frac{(b-a)}{max(a,b)}$ where "a" is the mean intra-cluster distance and "b" is the mean inter-cluster distance for each point in a cluster, and the mean silhouette coefficient is computed over the points in all clusters. We used silhouette_score implemented in the *scikit-learn* library [59] with its default parameter settings. Silhouette coefficient lies between -1 and 1 where negative values indicate that the points are incorrectly clustered, 0 indicates overlapping clusters, and 1 indicates the best possible clustering. Intuitively, silhouette coefficient rewards clusters with high inter-cluster distance and low intra-cluster distance. ELBOW curve plots a cluster goodness metric such as silhouette coefficient at various values of #clusters and picks a value resembling an elbow joint in the curve, beyond

Table 4 End-to-end evaluation breakdown of ALFA (default and ELBOW variants) vs. BERTMap

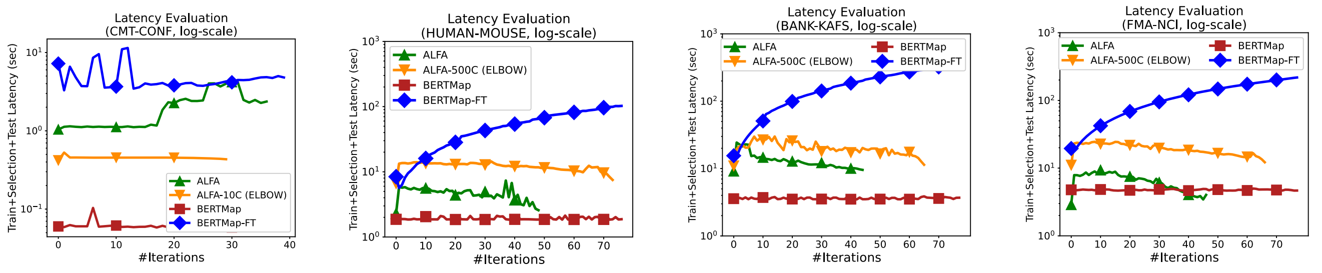
Dataset	#PB Pairs	%False negatives (blocking)		%Inferred labels (label propagation)			Overall convergent F1-score			
		ALFA (ELBOW)	BERTMap	ALFA	ALFA (ELBOW)	BERTMap	ALFA	ALFA (ELBOW)	BERTMap	BERTMap (FT)
D1	46	**26.7**	42.8	**2.3**	**18.6**	4.4	**0.85**	0.82	**0.2**	0.75
D2	3.2K	**23.9**	22.4	**40.4**	8.5	**3.9**	0.81	**0.85**	**0.49**	0.81
D3	6.9K	**39.3**	61.3	**44.6**	19.3	**0.3**	0.55	**0.68**	**0.01**	0.55
D4	6.4K	**53.8**	67.9	**42.8**	16.8	**0.5**	0.52	**0.58**	**0.01**	0.48

D1—CMT-CONF, D2—HUMAN-MOUSE, D3—BANK-KAFS, D4—FMA-NCI, #PB pairs—post-blocking pairs
The color green indicates best scores and color red indicates poor scores



(a) CMT-CONF (b) HUMAN-MOUSE (c) BANK-KAFS (d) FMA-NCI

Fig. 17 End-to-end evaluation of ALFA versus unsupervised and fine-tuned variants of BERTMap (F1-score)



(a) CMT-CONF (b) HUMAN-MOUSE (c) BANK-KAFS (d) FMA-NCI

Fig. 18 End-to-end evaluation of ALFA versus unsupervised and fine-tuned variants of BERTMap (latency)

which the cluster goodness does not improve significantly. The number of clusters at the elbow point in the curve is used as the optimal number of clusters.

In the end-to-end evaluation of ALFA, we ran ELBOW by clustering the post-blocking pairs on their USE embeddings. Besides the datasets we already have, we also chose the FMA-NCI biomedical dataset used by BERTMap (see Table 1 for dataset details) for this evaluation. Figure 15a shows #clusters on X-axis and the silhouette coefficient on the Y-axis. We plotted #clusters at uniform intervals on the X-axis irrespective of their numerical values to show the effect of using fewer clusters clearly in the charts.

From Fig. 15a, we observe that the silhouette score is always above 0 (note that it can range from − 1 to 1) which emphasizes that our clustering is generally good without any overlapping clusters or incorrect clustering. On the flipside, we had multiple candidates for the elbow point while plotting silhouette coefficient and could not find a decisive elbow point in Fig. 15a. We therefore plotted the inverse silhouette coefficient in Fig. 15b which clearly gave us 10 clusters for CMT-CONF and 500 clusters as the elbow points for the larger datasets.

Choosing the mode of label propagation The choice of the mode of label propagation depends on the actual cost of human labeling and the available labeling budget. While unrestricted mode of label propagation can be used for maximum reduction in human labeling, it comes at the cost of lower model quality. The conservative mode allows a fine grained control over the amount of label propagation and should be a suitable choice in most cases based on available budget for human labeling.

