Deep Discourse Analysis for Generating Personalized Feedback in Intelligent Tutor Systems

Matt Grenander,[1] Robert Belfer,[2] Ekaterina Kochmar,[2,3] Iulian V. Serban,[2]
François St-Hilaire,[2] Jackie C. K. Cheung[4,5]

[1] University of Edinburgh
[2] Korbit Technologies
[3] University of Cambridge
[4] McGill University
[5] Mila
matt.grenander@ed.ac.uk, {robert, ekaterina, iulian, francois}@korbit.ai, jcheung@cs.mcgill.ca

Abstract

We explore creating automated, personalized feedback in an intelligent tutoring system (ITS). Our goal is to pinpoint correct and incorrect concepts in student answers in order to achieve better student learning gains. Although automatic methods for providing personalized feedback exist, they do not explicitly inform students about which concepts in their answers are correct or incorrect. Our approach involves decomposing students answers using neural discourse segmentation and classification techniques. This decomposition yields a relational graph over all discourse units covered by the reference solutions and student answers. We use this inferred relational graph structure and a neural classifier to match student answers with reference solutions and generate personalized feedback. Although the process is completely automated and data-driven, the personalized feedback generated is highly contextual, domain-aware and effectively targets each student's misconceptions and knowledge gaps. We test our method in a dialogue-based ITS and demonstrate that our approach results in high-quality feedback and significantly improved student learning gains.

Introduction

Intelligent tutoring systems (ITS) are AI-powered computer systems that provide personalized tutoring without human teachers (Nye, Graesser, and Hu 2014). ITSs have been shown to be more effective teachers than other computer tutoring methods and conventional classroom learning (Kulik and Fletcher 2016). Dialogue-based ITSs use natural language conversations and are particularly appealing as they closely simulate natural human-teacher interactions.

Developing dialogue-based ITSs face many challenges due to the difficulty of parsing and responding to natural text. One critical issue is their ability to provide effective automated feedback. Conventional tutoring typically includes direct guidance from a human teacher, and we would like to incorporate this teaching aspect into ITSs. In fact, previous work has shown that personalized feedback can considerably improve student learning outcomes in an ITS (Kochmar et al. 2020).

System: You are given a dataset of images of wildlife in Africa. You are tasked with building a model which can identify animals in the images. Is this a regression or classification problem? Try to explain why.
Student: I think it's a classification task.
System: 'it's a classification task' is correct. Try supplying a reason for this idea.
Student: It's a classification task, because each animal is a discrete class.
System: That's correct!

Table 1: An example interaction with the Korbit ITS, with our feedback mechanism in the red box. The feedback system identifies which concepts are correct and incorrect in the student answer attempt, then provides guidance to fix their answer. In this case, the student has forgotten to justify their answer and is prompted for the missing concept.

In this paper, we present a novel, automated feedback system based on deep learning approaches to discourse. Deep learning-based discourse models have produced successful results in recent years across several discourse tasks, such as segmentation (Wang, Li, and Yang 2018), classification (Dai and Huang 2018), parsing (Mabona et al. 2019) and tagging (Dasigi et al. 2017). These successes encourage us to explore applying neural discourse techniques in an ITS.

Our model takes advantage of established discourse frameworks such as Rhetorical Structure Theory (Mann and Thompson 1988) to break apart student solution attempts and analyze individual concepts within the student answer. After the analysis, the system personalizes its feedback to the student attempt and directly indicates which parts are correct, incorrect or missing. An example interaction with the feedback system is shown in Table .

Our method extends beyond simply applying existing discourse analysis techniques, since these methods cannot be used on their own for providing feedback in ITSs. In particular, we introduce a new structure called an exercise graph. We construct this graph using discourse components from reference solutions and previous student solution attempts

for each exercise in the ITS. Each graph represents a general solution to the problem, and the model uses them extensively to determine the best feedback.

Our approach contrasts considerably with existing feedback models, such as Rus, Niraula, and Banjade (2015) and Kochmar et al. (2020). While these systems can capably guide students towards fixing incorrect ideas, they cannot directly comment on which concepts the students have misunderstood. In this way, our system is a novel, practical approach for automatically providing feedback in dialogue-based ITSs.

We test our method using the Korbit ITS,[1] a large-scale, mixed-interface, dialogue-based ITS. In Korbit, students watch video lectures on STEM topics, then answer exercises on the topic. Students' attempted solutions are matched against numerous reference solutions using a machine learning algorithm. We compare student learning gains on Korbit using our feedback mechanism against both simple and strong baselines, and demonstrate that the feedback model results in the highest learning gains and feedback quality.

Our main contribution is an original, personalized feedback mechanism for dialogue-based ITSs able to directly indicate correct and incorrect concepts to students. Moreover, our model surpasses previous state-of-the-art benchmarks on discourse segmentation and classification tasks.

Feature Extraction

In order to reason about concepts within a student answer, the model first fragments their solution and extracts various discourse features. It decomposes the student's answer into segments called **EDUs** (described below), computes a **semantic embedding** for each segment and predicts a **discourse relation** for each pair of adjacent EDUs.

Segmentation

The segmenter's goal is to divide student input text into distinctive concepts. We follow the Rhetorical Structure Theory (RST) framework for modelling discourse structure (Mann and Thompson 1988) and use the RST Discourse Treebank (RST-DT) as our training data (Carlson, Marcu, and Okurovsky 2001). In RST, text is segmented into non-overlapping units called *elementary discourse units* (EDUs), each of which represents a basic unit of discourse.

We model discourse segmentation as a sequence classification task, where each word is assigned 1 if it denotes the start of an EDU and 0 otherwise. The input text s is first encoded with RoBERTa (Liu et al. 2019) into m embeddings where m is the number of subword tokens in the input:[2]

$$h_1, \ldots, h_m = \text{BERT}(s) \tag{1}$$

A binary logistic classifier then classifies each encoding h_i:

$$\hat{y}_i = \sigma(w_s^T h_i + b_s) \tag{2}$$

where w_s and b_s are learnable parameters.

[1] https://www.korbit.ai

[2] See Liu et al. (2019) and Wolf et al. (2019) for full implementation details.

