CS473 Project – Stage 2

Stage 2: In stage 2, your group will develop additional approaches to the ERD grade prediction tool.

Goals for this project:

The project is designed to promote self-directed learning and to provide valuable teamwork experience through collaborative problem-solving. It also aims to strengthen students' research and communication skills, including both technical writing and oral presentation.

In addition, the project gives students the opportunity to apply and practice key course concepts, such as text and graph processing, TF–IDF and word embeddings, and K-nearest neighbors (KNN) for similarity and prediction tasks. These skills collectively help students bridge theoretical knowledge with real-world data analysis and modeling challenges.

Grading: We will use RMSE, as in Stage 1. However, the highest (worst) 15% of your predictions will be excluded from the calculation. This adjustment accounts for cases where a test ERD has no similar graded ERDs and helps mitigate possible manual grading inconsistencies (e.g., two similar ERDs receiving different grades from the TA).

Prediction rule (KNN)

Use **KNN regression** on the training set:

- Similarity $si \in [0,1]$
- Find the top K similar graded ERDs
- Prediction using weighted average.
- If similarity of an ERD is below some threshold q then you may discard the ERD (e.g., use K=2 instead of K=3).
- If all similarities are below threshold q, then fall back to the global training dataset mean.

Text Normalization: Before generating vectors/embeddings for attributes or entity names, you should apply text normalization to make similar terms consistent and easier to compare. Text normalization converts messy, inconsistent labels into a clean, uniform form. You can start by lowercasing all words and replacing special characters like underscores (_) or hyphens (-) with spaces. Then, split camelCase or PascalCase names (e.g., studentName → student name) and remove extra spaces or punctuation. It also helps to standardize abbreviations and synonyms, such as mapping no, num, or id to "number" or "identifier." In TF-IDF approaches, you can apply stemming or lemmatization to reduce words to a common base form (e.g., students → student, enrolled → enroll). However, if you are using word embeddings, such as BERT and Sentence-BERT, be aware that stemming or lemmatization may hurt performance as some stems may not be recognized. Finally, you can use edit distance–based matching to fix minor typos (e.g., studnt → student) before embedding (see edit distance in section 2.5 in the NLP textbook). One approach to do so is checking whether a word is rare in the collection (possible

typo, such as studnt) and replace it with the closest popular word in the collection (e.g., student).

These normalization steps will make your model more robust by ensuring that semantically similar attributes/names produce similar vectors.

Approaches:

You will develop the following five approaches for grade prediction. Select the best approach for the final version of your tool that you will use during the grading meeting.

Approach 1 (TF_IDF_KNN): Same approach 1 from stage 1. The input files will use the JSON format we used in manual labeling.

Approach 2 (embeddings_KNN): Same approach 2 from stage 1. The input files will use the JSON format we used in manual labeling.

Approach 3 (Graph2vec_embeddings_KNN or Graph2vec_TF-IDF_KNN). Graph2Vec converts each ERD graph into a single vector. Those vectors can be used to compute the graph similarity of two ERDs. You may represent an ERD as a graph of connected nodes with node types: entity, weak entity, relationship, identifying relationship, relationship attribute, and cardinality. Node labels must include at least the node type. You may also embed other information in node type, such as name for entities (e.g., node label for an entity is "entity:Employee" or "weakEntity:dependent").

Given that Graph2vec focuses only on the graph structure, you will develop a hybrid similarity model that combines Graph2ve similarity with text similarity from approach 1 or 2 as follows:

Similarity = A * Graph2vec-similarity + (1-A) embeddings similarity.
or
Similarity = A * Graph2vec-similarity + (1-A) TF-IDF similarity.

Where 0 <= A <= 1 is a hyperparameter that can be tuned.

Finally use KNN to find similar ERDs and predict the grade.

Approach 4 (custom_graph_text): Approach 4 is a custom module that will consider both text and graph features of ERDS.