Choosing the degree of blocking If the criterion is avoiding false negatives, a conservative blocking setting is preferred. On the other hand, if the end user has limited resources and time to train a model, we need to reduce label skew in a dataset. In such cases, we recommend selecting a high blocking degree by trading false negatives for label skew reduction on larger datasets. As we have mentioned in Sect. 5.4, active learning demands the usage of lightweight models that benefit from a high blocking degree by achieving convergence to low training loss.

Following our discussion in Sect. 5.4, we represent the number of blocking clusters as the percentage of distinct nodes for uniformity across datasets. We have used this notation earlier in Figs. 13 and 14. Figure 16 shows how ELBOW is applied to auto-detect #blocking clusters. Note that this is a different evaluation from the detection of #clusters using ELBOW for sample selection shown in Fig. 15. Here, we show how ELBOW can be applied to the entire set of concept pairs exhaustively enumerated in the Cartesian Product between the ontologies.

In Fig. 16a, we do not observe a decisive ELBOW point using silhouette coefficient. Therefore, we plot the inverse silhouette coefficient in Fig. 16b which shows that elbow points exist at 10% and 50% for CMT-CONF and at 20% and 40% clusters for HUMAN-MOUSE. Using 50% clusters for CMT-CONF and 40% clusters is preferred as they are the closest points to convergence. Another important criterion while choosing an elbow point for blocking is that BERTMap's inverted index-based blocking should be able to generate an equivalent number of post-blocking pairs to ALFA. Although we have elbow points at both 20% and 40% for BANK-KAFS, we choose 20% as the final elbow point because beyond that setting, there is no support for equivalent blocking in BERTMap. In the case of FMA-NCI, we choose 30% as the elbow point.

Comparison with BERTMap We evaluate the end-to-end implementation of ALFA against the unsupervised and fine-tuned (FT) variants of BERTMap. We compare our default implementation of ALFA that uses 20 clusters for sample selection against ALFA (ELBOW) which uses the automatically discovered ELBOW points of 10 clusters for CMT-CONF and 500 clusters for HUMAN-MOUSE, BANK-KAFS and FMA-NCI. We use the conservative mode of label propagation as the default settings of ALFA. We use the default parameter settings from [26] for BERTMap. Note that we fine-tune BERTMap-FT using the accurate oracle-provided labels in each AL iteration instead of relying on weak labeling heuristics such as synonym and antonym detection. Since we re-purpose the inverted index-based negative sampler in BERTMap for blocking, we include the blocking penalty for both BERTMap and ALFA in the convergent F1-scores reported in Table 4.

Corresponding to the ELBOW points detected in Fig. 16, we set equivalent blocking degrees for both ALFA and BERTMap to obtain the same number of post-blocking pairs for all the approaches compared in Table 4. Table 4 shows that blocking in BERTMap incurs a higher percentage of False Negatives and label skew (of 1:4, 1:2, 1:54 and 1:6) than that of ALFA whose skew is 1:3, 1:2, 1:36 and 1:4 respectively on CMT-CONF, HUMAN-MOUSE, BANK-KAFS and FMA-NCI. Figure 17 shows that unsupervised BERTMap using pre-trained BERT model does not improve with more AL iterations unlike BERTMap-FT which is iteratively fine-tuned on labeled pairs.

Although both the default implementation of ALFA and BERTMap-FT have similar convergent F1-scores, ALFA requires half of the AL iterations as BERTMap-FT to convergence because of its effective label propagation (see the percentage of inferred labels in Table 4). Note that the results shown in Fig. 17 and Table 4 are on the same labeling budget for both systems. Despite utilizing GPUs, BERTMap-FT incurs at least 10$\times$ longer training and inference latencies which render it less effective for the AL setting than ALFA (see Fig. 18). This is because pre-trained language models like BERT need rigorous training to be fine-tuned effectively that also causes them to run out of the limited CUDA memory on GPUs. BERTMap uses smaller mini-batches of 50 training examples which overcomes the memory issue but slows down training. If we reduce the training time budget for BERTMap-FT, we cannot fine-tune it and it will perform as poorly as its unsupervised variant. Although ALFA achieves over 80% F1 on CMT-CONF and HUMAN-MOUSE, its lower convergent F1 of 55% and 52% on BANK-KAFS and FMA-NCI are due to 39.3% and 53.8% False Negatives from the high degree of blocking. ALFA's convergent F1-scores prior to applying blocking penalty on these two datasets were 72% and 82% respectively.

Table 4 shows that ALFA (ELBOW) achieves the highest convergent F1-score on all the larger datasets—HUMAN-MOUSE, BANK-KAFS and FMA-NCI. The reason for this is the larger number of clusters (i.e., 500) auto-detected by ALFA (ELBOW) as compared to the default implementation of ALFA that uses 20 clusters. On CMT-CONF, ALFA (ELBOW) uses 10 clusters which leads to a slight drop in F1-score than ALFA. Another interesting observation is that the number of labels propagated is higher for ALFA (ELBOW) than ALFA on CMT-CONF, and is lesser on larger datasets. This observation emphasizes the importance of the number of clusters, and how it symmetrically influences the percentage of inferred labels during label propagation and the overall convergent F1-score. Using fewer clusters leads to higher label propagation and lower convergent F1-score compared to using more clusters.