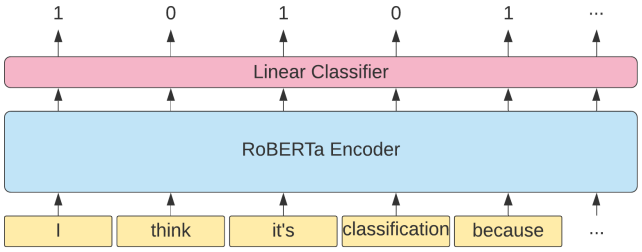

Figure 1: Our neural discourse segmentation model. The input text is tokenized and encoded with RoBERTa. The embeddings are then classified using a linear classifier. Predicting *1* indicates the start of a new EDU, while *0* denotes a continuation.

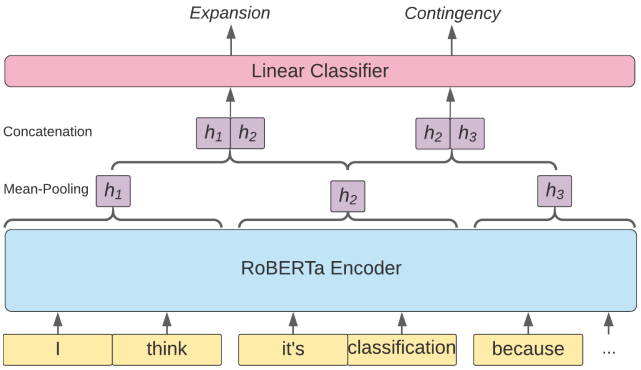

Figure 2: Our neural discourse classification model. After segmentation, the EDUs are encoded with a RoBERTa model, and the output embeddings within each EDU are averaged. These embeddings are then concatenated with adjacent embeddings and classified using a linear classifier. Note that the RoBERTa and linear classifier used here are separate from the ones used in segmentation.

Figure 1 provides an example of our segmentation approach. For the student input text "*I think it's a classification task because we are choosing between continuous values*", the segmenter should predict 1 for tokens *I*, *it's*, and *because*, and 0 for all other tokens. This labelling indicates the correct segments are "*I think*", "*it's a classification task*", and "*because we are choosing between continuous values*".

Semantic Equivalence

Providing feedback requires understanding which EDUs are semantically equivalent, so the model can match student EDUs to reference solution EDUs and reason about whether these concepts are correct or not. We use the open-source Sentence Transformers library (SBERT) from Reimers and Gurevych (2019).[3] The model is based on fine-tuning BERT-like models on semantic textual similarity (STS) (Cer et al. 2017) and natural language inference (NLI) data (Williams, Nangia, and Bowman 2018; Bowman et al. 2015). Given raw text, SBERT produces a vector representing semantic content. This vector can then be compared to other SBERT

[3] https://github.com/UKPLab/sentence-transformers

outputs using their cosine similarity. A high similarity score indicates the two texts are highly semantically related, while low or negative scores imply orthogonal or opposite meaning. Although the library is designed for sentences, we find that it works equally well for EDU sentence fragments. We use SBERT as pretrained by the authors and do not train the model further on our own data.

Discourse Classification

Our approach to feedback generation involves modelling the flow of ideas among typical reference answers, and suggesting which types of ideas are missing or incorrect. Accordingly, the model needs to understand the discourse relations between adjacent EDUs. We therefore train a classifier to predict such relations.

We follow relations defined in the PDTB 2.0 dataset (Prasad et al. 2008), which groups discourse relations into four top-level categories: *Temporal*, *Contingency*, *Comparison* and *Expansion*. PDTB 2.0 defines a classification task in which pairs of discourse units are labelled with the discourse relation between them. The PDTB annotation guide describes discourse units more indistinctly than EDUs in RST, but in practice they are often similar.

PDTB 2.0 contains *pairs* of discourse units, which do not closely resemble the arbitrarily long sequences returned by our neural segmenter model. We fix this mismatch by extracting paragraphs from PDTB containing consecutive discourse units, using the scripts from Dai and Huang (2018).[4] This format better resembles the sequences of EDUs in our data and is more suitable for our purposes.

Our discourse classifier model is similar to Dai and Huang (2018), though we replace their LSTM encoders with the RoBERTa encoder. Given a sequence of discourse units $(\mathrm{DU}_1, \ldots, \mathrm{DU}_L)$, we encode each discourse unit using RoBERTa, followed by mean-pooling:

$$h_{i,1}, \ldots, h_{i,|\,\mathrm{DU}_i\,|} = \mathrm{BERT}(\mathrm{DU}_i) \tag{3}$$

$$\overline{h}_i = \frac{1}{|\,\mathrm{DU}_i\,|} \sum_{j=1}^{|\,\mathrm{DU}_i\,|} h_{i,j}, \tag{4}$$

where $|\,\mathrm{DU}_i\,|$ is the number of tokens in discourse unit i.

The embeddings of each adjacent pair of discourse units are then concatenated and classified into relation categories using a logistic decoder.

$$\hat{y}_i = \sigma\left(w_d^T \left[\,\overline{h}_i, \overline{h}_{i+1}\right] + b_d\right) \tag{5}$$

where w_d and b_d are learnable parameters. Figure 2 illustrates this process.

PDTB 2.0 classifies relations as being either *implicit* or *explicit*, depending on whether there is a cue word such as *"because"* or *"while"*. Following Dai and Huang (2018), we use separate decoder parameters for implicit and explicit cases. At inference time, since we do not have access to the PDTB explicit/implicit labels, we first scan the discourse units for cue words before applying the relevant decoder.[5]

PDTB – RST-DT Mismatch Since the segmenter and discourse classifier models are trained using different discourse frameworks (RST-DT vs. PDTB), the two models will produce incompatible features. RST-DT-style EDUs are generally more fine-grained than in PDTB, and certain discourse features are not present in PDTB. For example, the sentence "*It uses a logistic function to model a binary dependent variable*" would not be segmented at all in PDTB, but would contain EDUs "*It uses a logistic function*" and "*to model a binary dependent variable*" in RST-DT.