1. Entity similarity: The entity similarity between two entities/weak entities E1 and E2 is computed as follows:

   entity similarity ES(E1, E2) = A * type similarity
   + B * name similarity (e.g., cosine of word embedding vectors)
   + C * similarity of the number of attributes
   + D * similarity of the attributes in the two entities (can compute average of attributes' word embeddings in each entity then compute cosine)

where:
- A + B + C + D =1, are hyperparameters that control the importance of each component.
- Type = entity or weak entity.
- type similarity of {entity, entity} = similarity of {weak-entity, weak-entity} = 1, while type similarity of {entity, weak-entity} = some constant c, c < 1 (e.g., c=0.5).

Notes:

- Key attributes are considered as regular attributes, but they can be considered as an additional separate component.
- Try different embeddings models, such as word2vec, Sentence-BERT, and Sentence-Roberta.

2. All Entities similarity: The similarity between entities {e1, e2, … en} in ERD1 and the entities {t1, t2, … tn} in ERD2 can be computed as follows:
    a. Compute the similarity of all possible pairs (e, t), where e is in ERD1 and t is in ERD2.
    b. Find the best matching pairs (e,t), where e is an entity or weak entity in ERD1 and t is an entity or weak entity in ERD2. A greedy approach may be sufficient for this task: start by pairing the entities e and t with the highest similarity score, then continue with the next highest pair, and so on.
    c. If the number of entities in ERD1 is greater than in ERD2, some entities in ERD1 will remain unmatched, resulting in pairs of the form (e,NULL), where an entity has no corresponding match in the other ERD.
    d. Compute the similarity between the matched pairs of entities (e,t). The similarity (e, NULL) = 0.

Option 1 – All Entities Similarity = The average similarity across all matched pairs.

Option 2 – All Entities Similarity (excluding NULL) = The average similarity across all matched pairs, excluding those of the form (e, NULL).

Option 2 avoids the potentially large penalty caused by differences in the number of entities.

For example, suppose ERD1 and ERD2 share four identical entities, but ERD2 has one additional entity that does not appear in ERD1. In this case, the average similarity in Option 1 would be 80%, while in Option 2 it would be 100%.

Note that Option 2 still accounts for differences in the number of entities in Step 5 below.

3. Relationship/identifying relationship similarity.
   Given two relationships rel1 and rel2, where each has the following:

- a type (relationship or identifying relationship)
- Arity (number of entities/weak entities involved in this relationship) (i.e., 2 for binary, 3 for ternary relationship, 4 for 4-way relationship, … etc.)

- A list of references for entities/weak entities involved in this relationship.
- number of relationship attributes (number of ovals connected to this relationship/diamond)
- a list of the relationship attributes. The list of attributes can be represented using an average of the word embeddings of those attributes (i.e., one vector for all attributes).
- For a binary relationship, maximum cardinality of the relationship sides (this can be NN, 1N, N1, 11, Unknown). Note that CS348 students are not required to specify cardinalities for ternary and other N-way relationships.
- For a binary relationship, minimum cardinality of the relationship sides (this can be 00, 01, 10, 11, Unknown).

The similarity of rel1 and rel2 is computed as
similarity S(rel1, rel2) =

T * type similarity (1 for similar types or less than 1 for different types)
+ U * Arity similarity
+ V * similarity of participating entities (based on matchings in step 2)
+ X * similarity of embeddings vectors of the attributes in rel1 and rel2
+ Y * similarity of max cardinality in rel1 and rel2
+ Z * similarity of min cardinality in rel1 and rel2

where T + U + V + X + Y + Z = 1 (likely U, V > T > X > Y > Z)
Notes:
- relationship label is not included as it is irrelevant (i.e., two relationships can be identical even if the labels are different).
- if the number of participating entities is different (e.g., one relationship is binary and the other is ternary), then discard the additional entity. This difference in the number of attributes is already included in the Arity component.

4. All relationships similarity. This will be computed by matching relationships in ERD1 and ERD2 the same way we match entities.

   Option 1 – All relationships Similarity = The average similarity across all matched pairs.
   Option 2 – All relationships Similarity (excluding NULL) = The average similarity across all matched pairs, excluding those of the form (r, NULL).