We can also observe from Fig. 17b–d that ALFA (ELBOW) sustains for more AL iterations than ALFA due to its reduced extent of label propagation. This further establishes the trade-off inherent in label propagation between the savings in labeling by an oracle and the convergent F1-score attained. We can also notice that ALFA (ELBOW) incurs longer overall latency than ALFA in Fig. 18b–d due to the larger number of clusters and longer clustering latency in each active learning iteration. On the other hand, ALFA (ELBOW) uses fewer clusters than ALFA on CMT-CONF which leads to its lower overall latency in Fig. 18a, and a slightly earlier termination in Fig. 17a owing to a higher percentage of inferred labels observed in Table 4 on that dataset.

6 Conclusions

In this work, we propose ALFA, a GNN-based AL framework for semantic schema alignment. ALFA addresses the limitations of existing AL techniques by exploiting the rich semantic information in the underlying schemas captured as ontologies. We present ontology-aware techniques for sample selection, label propagation and semantic blocking. We have provided an extensive evaluation of ALFA over three real-world datasets while comparing its performance with several state-of-the-art AL baselines. We show that exploiting the rich semantics in the underlying schema substantially reduces the cost of human labeling of training data (27–82%) as compared to other AL techniques. Our ontology-aware sample selector in ALFA achieves this cost reduction while maintaining low sample selection times (in the order of a few seconds) and comparable schema matching quality (90% F1-score) to models trained on the entire set of available training data. Our end-to-end system evaluation shows the individual effectiveness of each standalone component in ALFA over a state-of-the-art ontology alignment solution, BERTMap, while also surpassing the latter on the speed of convergence to the best possible F1-score and overall latency.

A Appendix

In the appendix, we provide implementation details for ALFA baselines as well as describe the content made available on our GitHub repository [1] that facilitates the reproducibility of our proposed system.

A.1 Implementation details for the baseline sample selectors

A.1.1 Entropy-based selection

For each unlabeled concept pair, we compute the Shannon entropy, $\sum_{i \in L} -p_i.log_2(p_i)$, where L={0,1} indicates the class labels, 1 for matching and 0 for non-matching, and p_i indicates the probability with which the pair is a match. Those pairs which have the highest entropy are selected in each active learning (AL) iteration [2, 60]. Interestingly, the pairs whose probabilities are highly ambiguous (close to 0.5), yield the highest entropy of 1.0. Therefore, for probabilistic classifiers, entropy-based selection can be seen as an analogous variant of margin-based selection which selects examples that have the least distance from the class-separator probability 0.5.

A.1.2 Query-by-committee

Algorithm 7 shows how QBC [40] is implemented. In each AL iteration, we create a committee of classifiers trained on several sampled sets of training data drawn with replacement (lines 2 and 3). The size of each training sample set is equal to the size of the training set of sampled pairs accumulated until the current AL iteration. Subsequently, we compute the labeling variance among the classifier committee for each unlabeled pair (lines 5 to 10) in $P_{remaining}$ and find the top-k remaining pairs with the highest variance (line 12). The committee variance for each unlabeled pair is computed based on the fraction of classifiers that label the pair as 'matching' and 'non-matching' (lines 6 to 8). It is essential to note that we train the committee in parallel to save on example selection time to give a fair advantage to the QBC baseline.

Algorithm 7 QBC(P_{train}, $P_{remaining}$, $batchSize$, $committeeSize$)

```
1: init $P_{sel} = \{\}$
2: $Samples \leftarrow$ sampleWithReplacement($P_{train}$, $committeeSize$)
3: $committee \leftarrow$ trainClassifiers($Samples$)
4: $scores \leftarrow []$
5: for $j: 0$ to $|P_{remaining}| - 1$ do
6:     $posModels \leftarrow$ findClassifiers($P_{remaining}[j]$, $committee$, '$matching$')
7:     $negModels \leftarrow$ findClassifiers($P_{remaining}[j]$, $committee$, '$non\text{-}matching$')
8:     $variance \leftarrow \frac{|posModels| \times |negModels|}{committeeSize}$
9:     $scores[j] \leftarrow variance$
10: end for
11: $sortedPairs \leftarrow$ Sort $P_{remaining}$ DESC on $scores$
12: $P_{sel} \leftarrow$ choose $batchSize$ pairs with the highest score from $sortedPairs$
13: return $P_{sel}$
```

A.1.3 OASIS

Algorithm 8 shows the working of OASIS [38] which is an adaptive importance weighted sampling (AIS) technique that was originally developed as an F1-score estimator for Entity Matching (EM). We adapt it to GNN-based semantic schema matching for ontologies by including a few implementation-level changes, without altering the fundamental sample selection mechanism described in the original paper [38].