We reconcile this data mismatch by fine-tuning the discourse classifier on RST-DT-style segments. We create a dataset with RST-DT-style segments and PDTB-style relations in several steps. First, we segment the PDTB development and test set with the RST-DT-trained segmenter. We then label the EDU boundaries as follows:

1. We map any relations corresponding to shared EDU boundaries, as many EDU boundaries are shared between the two frameworks.
2. We use cue word heuristics to automatically label EDU boundaries. For example, if we detect "*if*" or "*because*" near the EDU boundary, we label it as *Contingency*, or if we observe "*to*" we annotate it as *Expansion*.
3. We manually annotate any remaining unlabelled EDU boundaries.

In total, 40% of relations arise from shared EDU boundaries, 50% from cue word heuristics and 10% are labelled manually. After first training the discourse classifier on PDTB, we fine-tune it on the generated dataset.

We assess the fine-tuned model on 103 hand-annotated student solution attempt samples. We find that before fine-tuning on RST-DT-style EDUs, the model reaches 65.7% accuracy. After fine-tuning, the resulting classifier achieves 81.1% accuracy, indicating that it can accurately predict relations between RST-DT-style EDUs.

Feedback Model

Using the features from Section , we can identify student answer deficiencies and suggest improvements. For each exercise in the ITS, we construct and store a relational graph using reference solutions and all solution attempts across previous students. Each exercise graph represents what a standard solution should resemble for that exercise (Figure 3). We train a classifier to recognize which graph transitions are valid. Then, at inference time, we use a local search algorithm in conjunction with the classifier in order to determine what edits, if any, can transform the student's solution attempt into a better one. Finally, the model uses these results to generate feedback for the student.

Graph

The Korbit ITS allows us to collect previous student solution attempts, and we can combine these with the existing refer-

[4] https://github.com/ZeyuDai/paragraph-level_implicit_discourse_relation_classification

[5] The cue words are listed at https://github.com/korbit-ai/deep-discourse-feedback.

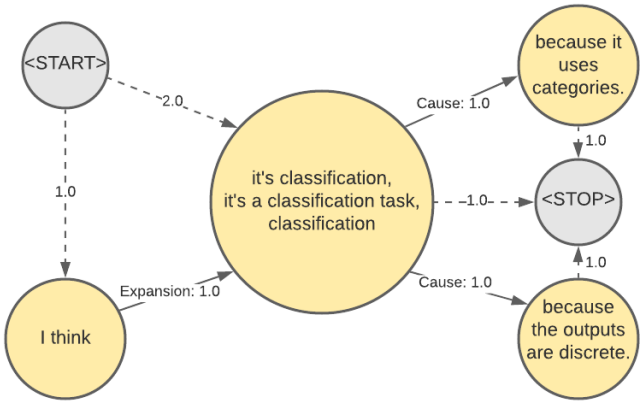

Figure 3: In this exercise graph example, the input texts are "*I think it's classification*", "*it's a classification task, because it uses categories*", and "*classification because the outputs are discrete.*" After the EDUs, semantic embeddings and discourse relations are extracted, the semantic embeddings are clustered, and weighted edges added as described in Section . Larger clusters represent commonly occurring concepts, while higher edge weights denote frequently occurring discourse relations.

ence solutions to construct a graph for each exercise, called an **exercise graph**.

We first parse all stored student and reference solutions into EDUs, semantic embeddings and relations. We use DBSCAN (Ester et al. 1996) to then cluster the semantic embeddings within each exercise. For each exercise, we therefore have a set of clusters $C_1, \ldots, C_D$, where D is the number of clusters determined by DBSCAN. Note that D varies depending on the exercise. Within each cluster, we also track which EDUs correspond to reference or student solutions. Each cluster represents a distinct concept in the exercise, though in practice clustering mistakes may lead to a cluster containing more than one concept, or the same concept mapping to two clusters. Larger clusters, especially ones with reference solution EDUs, correspond to more important ideas, while smaller and outlier clusters contain uncommon errors and noise. Each cluster represents one node in the graph.

Next, we add weighted edges between the nodes. For a given reference or student solution attempt, we denote its EDUs as $\{\mathrm{EDU}_1, \ldots, \mathrm{EDU}_L\}$, and the associated clusters as $\{C_{\mathrm{EDU}_1}, \ldots, C_{\mathrm{EDU}_L}\} \subseteq \{C_1, \ldots, C_D\}$. We create directed edges $(C_{\mathrm{EDU}_i}, C_{\mathrm{EDU}_{i+1}})$ for all $i \in \{1, \ldots, L\}$, i.e. a directed edge from each EDU's associated cluster to the following EDU's cluster. We repeat this procedure for all reference and student solutions. For an edge (C_s, C_t), we set the edge weight equal to the number of adjacent EDU pairs that map to clusters C_s and C_t. Furthermore, edges are subdivided by the relation type predicted by the discourse classifier, meaning that each cluster pair may have up to four weighted edges between them, one for each relation type. See Figure 3 for an illustration of an exercise graph.

The edge weights and relation type determine the importance and discourse relation between two clusters. For example, if two large clusters are primarily joined by a cause relation, then we may expect that the student's answer should contain these two EDUs linked by a causal relation.

In a cluster, if the majority of EDUs correspond to the first EDU in the original text (i.e. EDU_1), we designate the cluster as a **start node**. Similarly, if over half the EDUs correspond to the last EDU in the original text (i.e. EDU_L), we mark the cluster as a **terminal node**. We can therefore infer whether an archetypal solution should start or end at a particular node. Note that nodes may be both start and terminal nodes, e.g. if the typical solution consists of a single EDU.

Triplet Classifier

Although the reference solutions often cover the most likely correct solutions, there are often other valid answers not present in the set of reference solutions, which therefore may be non-existent in the graph. For this reason, rather than working directly with the graphs, we train a classifier to recognize valid and invalid graph transitions. Following similar prior work (Bordes et al. 2013), we refer to this model as a triplet classifier, although in our case the model classifies EDU pairs. The classifier's aim is to generalize beyond the existing observed graph transitions and learn to infer plausible transitions as well. This approach allows the feedback mechanism to recognize not only reference answers, but other correct solutions outside of the reference solution set. There are two stages to training the model: **data generation** and the **model architecture**.