5. Similarity of additional ERD features. Represent additional properties of ERDs as a vector and compute the similarity. Those properties include number of entities, number of weak entities, number of relationship attributes, number of binary relationships, number of binary identifying relationships, number of n-way relationships (ternary, 4-way, or more), and number of n-way identifying relationships.

6. ERD similarity. The similarity of ERD1 and ERD2 is computed as follows:

   ERD similarity = α * All Entities similarity + β * All relationships similarity + λ * Similarity of

additional ERD features

Where $\alpha + \beta + \lambda = 1$

7. Use KNN regression for grade prediction.

Approach 5 (app5) (optional): You are allowed to modify or use any new approaches for grade prediction. Your team is still required to develop approaches 1 to 4.

Research Report (2 to 3 pages): Working on this tool and having some evaluation collections allow us to investigate some research questions that can help develop similar grade-prediction and diagram-processing tools. Discuss the following research questions by comparing the performance of the tool when using different configurations.

1. RQ1: Which model is better (1 to 5)?
   Compare the performance of Approaches 1 through 5 across all datasets, and present the results in a graph. Discuss any interesting observations (e.g., why embeddings are better/worse than TF.IDF in a particular dataset, why graph2vec is better/worse than the custom model in some datasets)

2. RQ2: What is the relationship between T, the number of graded ERDs, and K, the number of neighbors in KNN? Using the best performing approach and all available datasets, try different values for T (10, 20, 30, … 100) and K (1, 2, 3, 4, 5, 10, 20, 50). Compute RMSE in each (T,K) pair and plot the results using a heatmap.

3. RQ3: Is there grade inconsistency in the datasets? Using the best-performing approach, identify the top 20 most similar pairs of ERDs and examine whether their grades are consistent. Discuss which dataset, if any, shows greater grade inconsistency (i.e., similar ERDs receiving noticeably different grades).

4. RQ4: A research question of your choice.
   Compare two different configurations that you think are interesting. Discuss any observations.
   Note: interesting configurations are the ones that have a large gap in performance or tend to perform differently in different datasets.

5. In two paragraphs, summarize two research papers in the education domain that address grade prediction or the autograding of short-answer or diagram-based questions (papers on essay grading are also acceptable). Avoid papers that focus on programming autograders. Example venues for relevant research include SIGCSE, FIE, Educational Data Mining (EDM), and Learning at Scale (L@S).

Insights about ERD grading (may help in building your tool):

1. For a particular ERD question, we usually get multiple correct answers that may contain different number of ERD components. For example, one correct ERD may contain 4 entities and three relationships while another correct ERD contains 3 entities, one weak entity, and one identifying relationship.
2. Relationship names (text in diamonds) are not important in grading and should not be used in computing ERD similarity. Two ERDs with a relationship connecting trip and bag are similar even if they use different names in the relationship.

Deliverables for Stage 2 and grading.

Stage 2 accounts for 60% of the project grade.

1. A copy of the source code. A readme file that includes instructions on how to use your tool. Submit that to Brightspace.

2. (40% of project grade) A grading meeting with your group TA. Grading meetings should be scheduled by the end of 12/10. **Extra credit:** There will be 2% extra credit for the top few teams (teams with outstanding low RMSE).

3. (8% of project grade) 2-3 pages research report.

4. (2% of project grade) self and peer review form. The form will be posted later in Ed. Note that we will consider peer reviews in your overall project grade.

5. (10% of project grade) A 20-30 minutes recorded presentation that includes:
    a. A short introduction about the problem.
    b. A short demo of using the tool (do not include training!).
    c. A discussion of the techniques you have used in stage2, including text preprocessing, vector representation, and model parameters.
    d. You may discuss RQ4.
    e. Lessons learned and future work, such as ideas you did not have time to try or things you wish you did differently.

    For the presentation, assume CS audience with some IR and ML background (i.e., imagine you are presenting to some senior CS students outside of this class).

Due Dates and Submission: Submit your report and presentation to Brightspace by 12/10 11:59 pm. Submit your code and self/peer review to Brightspace by 12/12 11:59 pm.

The presentations of a one/two teams will be included in the 12/12 lecture. Those teams may also present in class if they wish.