Algorithm 8 OASIS($P_{remaining}$, $batchSize$, $model$, $n_{cluster}$)

```
1: init $P_{sel} = \{\}$
2: $Probs_{rem} \leftarrow model.PredProb(P_{remaining})$
3: $strata \leftarrow$ equiWidthBinning($P_{remaining}$, $Probs_{rem}$, $n_{cluster}$)
4: $weights \leftarrow$ computeImportanceWeights($strata$)
5: $sortedStrata \leftarrow$ Sort $strata$ DESC on $weights$
6: $P_{sel} \leftarrow$ choose $batchSize$ pairs at random from $sortedStrata$
7: return $P_{sel}$
```

- OASIS replaces the discriminative model (classifier) with a generative model. We instead use GNNs as the discriminative model for a consistent implementation of all baselines.
- OASIS creates a stratum (or a grouping) by using equi-width binning on the record similarities between a pair of entities computed using string similarity functions [58]. Since we work on schema graphs, we use the predicted matching probabilities for concept pairs as the similarity scores upon which we employ equi-width binning to create the strata (line 2 in Algorithm 8).
- OASIS creates the strata only once and uses them through all the AL iterations to select one sample in each AL iteration. We extend OASIS to perform batched sampling and while doing so, we encountered several empty strata

in the later AL iterations, that led to highly sub-optimal matching quality. Therefore, we refine the strata in each AL iteration by discarding the unlabeled pairs that have been labeled by the oracle. This was done to give more advantage to our implementation of the OASIS baseline.

In our implementation, OASIS creates the strata on the remaining unlabeled pairs in each AL iteration (line 3 in Algorithm 8). It then computes the importance weights for each stratum (line 4) according to Equation 12 in the original OASIS paper [38], which balances an exploration vs. exploitation trade-off. It assigns more weight to the largest representative stratum that contains the most number of unlabeled pairs (exploitation) with ϵ probability vs. using an asymptotic formula for stratum importance computation (exploitation) with (1-ϵ) probability. We used the default settings from the original paper for the parameters used in importance weight computation such as the estimated F-measure weight (α=0.5) and ϵ=0.001. After the computation of importance weights, we sort the strata and pick the top-k unlabeled pairs for sampling at random from the topmost strata with the highest importance weights, where k=$batchSize$.

A.1.4 Degree and centrality-based selection

While degree-based selection [6] chooses a batch of concept pairs with the highest sum of node degrees, centrality-based selection chooses those node pairs whose constituent nodes have the least distance from their respective ontology cluster centroids. The stratified variants of these selection strategies diversify the selection by uniformly selecting a fixed number of locally best candidate pairs within each ontology cluster.

A.2 Availability

The techniques implemented in ALFA are being patented. The BANK-KAFS dataset contains sensitive data and is proprietary. Hence both the source code for ALFA and the BANK-KAFS dataset cannot be made publicly available at this time. However, in order to help with reproducibility of our proposed solution, we are making the pre-processed input files, input embeddings and the output logs corresponding to the other datasets used in our paper, CMT-CONF, HUMAN-MOUSE anatomy and FMA-NCI available. Note that the original ontologies of these three datasets are available at [44], [45] and [26]. Following is a description of the items available on our GitHub repository [1].

- **Ontology_Node_Id_Label_Mappings** The mapping files contain the Node IDs and OWL class labels i.e., the names of all the schema concepts which include numerical (measures) and categorical (dimensions) attributes as well as the data properties within the unified ontology graph that combines both the input ontologies into a single graph. All the concepts and data properties in the unified ontology graph are further annotated by the values 1 and 2 respectively to distinguish between them.
- **Ontology_Edge_List** This contains all the edges between the nodes that correspond to the object properties in the unified ontology graph.
- **Ontology_Node_Embeddings** These are the Universal Sentence Encodings (USE) for all the nodes in the unified ontology graph. The ontology graph annotated with the USE embeddings for its nodes is fed as input to the GNN, along with the training data consisting of a progressively increasing subset of oracle-labeled node pairs in each active learning iteration.
- **Pos_neg_pairs** These are the ground truth labels for the node pairs in the unified ontology graph. Note that we used negative sampling from Qin et al. [53] to draw hard-to-classify non-matching pairs. All the matching node pairs referring to the same ontology concept are labeled 1 and the non-matching pairs are labeled 0.
- **Seed_pairs** These are the seed pairs used to train the initial GNN model prior to active learning.
- **Remaining_pairs** These are the remaining pairs treated as the unlabeled set for the purpose of active learning.
- **Logs** These contain the active learning progressive F1-scores and latencies in each iteration for all the approaches compared in the paper.