Data Generation Training the triplet classifier requires labelled data representing each graph. For each exercise, we create positive samples representing valid transitions in the graph and negative samples for invalid ones.

For a cluster C, we use E_C to describe a randomly chosen EDU from C. We create positive samples as follows:

1. A random cluster C is sampled from the graph, with a sampling probability proportional to the number of EDUs it contains.
2. We randomly sample a cluster C' from C's successors, weighted by C's outgoing edge weights. We designate the tuple $(E_C, E_{C'})$ as a positive sample.

Negative samples are constructed similarly:

1. A random cluster C is sampled from the graph, exactly as in the positive case.
2. We sample a second cluster C'', excluding the direct successors of C. C'' must also contain an incoming edge that matches a random outgoing edge from C. This matching ensures that we are not simply training the model to recognize incoherent relations, but rather plausibly true, incorrect graph transitions. In 20% of cases, we instead select a random C'', to ensure sample diversity. The tuple $(E_C, E_{C''})$ is labelled negative.

In addition, we generate data representing correct start and terminal nodes.

- If a given cluster C is a start node, we create a positive sample $(\langle \mathrm{START} \rangle, E_C)$ and a negative sample

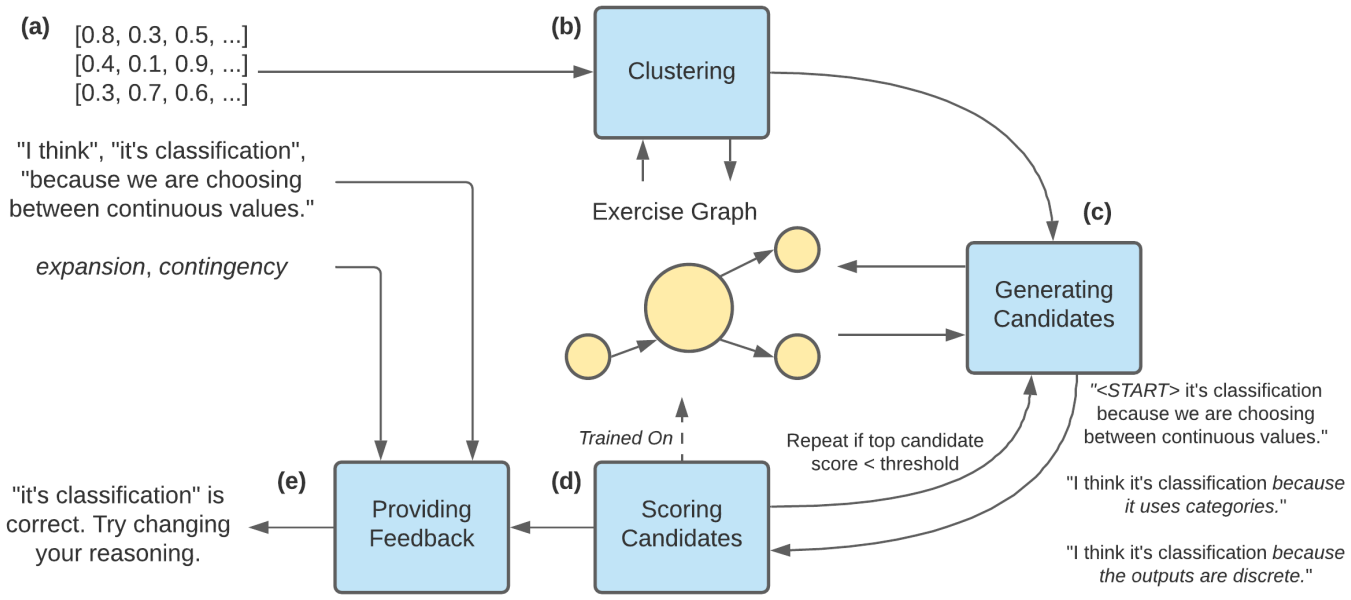

Figure 4: An overview of our feedback system after the extraction steps: (a) Features are extracted using the methods described in Section . From top to bottom, these features include the semantic embeddings from SBERT, the segmented EDUs, and discourse relations between adjacent EDUs. (b) The semantic embeddings are clustered according to the relevant exercise graph. (c) Candidate improvements are generated based on which clusters were matched to the student EDUs, and the clusters' neighbours in the graph. (d) Candidates are scored using the triplet classifier, which is trained offline using the graphs. If the top scoring candidate does not surpass a pre-defined threshold, new candidates are generated and the process is repeated. (e) A feedback message is assembled from the top-scoring candidate and previously computed EDU segments / discourse relations.

$(E_{C''}, E_C)$, where C'' is randomly sampled from the graph.

- If C is terminal, we create a positive sample $(E_C, \langle\text{STOP}\rangle)$ and a negative sample $(E_C, E_{C''})$ where again C'' is sampled randomly from the graph.

Model Architecture The model primarily consists of fine-tuning RoBERTa to minimize a cross-entropy loss. Given a sample (E_1, E_2), we encode both EDUs using RoBERTa and average the output embeddings:

$$h_{1i}, \ldots, h_{L_i,i} = \text{BERT}(E_i), i \in \{1, 2\} \quad (6)$$

$$\overline{h}_i = \sum_{j=1}^{L_i} h_{j,i} \quad (7)$$

where L_i is the length of EDU_i. The two pooled embeddings are then concatenated and the prediction is generated using a two-layer neural network.

$$\hat{y} = \text{MLP}\left(\left[\overline{h}_1, \overline{h}_2\right]\right) \quad (8)$$

This model formulation, which we call the Baseline MLP, does not distinguish between distinct exercises. In order to encode this information, we experiment using a one-hot vector to mark the exercise index. This vector $\mathbf{v}$ is concatenated to $\overline{h}_1, \overline{h}_2$ before the MLP stage:

$$\hat{y} = \text{MLP}\left(\left[\overline{h}_1, \overline{h}_2, \mathbf{v}\right]\right) \quad (9)$$

We also experimented with using a separate logistic layer for each exercise instead of the one-hot exercise encoding. However, preliminary experiments showed a clear performance degradation, and we discarded this variant in later experiments.