References

1. ALFA: Active Learning for Graph Neural Network-based Semantic Schema Alignment (2023). https://github.com/vamsikrishna1902/ALFA
2. Aggarwal, C.C., Kong, X., Gu, Q., Han, J., Yu, P.S.: Active learning: a survey. In: Aggarwal, C.C. (Ed.) Data Classification: Algorithms and Applications, pp. 571–606. CRC Press (2014). https://doi.org/10.1201/b17320-23
3. Alsentzer, E.: ClinicalBERT—Bio + Clinical BERT Model. https://huggingface.co/emilyalsentzer/Bio_ClinicalBERT (2020)
4. Atzeni, P., Bellomarini, L., Papotti, P., Torlone, R.: Meta-mappings for schema mapping reuse. Proc. VLDB Endow. **12**(5), 557–569 (2019). https://doi.org/10.14778/3303753.3303761
5. Bento, A., Zouaq, A., Gagnon, M.: Ontology matching using convolutional neural networks. In: Proceedings of The 12th Language Resources and Evaluation Conference (LREC), pp. 5648–5653. European Language Resources Association (2020)
6. Berrendorf, M., Faerman, E., Tresp, V.: Active learning for entity alignment. In: Hiemstra, D., Moens, M., Mothe, J., Perego, R., Potthast, M., Sebastiani, F. (Eds.) Advances in Information Retrieval—43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28–April 1, 2021, Proceedings, Part I, Lecture Notes in Computer Science, vol. 12656, pp. 48–62. Springer (2021). https://doi.org/10.1007/978-3-030-72113-8_4
7. Beygelzimer, A., Dasgupta, S., Langford, J.: Importance weighted active learning. In: Danyluk, A.P., Bottou, L., Littman, M.L.

(Eds.) Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14–18, 2009, ACM International Conference Proceeding Series, vol. 382, pp. 49–56. ACM (2009). https://doi.org/10.1145/1553374.1553381

8. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Trans. Assoc. Comput. Linguist. **5**, 135–146 (2017). https://doi.org/10.1162/tacl_a_00051
9. Cai, H., Zheng, V.W., Chang, K.C.C.: Active learning for graph embedding. arXiv preprint arXiv:1705.05085 (2017)
10. Cer, D., Yang, Y., Kong, S., Hua, N., Limtiaco, N., John, R.S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y., Strope, B., Kurzweil, R.: Universal sentence encoder. CoRR arXiv:1803.11175 (2018)
11. Cer, D., Yang, Y., Kong, S., et al.: https://tfhub.dev/google/universal-sentence-encoder-large/5
12. Cesa-Bianchi, N., Gentile, C., Vitale, F., Zappella, G.: A linear time active learning algorithm for link classification. In: Bartlett, P.L., Pereira, F.C.N., Burges, C.J.C., Bottou, L., Weinberger, Q. (Eds.) Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3–6, 2012, Lake Tahoe, Nevada, United States, pp. 1619–1627 (2012). https://proceedings.neurips.cc/paper/2012/hash/bf62768ca46b6c3b5bea9515d1a1fc45-Abstract.html
13. Chen, J., Jiménez-Ruiz, E., Horrocks, I., Antonyrajah, D., Hadian, A., Lee, J.: Augmenting ontology alignment by semantic embedding and distant supervision. In: Verborgh, R., Hose, K., Paulheim, H., Champin, P., Maleshkova, M., Corcho, Ó., Ristoski, P., Alam, M. (Eds.) The Semantic Web—18th International Conference, ESWC 2021, Virtual Event, June 6–10, 2021, Proceedings, Lecture Notes in Computer Science, vol. 12731, pp. 392–408. Springer (2021). https://doi.org/10.1007/978-3-030-77385-4_23
14. Chen, X., Wang, T.: Combining active learning and semi-supervised learning by using selective label spreading. In: 2017 IEEE International Conference on Data Mining Workshops (ICDMW), pp. 850–857 (2017). https://doi.org/10.1109/ICDMW.2017.154
15. Cheng, A., Zhou, C., Yang, H., Wu, J., Li, L., Tan, J., Guo, L.: Deep active learning for anchor user prediction. In: Kraus, S. (Ed.) Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10–16, 2019, pp. 2151–2157. ijcai.org (2019). https://doi.org/10.24963/ijcai.2019/298
16. Cohn, D., Atlas, L., Ladner, R.: Improving generalization with active learning. Mach. Learn. **66**, 201–221 (1994)
17. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4171–4186. Association for Computational Linguistics, Minneapolis (2019). https://doi.org/10.18653/v1/N19-1423
18. Faria, D., Pesquita, C., Santos, E., Palmonari, M., Cruz, I.F., Couto, F.M.: The agreementmakerlight ontology matching system. In: Meersman, R., Panetto, H., Dillon, T.S., Eder, J., Bellahsene, Z., Ritter, N., Leenheer, P.D., Dou, D. (Eds.) On the Move to Meaningful Internet Systems: OTM 2013 Conferences—Confederated International Conferences: CoopIS, DOA-Trusted Cloud, and ODBASE 2013, Graz, Austria, September 9–13, 2013. Proceedings, Lecture Notes in Computer Science, vol. 8185, pp. 527–541. Springer (2013). https://doi.org/10.1007/978-3-642-41030-7_38
19. Firmani, D., Saha, B., Srivastava, D.: Online entity resolution using an oracle. Proc. VLDB Endow. **9**(5), 384–395 (2016). https://doi.org/10.14778/2876473.2876474
20. Freund, Y., Seung, H., Shamir, E., Tishby, N.: Selective sampling using the query by committee algorithm. Mach. Learn. **28**(2–3), 133–168 (1997)
21. Gal, A., Roitman, H., Sagi, T.: From diversity-based prediction to better ontology & schema matching. In: Bourdeau, J., Hendler, J., Nkambou, R., Horrocks, I., Zhao, B.Y. (Eds.) Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11–15, 2016, pp. 1145–1155. ACM (2016). https://doi.org/10.1145/2872427.2882999
22. Galhotra, S., Firmani, D., Saha, B., Srivastava, D.: Robust entity resolution using random graphs. In: Proceedings of the 2018 International Conference on Management of Data, SIGMOD'18, pp. 3–18. ACM, New York (2018). https://doi.org/10.1145/3183713.3183755
23. Gao, L., Yang, H., Zhou, C., Wu, J., Pan, S., Hu, Y.: Active discriminative network representation learning. In: IJCAI International Joint Conference on Artificial Intelligence (2018)
24. Guo, Y., Greiner, R.: Optimistic active-learning using mutual information. In: Veloso, M.M. (Ed.) IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6–12, 2007, pp. 823–829 (2007). http://ijcai.org/Proceedings/07/Papers/132.pdf
25. Hao, J., Lei, C., Efthymiou, V., Quamar, A., Özcan, F., Sun, Y., Wang, W.: MEDTO: medical data to ontology matching using hybrid graph neural networks. In: Zhu, F., Ooi, B.C., Miao, C. (Eds.) KDD'21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14–18, 2021, pp. 2946–2954. ACM (2021). https://doi.org/10.1145/3447548.3467138
26. He, Y., Chen, J., Antonyrajah, D., Horrocks, I.: Bertmap: a BERT-based ontology alignment system. https://github.com/KRR-Oxford/BERTMap
27. He, Y., Chen, J., Antonyrajah, D., Horrocks, I.: Bertmap: a BERT-based ontology alignment system. In: Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22–March 1, 2022, pp. 5684–5691. AAAI Press (2022). https://ojs.aaai.org/index.php/AAAI/article/view/20510
28. Hernández, M.A., Miller, R.J., Haas, L.M.: Clio: a semi-automatic tool for schema mapping. In: Mehrotra, S., Sellis, T.K. (Eds.) Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data, Santa Barbara, CA, USA, May 21–24, 2001, p. 607. ACM (2001). https://doi.org/10.1145/375663.375767
29. Jain, A., Sarawagi, S., Sen, P.: Deep indexed active learning for matching heterogeneous entity representations. Proc. VLDB Endow. **15**(1), 31–45 (2021). https://doi.org/10.14778/3485450.3485455
30. Jiménez-Ruiz, E., Grau, B.C.: Logmap: logic-based and scalable ontology matching. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N.F., Blomqvist, E. (Eds.) The Semantic Web—ISWC 2011—10th International Semantic Web Conference, Bonn, Germany, October 23–27, 2011, Proceedings, Part I, Lecture Notes in Computer Science, vol. 7031, pp. 273–288. Springer, Berlin (2011). https://doi.org/10.1007/978-3-642-25073-6_18
31. Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with GPUs. IEEE Trans. Big Data **7**(3), 535–547 (2019)
32. Jurisch, M., Igler, B.: Graph-convolution-based classification for ontology alignment change prediction. In: Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG2019) Co-located with (ESWC 2019), vol. 2377, pp. 11–20. CEUR-WS.org (2019)
33. Kasai, J., Qian, K., Gurajada, S., Li, Y., Popa, L.: Low-resource deep entity resolution with transfer and active learning. In: Korho-

nen, A., Traum, D.R., Màrquez, L. (Eds.) Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28–August 2, 2019, Volume 1: Long Papers, pp. 5851–5861. Association for Computational Linguistics (2019). https://doi.org/10.18653/v1/p19-1586

34. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24–26, 2017, Conference Track Proceedings. OpenReview.net (2017). https://openreview.net/forum?id=SJU4ayYgl
35. Konda, P., Das, S.C., Gory, P.S., Doan, A., Ardalan, A., Ballard, J.R., Li, H., Panahi, F., Zhang, H., Naughton, J.F., Prasad, S., Krishnan, G., Deep, R., Raghavendra, V.: Magellan: Toward building entity matching management systems. PVLDB **9**(12), 1197–1208 (2016). https://doi.org/10.14778/2994509.2994535
36. Lloyd, S.P.: Least squares quantization in PCM. IEEE Trans. Inf. Theory **28**(2), 129–136 (1982). https://doi.org/10.1109/TIT.1982.1056489
37. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: Cam, L.M.L., Neyman, J. (Eds.) Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297. University of California Press (1967)
38. Marchant, N.G., Rubinstein, B.I.P.: In search of an entity resolution OASIS: optimal asymptotic sequential importance sampling. Proc. VLDB Endow. **10**(11), 1322–1333 (2017). https://doi.org/10.14778/3137628.3137642
39. Meduri, V.V., Popa, L., Sen, P., Sarwat, M.: A comprehensive benchmark framework for active learning methods in entity matching. In: Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, Online Conference [Portland, OR, USA], June 14–19, 2020, pp. 1133–1147 (2020). https://doi.org/10.1145/3318464.3380597
40. Mozafari, B., Sarkar, P., Franklin, M., Jordan, M., Madden, S.: Scaling up crowd-sourcing to very large datasets: a case for active learning. PVLDB **8**(2), 125–136 (2014)
41. Mudgal, S., Li, H., Rekatsinas, T., Doan, A., Park, Y., Krishnan, G., Deep, R., Arcaute, E., Raghavendra, V.: Deep learning for entity matching: a design space exploration. In: Proceedings of the 2018 International Conference on Management of Data, SIGMOD'18, pp. 19–34. ACM, New York (2018). https://doi.org/10.1145/3183713.3196926
42. NLTK Word Tokenizer. https://www.nltk.org/api/nltk.tokenize.punkt.html
43. Nguyen, T.T., Sanner, S.: Algorithms for direct 0-1 loss optimization in binary classification. In: Proceedings of the 30th International Conference on International Conference on Machine Learning—Volume 28, ICML'13, pp. III–1085–III–1093. JMLR.org (2013). http://dl.acm.org/citation.cfm?id=3042817.3043058
44. OAEI Conference Dataset. http://oaei.ontologymatching.org/2021/conference/ (2021)
45. OAEI Human-Mouse Anatomy Dataset. http://oaei.ontologymatching.org/2021/anatomy/ (2021)
46. OAEI: OAEI 2021::Large BioMed Track. https://www.cs.ox.ac.uk/isg/projects/SEALS/oaei/2021/ (2021)
47. Ostapuk, N., Yang, J., Cudré-Mauroux, P.: Activelink: deep active learning for link prediction in knowledge graphs. In: Liu, L., White, R.W., Mantrach, A., Silvestri, F., McAuley, J.J., Baeza-Yates, R., Zia, L. (Eds.) The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13–17, 2019, pp. 1398–1408. ACM (2019). https://doi.org/10.1145/3308558.3313620
48. Papadakis, G., Fisichella, M., Schoger, F., Mandilaras, G., Augsten, N., Nejdl, W.: Benchmarking filtering techniques for entity resolution. In: 2023 IEEE 39th International Conference on Data Engineering (ICDE), pp. 653–666 (2023). https://doi.org/10.1109/ICDE55515.2023.00389
49. pinecone: Nearest Neighbor Indexes for Similarity Search. https://www.pinecone.io/learn/series/faiss/vector-indexes/ (2019)
50. Qian, K., Popa, L., Sen, P.: Active learning for large-scale entity resolution. In: Lim, E., Winslett, M., Sanderson, M., Fu, A.W., Sun, J., Culpepper, J.S., Lo, E., Ho, J.C., Donato, D., Agrawal, R., Zheng, Y., Castillo, C., Sun, A., Tseng, V.S., Li, C. (Eds.) Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 6–10, 2017, pp. 1379–1388. ACM (2017). https://doi.org/10.1145/3132847.3132949
51. Qian, K., Popa, L., Sen, P.: Systemer: a human-in-the-loop system for explainable entity resolution. Proc. VLDB Endow. **12**(12), 1794–1797 (2019). https://doi.org/10.14778/3352063.3352068
52. Qian, K., Raman, P.C., Li, Y., Popa, L.: Learning structured representations of entity names using active learning and weak supervision. In: Webber, B., Cohn, T., He, Y., Liu, Y. (Eds.) Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16–20, 2020, pp. 6376–6383. Association for Computational Linguistics (2020). https://doi.org/10.18653/v1/2020.emnlp-main.517
53. Qin, X., Sheikh, N., Reinwald, B., Wu, L.: Relation-aware graph attention model with adaptive self-adversarial training. In: Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2–9, 2021, pp. 9368–9376. AAAI Press (2021). https://ojs.aaai.org/index.php/AAAI/article/view/17129
54. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB J. **10**(4), 334–350 (2001). https://doi.org/10.1007/s007780100057
55. Ratner, A., Bach, S.H., Ehrenberg, H., Fries, J., Wu, S., Ré, C.: Snorkel: rapid training data creation with weak supervision. Proc. VLDB Endow. **11**(3), 269–282 (2017). https://doi.org/10.14778/3157794.3157797
56. Rousseeuw, P.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math. **20**(1), 53–65 (1987)
57. Roy, N., McCallum, A.: Toward optimal active learning through sampling estimation of error reduction. In: Brodley, C.E., Danyluk, A.P. (Eds.) Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28–July 1, 2001, pp. 441–448. Morgan Kaufmann (2001)
58. Simmetrics Java Library. https://github.com/Simmetrics/simmetrics
59. scikit learn: sklearn.metrics.silhouette-score. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html (2007)
60. Sarawagi, S., Bhamidipaty, A.: Interactive deduplication using active learning. In: KDD, pp. 269–278 (2002)
61. Satopaa, V., Albrecht, J.R., Irwin, D.E., Raghavan, B.: Finding a "kneedle" in a haystack: detecting knee points in system behavior. In: ICDCS Workshops, pp. 166–171 (2011)
62. Schlichtkrull, M.S., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. CoRR **abs/1703.06103** (2017). http://arxiv.org/abs/1703.06103
63. Seung, H., Opper, M., Sompolinsky, H.: Query by committee. In: Workshop on COLT, pp. 287–294 (1992)
64. Shraga, R., Gal, A., Roitman, H.: Adnev: cross-domain schema matching using deep similarity matrix adjustment and evaluation. Proc. VLDB Endow. **13**(9), 1401–1415 (2020). https://doi.org/10.14778/3397230.3397237