Local Search Inference

Using the previously described components, we create feedback messages using a local search algorithm. For a given exercise and student solution attempt, the system parses the student's solution attempt, and matches the EDUs to the exercise graph. It then attempts to modify the student solution attempt into a correct reference solution by iteratively constructing and scoring candidate solutions drawn from the exercise graph. Based on the edits needed to transform the student answer into a high-quality reference one, the system devises appropriate feedback for the student.

Clustering After the student solution is parsed into EDUs, relations, and semantic embeddings, we map student EDUs $\{E_1, \ldots, E_L\}$ to graph clusters $\{C_{E_1}, \ldots, C_{E_L}\}$ using DBSCAN (Ester et al. 1996). Note that $\{C_{E_1}, \ldots, C_{E_L}\} \subseteq \{C_1, \ldots, C_D\}$ (the full set of clusters for a graph). EDUs that cannot be clustered are marked as outliers.

Generating Candidates Next, we generate candidate solutions based on the student's solution attempt, similar to generating a beam in beam search. For each cluster $C_i \in \{C_{E_1}, \ldots, C_{E_L}\}$, we find C_i's neighbours in the graph,

and create candidates by interchanging the EDUs adjacent to E_i with C_i's neighbours. We generate candidates with $\langle \text{START} \rangle$ and $\langle \text{STOP} \rangle$ tokens if C_i is a start or terminal node in the graph. For example, using the exercise in Figure 3, if the student inputs "I think it's classification", we generate candidates "$\langle \text{START} \rangle$ classification", "I think it's classification *because it uses categories*", and "I think it's classification *because the outputs are discrete*".

Scoring Candidates The triplet classifier then scores each candidate by averaging the likelihood of its adjacent EDU pairs. If none of the candidate scores surpass a predetermined acceptance threshold α, we generate and score new candidates using the top-ranked candidate until a maximum number of steps has been reached or the top candidate surpasses α.

Providing Feedback Using the top-scoring candidate in the *first* scoring iteration, we determine how the student's EDUs were edited. We check four edit types:

- **Missing**: the student solution is missing EDUs when compared to the top candidate. E.g. if the student has forgotten to explain their reasoning, the system may say, "*Try supplying a reason for your idea.*".
- **Excess**: the student's solution contained extraneous EDUs not present in the top candidate. In this case, the system may encourage the student to shorten their answer.
- **Correct Relation**: an EDU was changed, but the discourse relations in both solutions are the same. This situation occurs when the student has supplied the expected type of answer, but some aspects are incorrect: for example, the student entered an answer and an explanation, but the explanation is incorrect. This example is illustrated by Figure 4.
- **Incorrect Relation**: similar to 'Correct Relation' but the discourse relations have changed. In practice, this scenario rarely occurs.

Finally, we record student EDUs that were matched to clusters containing reference solution EDUs. These EDUs are highly likely to be correct, so we attach the correct student EDUs to the final feedback message. See Figure 4 for an end-to-end overview of the Local Search procedure.[6]

Experiments

For all experiments using RoBERTa, we use the pre-trained HuggingFace 'roberta-base' implementation (Wolf et al. 2019).[7] All graphs are created using scikit-learn's DBSCAN implementation (Pedregosa et al. 2011), with $\varepsilon = 0.15$ and $min\_samples = 2$.[8] We base these parameter choices on qualitative observations that a lower ε value tended to yield better cluster quality. We set the acceptance threshold $\alpha = 0.95$ and restrict the local search algorithm to at most 2 iterations (i.e. edits) after observing that student solution attempts are often only missing a single EDU.

[6]We provide additional examples in the appendix at https://github.com/korbit-ai/deep-discourse-feedback.

[7]https://github.com/huggingface/transformers

[8]https://scikit-learn.org/stable/index.html

Segmentation

We train the segmenter model using the RST-DT dataset, which contains 385 Wall Street Journal (WSJ) articles divided into train (347 articles) and test (38 articles) splits (Carlson, Marcu, and Okurovsky 2001). We compare the model to Wang, Li, and Yang (2018), an LSTM-based segmenter, and Lukasik et al. (2020), a BERT-based model with extra LSTM layers. We use the same development set as Wang, Li, and Yang (2018) to fine-tune hyperparameters and we report precision, recall and F_1 score on the test set.

Discourse Classification

The discourse classifier is trained using PDTB 2.0 (Prasad et al. 2008), a collection of around 36,000 paired sentences and annotated discourse relations across 2,159 WSJ articles. We follow the same train-validation-test splits as in previous work (Rutherford and Xue 2015), and use scripts from Dai and Huang (2018) to preprocess the dataset into paragraphs.

As in previous work, we evaluate implicit and explicit relations separately, due to the increased difficulty of implicit cases. For both cases, we report the macro-F_1 score and overall accuracy. We compare our model against the following two discourse classifiers:

- Dai and Huang (2018) extract discourse unit pairs from PDTB to create paragraphs of consecutive discourse units. The main difference between our models is their use of LSTM encoders and a final CRF decoder layer.
- Liu et al. (2020) is a recent model based on RoBERTa with additional attention and pooling layers. Their model is not trained on explicit relations, and therefore we only compare implicit F_1 and accuracy scores.

Triplet Classifier

Data Generation We generate 124,000 samples, split evenly between positive and negative samples. Since the data generation procedure may produce duplicates, we cannot split the data trivially to test generalization. We instead group duplicates together and place each group into a single data split only, ensuring that the top 3 most frequent answers are placed in the training set. This ensures that duplicates are not included in multiple datasets and that generalization accuracy can be properly assessed. We create 80/10/10 splits using this procedure for training/validation/test sets, and fine-tune hyperparameters on the validation set. We report test set accuracy, comparing our models against a majority baseline.

Model We initialize the RoBERTa embeddings using the pretrained SBERT weights (Reimers and Gurevych 2019), as semantic equivalence plays a central role in the triplet classifier's task. At training time, the Korbit ITS holds 253 exercises in total, and the one-hot vector uses this length. We encode $\langle \text{START} \rangle$ and $\langle \text{STOP} \rangle$ symbols using RoBERTa's $< s >$ and $< /s >$ tokens respectively. The two-layer neural network decoder uses a 200-dimensional hidden size between first and second layers. We train the model for 2 epochs.