65. Snyder, T.: The Benefits of Machine Learning for Large Scale Schema Mapping. https://tinyurl.com/4nxmkevr (2019)
66. ten Cate, B., Kolaitis, P.G., Qian, K., Tan, W.: Active learning of GAV schema mappings. In: den Bussche, J.V., Arenas, M. (Eds.) Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Houston, TX, USA, June 10–15, 2018, pp. 355–368. ACM (2018). https://doi.org/10.1145/3196959.3196974
67. Thirumuruganathan, S., Li, H., Tang, N., Ouzzani, M., Govind, Y., Paulsen, D., Fung, G., Doan, A.: Deep learning for blocking in entity matching: a design space exploration. Proc. VLDB Endow. **14**(11), 2459–2472 (2021). https://doi.org/10.14778/3476249.3476294
68. Vesdapunt, N., Bellare, K., Dalvi, N.N.: Crowdsourcing algorithms for entity resolution. Proc. VLDB Endow. **7**(12), 1071–1082 (2014). https://doi.org/10.14778/2732977.2732982
69. Wang, J., Kraska, T., Franklin, M.J., Feng, J.: CrowdER: crowdsourcing entity resolution. PVLDB **5**(11), 1483–1494 (2012)
70. Wang, J., Li, G., Yu, J.X., Feng, J.: Entity matching: how similar is similar. Proc. VLDB Endow. **4**(10), 622–633 (2011). https://doi.org/10.14778/2021017.2021020
71. Wang, Z., Cruz, I.F.: Agreementmakerdeep results for OAEI 2021. In: Shvaiko, P., Euzenat, J., Jiménez-Ruiz, E., Hassanzadeh, O., Trojahn, C. (Eds.) Proceedings of the 16th International Workshop on Ontology Matching co-located with the 20th International Semantic Web Conference (ISWC 2021), Virtual conference, October 25, 2021, CEUR Workshop Proceedings, vol. 3063, pp. 124–130. CEUR-WS.org (2021). http://ceur-ws.org/Vol-3063/oaei21_paper3.pdf
72. Wu, R., Chaba, S., Sawlani, S., Chu, X., Thirumuruganathan, S.: Zeroer: entity resolution using zero labeled examples. In: Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, Online Conference [Portland, OR, USA], June 14–19, 2020, pp. 1149–1164 (2020). https://doi.org/10.1145/3318464.3389743
73. Wu, Y., Xu, Y., Singh, A., Yang, Y., Dubrawski, A.: Active learning for graph neural networks via node feature propagation. arXiv preprint arXiv:1910.07567 (2019)
74. Yan, Y., Liu, L., Ban, Y., Jing, B., Tong, H.: Dynamic knowledge graph alignment. In: Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2–9, 2021, pp. 4564–4572. AAAI Press (2021). https://ojs.aaai.org/index.php/AAAI/article/view/16585
75. Zhang, R., Trisedya, B.D., Li, M., Jiang, Y., Qi, J.: A benchmark and comprehensive survey on knowledge graph entity alignment via representation learning. VLDB J. **31**(5), 1143–1168 (2022). https://doi.org/10.1007/s00778-022-00747-z
76. Zhang, W., Shen, Y., Li, Y., Chen, L., Yang, Z., Cui, B.: ALG: fast and accurate active learning framework for graph convolutional networks. In: Li, G., Li, Z., Idreos, S., Srivastava, D. (Eds.) SIGMOD'21: International Conference on Management of Data, Virtual Event, China, June 20–25, 2021, pp. 2366–2374. ACM (2021). https://doi.org/10.1145/3448016.3457325
77. Zhang, W., Wei, H., Sisman, B., Dong, X.L., Faloutsos, C., Page, D.: Autoblock: a hands-off blocking framework for entity matching. In: Caverlee, J., Hu, X.B., Lalmas, M., Wang, W. (Eds.) WSDM'20: The Thirteenth ACM International Conference on Web Search and Data Mining, Houston, TX, USA, February 3–7, 2020, pp. 744–752. ACM (2020). https://doi.org/10.1145/3336191.3371813