Student Learning Gains

After integrating the feedback models into the Korbit platform, we collect 247 distinct student interactions with the feedback mechanism in order to measure student learning gains.[9] The student's second attempt after seeing the feedback is manually labelled as *correct* or *incorrect*. Four domain experts provide the labels. The average learning gain is then measured as the percentage of times the second student attempt is labelled as correct.

We compare the feedback model against two baselines. The first is a *Minimal Feedback* baseline, which simply tells students that their solution is incorrect and they should try again. We also compare the feedback model against a *Cluster-Based Feedback* model. After parsing the student's attempted solution into EDUs and semantic embeddings, the Cluster-Based Feedback uses DBSCAN to match the student EDUs to clusters in the graph. Then, if any student's EDU successfully matches to a cluster containing at least one EDU from a reference solution, the system indicates to the student that their EDU is correct. It then advises the student to re-word other parts of their answer or add additional details without specifying further details.

This baseline tests the effectiveness of the semantic clusters without the relational edges, triplet classifier, and local search. While it uses segmentation and semantic information in the student's answer, it cannot take advantage of the full graph structure to prompt students about discourse relations such as justifications or comparisons. For example, for the interaction listed in Table , the Cluster-Based Feedback would respond "*'it's a classification task' is correct. Try rewording the other parts of your answer or adding additional details*".

Since other feedback mechanisms exist in Korbit (Kochmar et al. 2020), we record the number of previous student attempts on a given exercise before interacting with our discourse-based feedback system. We report both the average learning gain before the second attempt on an exercise, as well as the average learning gain across all attempts on an exercise.

We do not compare our system against other feedback mechanisms in Korbit (Kochmar et al. 2020), since their formulations differ greatly from ours and are largely orthogonal to our approach. For example, Kochmar et al. (2020) uses Wikipedia as a resource for hints, while ours does not.

Direct Feedback Comparison In order to intrinsically evaluate model's performance, we also compare the *Minimal Feedback*, *Cluster-Based Feedback* and *Full Feedback* outputs directly in a ranking-based comparison. Each model's feedback message is ranked w.r.t. its helpfulness on 91 incorrect student solution attempts. Four domain experts rank the generated feedback. For each model, we report the percentage of cases the model's feedback is the top-ranked one, allowing for ties.

[9]We discard 83 interactions in which the student's first solution attempt was labelled correct by a human annotator.

Model	P (%)	R (%)	F_1 (%)
Wang, Li, and Yang (2018)	92.9	95.7	94.3
Lukasik et al. (2020)	95.7	96.8	94.6
Our Model	**97.6**	**97.4**	**97.5**

Table 2: Discourse segmentation results on the RST-DT test set. We measure precision (P), recall (R), and F_1 score.

	Implicit		Explicit	
Model	F_1	Acc	F_1	Acc
D and H (2018)	48.82	57.44	93.21	93.98
Liu et al. (2020)	63.39	69.06	-	-
Our Model	**64.43**	**70.46**	**94.20**	**95.15**

Table 3: Discourse classification results on the PDTB 2.0 test set. We compare the F_1 and accuracy (Acc) scores (%) against Dai and Huang (2018) (D and H (2018)) and Liu et al. (2020). Liu et al. (2020)'s model is not trained to classify explicit discourse relations and we cannot compare performance on these cases.

Results

Segmentation

The segmentation results for RST-DT are presented in Table 2. Our model outperforms Lukasik et al. (2020)'s model, despite using a much simpler model formulation.

Discourse Classification

Discourse classification results on the PDTB 2.0 test set are presented in Table 3. Although Dai and Huang (2018)'s and our model both rely on chaining adjacent discourse units into paragraphs, using the RoBERTa encoder results in strong implicit relation accuracy gains. On the explicit cases, accuracy is closer to Dai and Huang (2018) but still remains high. Interestingly, our model achieves similar results to Liu et al. (2020), although our model design is considerably simpler and extends to explicit relations as well.

Triplet Classifier

The triplet classification results on the generated test set are shown in Table 4. Although there is no previous work for comparison, both models significantly outperform a majority baseline. Using a general z-test, we test if the one-hot exercise index vector significantly improves the Baseline MLP. Since the generated test set is so large, we find that the improvements are significant with $p < 0.0001$.

Student Learning Gains

Results from the learning gains experiments are shown in Table 5. The 'All Attempts' column indicates student learning gains across all attempts made, and the 'Before Second Attempt' column contains entries in which the student tried only once previously. Our experiments show the Full Feedback Model strongly improves over the baselines in both settings. We test the results for significance using an $N-1$ χ^2 test. The Full Feedback Model leads to significant student learning gains compared to the Cluster-Based Feedback in

Model	Accuracy (%)
Majority Baseline	50.39
Baseline MLP	79.18 ± 0.71*
+ One-Hot Exercise	$\mathbf{81.20 \pm 0.68}$**

Table 4: Test set results from generated data representing valid and invalid graph transitions. One-Hot Exercise refers to the one-hot vector indicating exercise index. * indicates significance at the 95% confidence level over the Majority Baseline, and ** marks significance at the 95% confidence level over the Baseline MLP.

the All Attempts setting with $p = 0.0047$ and in the Before Second Attempt setting with $p = 0.00006$. This result indicates that the graph structure and triplet classifier are crucial to our feedback approach, and the clusters by themselves are not adequate to achieve high-quality feedback. The inadequacy of the Cluster-Based Feedback is especially apparent in the All Attempts setting, where it does not noticeably improve over the Minimal Feedback Baseline. We measure inter-annotator agreement using Krippendorff's alpha using 30 samples. We obtain $\alpha = 0.831$, which indicates significant inter-annotator agreement.

It is interesting to note that the learning gains are considerably more pronounced when the student has only been given a single hint. Although the result may seem surprising, the difference is most likely due to selection bias of the students. Students who require many hints in order to solve an exercise may face external challenges or have knowledge gaps. For example, these students may experience language barriers or lack the prerequisites to solve the problem.

In general, we find that the most common feedback type is the *missing* class, as students often answer only half the question and omit aspects such as explaining their answer. In contrast, the *wrong relation* feedback is rarely used, as generally, students are aware of which relation is expected even if the EDU supplied is incorrect.

Direct Feedback Comparison Results from the direct feedback comparison are shown in Table 6. The Full Feedback Model's feedback is ranked best in 65.93% of cases, substantially higher than other models. The Full Feedback Model results compared to the Cluster-Based Feedback are significant with $p < 0.0001$, using an $N - 1$ χ^2-test. This result parallels the Student Learning Gains experiments, demonstrating the Full Feedback Model provides more meaningful feedback compared to baselines.

We measure inter-annotator agreement by gathering each pair of annotator's labels and computing the Spearman's rank correlation between all pairs. The result is $\rho = 0.868$, which indicates a very high agreement between annotators.

Discussion

The results of both the learning gains and feedback ranking evaluation demonstrate that the Full Feedback Model is an effective feedback tool for ITSs. The comparison against the Cluster-Based Feedback model further demonstrates the importance of incorporating the rhetorical structure and the inferred relationships into the feedback generation process.

Nevertheless, we did notice cases where the model's feedback was subpar or even nonsensical. One of the key weaknesses lies with the triplet classifier. Since the classifier's role is critical for selecting the right feedback, its errors are often amplified in the final feedback message. For example, suppose that the correct solution is the EDU sequence $[E_1, E_2]$ and the student's attempt is $[E_1, E_3]$. If the highest scoring candidate is instead $[E_1, \langle \text{STOP} \rangle]$, then the feedback model will mistakenly tell the student to shorten their answer, instead of recommending E_2.

Future methods may benefit from softening the classifier's role within the feedback mechanism, or by trying to increase accuracy. One option is to encode the exercise's problem text when learning graph transitions. The current formulation encodes the exercise as a one-hot vector, ignoring the problem statement text. Richer exercise representations that include the question may lead to better classifier accuracy. Another potential improvement is to change the triplet classification task into a ranking task, where the model must learn to rank reference solutions over student solution attempts.

Our approach leverages four distinct RoBERTa models, which consume a considerable memory footprint. Each subtask is also optimized separately rather than jointly optimized towards the final feedback generation task. Having numerous subtasks results in difficulties with maintaining many separate moving parts, as well as computational issues. We explored combining related tasks such as segmentation and discourse classification in a multi-task setup, but found that performance on both tasks declined substantially. Future work may analyze methods to simplify the feedback procedure without sacrificing accuracy.

Although our experiments only consider STEM exercises, the feedback mechanism is not exclusively tuned towards STEM and could easily be adapted to other subjects. We expect that topics requiring less domain-specific knowledge will be easier to adapt, as the semantic equivalence model (SBERT) is not intended for domain-transfer tasks. In our experiments, SBERT occasionally struggled with domain-specific concepts, e.g. equating the ideas of *bias term* and *intercept* in the context of linear models.

There are many other directions worth exploring. One possibility is to replace the fixed feedback rules with a seq2seq model, which would gather edits from each step of the local search and output feedback based on the edit path. Future work may also consider generating multiple feedback messages and training a classifier to choose among them.

Related Work

Discourse segmentation is traditionally viewed as a preliminary step in RST parsing (Mann and Thompson 1988), though other applications, such as summarization, exist as well (Li, Wu, and Li 2020). Past work on discourse segmentation includes using SVMs (Hernault et al. 2010), LSTMs with CRFs (Wang, Li, and Yang 2018), and more recently, BERT-based architectures (Lukasik et al. 2020). Discourse segmentation has also been explored in other contexts such as medical (Ferracane et al. 2019) and multilingual settings (Muller, Braud, and Morey 2019).

Model	Average Learning Gain (%)	
	All Attempts	Before Second Attempt
Minimal Feedback Baseline	25.64 ± 13.70	8.33 ± 15.64
Cluster-Based Feedback	23.53 ± 13.25	20.00 ± 10.08
Full Feedback Model	$\mathbf{51.11 \pm 14.61}$*	$\mathbf{75.00 \pm 18.98}$*

Table 5: Student learning gains on the Korbit ITS. Significant results at 95% confidence over the Cluster-Based feedback are marked with *.

Model	Top Ranked (%)
Minimal Feedback Baseline	24.18 ± 8.80
Cluster-Based Feedback	32.97 ± 9.66
Full Feedback Model	$\mathbf{65.93 \pm 9.73}$*

Table 6: Model feedback quality in a ranked comparison test. We measure the proportion each model provides the top ranked feedback (%), with corresponding 95% confidence intervals. Since we allow for ties, the values do not sum to 100. Significant results over the Cluster-Based feedback are marked with *.

The PDTB corpus (Prasad et al. 2008) has motivated many studies on discourse classification, especially on the challenging implicit discourse case (Pitler, Louis, and Nenkova 2009; Lin, Kan, and Ng 2009; Rutherford and Xue 2015). Past approaches have used feature-based statistical models (Pitler, Louis, and Nenkova 2009), LSTMs with attention (Liu et al. 2016), and BERT-based architectures (Liu et al. 2020). Our work is most similar to Dai and Huang (2018), since we follow their approach in organizing discourse units into paragraphs. Our model shares characteristics with Liu et al. (2020), whom also employ a BERT-based approach to discourse classification.

An alternative to our discourse approach is full RST parsing. Many RST parsers exist (Braud, Coavoux, and Søgaard 2017; Ji and Eisenstein 2014; Feng and Hirst 2014), but highly accurate RST parsing remains out of reach due to the complex nature of RST trees. The majority of student answers on Korbit are less than 3 EDUs and therefore building a complex tree structure may be unnecessary.

Previous work has studied representing graph transitions with neural networks (Bordes et al. 2013; Fan et al. 2014). Local coherence modelling, in which models learn to rank sentences in a logical order (Barzilay and Lapata 2008), is another similar task.

Many ITSs provide feedback guiding students towards the correct solution in an exercise (Ramachandran et al. 2018; Rus, Niraula, and Banjade 2015; Nye, Graesser, and Hu 2014; Ventura et al. 2018; Al-Nakhal and Abu-Naser 2017; Al-Dahdooh and Abu-Naser 2017; Tamura et al. 2015; Guo et al. 2016; Shah, Shah, and Kurup 2017; Serban et al. 2020; Kochmar et al. 2020). However, to our knowledge, none have previously employed a neural discourse-based mechanism for personalized feedback.

Conclusion

In this work, we investigate how we can use deep learning, relational graphs, and discourse analysis to develop a data-driven, automated, personalized feedback mechanism. Our approach decomposes students' attempted solutions into EDUs and relations, then matches their attempt to a graph representing the exercise. Based on a traversal of this graph, personalized feedback for each student is generated on the fly. Although it is not our primary goal, our discourse segmentation and classification models also attain new state-of-the-art performance on RST-DT and PDTB 2.0, respectively.

Evaluating the feedback quality and student learning gains on Korbit show that our feedback model markedly improves the students' learning experience. The results suggest that our feedback system can effectively help students and that discourse analysis may be a highly effective approach for building personalized feedback systems.

Acknowledgments

This work was conducted while the first author was employed at Korbit for an internship.

References

Al-Dahdooh, R.; and Abu-Naser, S. 2017. Development and Evaluation of the Oracle Intelligent Tutoring System (OITS). *European Academic Research* 4: 8711–8721.

Al-Nakhal, M.; and Abu-Naser, S. 2017. Adaptive Intelligent Tutoring System for Learning Computer Theory. *European Academic Research* 4: 8770–8782.

Barzilay, R.; and Lapata, M. 2008. Modeling Local Coherence: An Entity-Based Approach. *Computational Linguistics* 34(1): 1–34. doi:10.1162/coli.2008.34.1.1. URL https://www.aclweb.org/anthology/J08-1001.

Bordes, A.; Usunier, N.; Garcia-Durán, A.; Weston, J.; and Yakhnenko, O. 2013. Translating Embeddings for Modeling Multi-Relational Data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, 2787–2795. Red Hook, NY, USA: Curran Associates Inc.

Bowman, S. R.; Angeli, G.; Potts, C.; and Manning, C. D. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 632–642. Lisbon, Portugal: Association for Computational Linguistics. doi:10.18653/v1/D15-1075. URL https://www.aclweb.org/anthology/D15-1075.

Braud, C.; Coavoux, M.; and Søgaard, A. 2017. Cross-lingual RST Discourse Parsing. In *Proceedings of the 15th Conference*

of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, 292–304. Valencia, Spain: Association for Computational Linguistics. URL https://www.aclweb.org/anthology/E17-1028.

Carlson, L.; Marcu, D.; and Okurovsky, M. E. 2001. Building a Discourse-Tagged Corpus in the Framework of Rhetorical Structure Theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue*. URL https://www.aclweb.org/anthology/W01-1605.

Cer, D.; Diab, M.; Agirre, E.; Lopez-Gazpio, I.; and Specia, L. 2017. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation. *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* doi:10.18653/v1/s17-2001. URL http://dx.doi.org/10.18653/v1/S17-2001.

Dai, Z.; and Huang, R. 2018. Improving Implicit Discourse Relation Classification by Modeling Inter-dependencies of Discourse Units in a Paragraph. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 141–151. New Orleans, Louisiana: Association for Computational Linguistics. doi:10.18653/v1/N18-1013. URL https://www.aclweb.org/anthology/N18-1013.

Dasigi, P.; Burns, G. A. P. C.; Hovy, E.; and de Waard, A. 2017. Experiment Segmentation in Scientific Discourse as Clause-level Structured Prediction using Recurrent Neural Networks. *ArXiv* abs/1907.11692.

Ester, M.; Kriegel, H.-P.; Sander, J.; Xu, X.; et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, 226–231.

Fan, M.; Zhou, Q.; Chang, E.; and Zheng, T. F. 2014. Transition-based Knowledge Graph Embedding with Relational Mapping Properties. In *Proceedings of the 28th Pacific Asia Conference on Language, Information and Computing*, 328–337. Phuket,Thailand: Department of Linguistics, Chulalongkorn University. URL https://www.aclweb.org/anthology/Y14-1039.

Feng, V. W.; and Hirst, G. 2014. A Linear-Time Bottom-Up Discourse Parser with Constraints and Post-Editing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 511–521. Baltimore, Maryland: Association for Computational Linguistics. doi:10.3115/v1/P14-1048. URL https://www.aclweb.org/anthology/P14-1048.

Ferracane, E.; Page, T.; Li, J. J.; and Erk, K. 2019. From News to Medical: Cross-domain Discourse Segmentation. In *Proceedings of the Workshop on Discourse Relation Parsing and Treebanking 2019*, 22–29. Minneapolis, MN: Association for Computational Linguistics. doi:10.18653/v1/W19-2704. URL https://www.aclweb.org/anthology/W19-2704.

Guo, Q.; Kulkarni, C.; Kittur, A.; Bigham, J. P.; and Brunskill, E. 2016. Questimator: Generating knowledge assessments for arbitrary topics. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI'16). AAAI Press*.

Hernault, H.; Prendinger, H.; Ishizuka, M.; et al. 2010. HILDA: A discourse parser using support vector machine classification. *Dialogue & Discourse* 1(3).

Ji, Y.; and Eisenstein, J. 2014. Representation Learning for Text-level Discourse Parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 13–24. Baltimore, Maryland: Association for Computational Linguistics. doi:10.3115/v1/P14-1002. URL https://www.aclweb.org/anthology/P14-1002.

Kochmar, E.; Vu, D. D.; Belfer, R.; Gupta, V.; Serban, I. V.; and Pineau, J. 2020. Automated Personalized Feedback Improves Learning Gains in An Intelligent Tutoring System. In Bittencourt, I. I.; Cukurova, M.; Muldner, K.; Luckin, R.; and Millán, E., eds., *Artificial Intelligence in Education*, 140–146. Cham: Springer International Publishing. ISBN 978-3-030-52240-7.

Kulik, J. A.; and Fletcher, J. 2016. Effectiveness of intelligent tutoring systems: a meta-analytic review. *Review of educational research* 86(1): 42–78.

Li, Z.; Wu, W.; and Li, S. 2020. Composing Elementary Discourse Units in Abstractive Summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 6191–6196. Online: Association for Computational Linguistics. doi:10.18653/v1/2020.acl-main.551. URL https://www.aclweb.org/anthology/2020.acl-main.551.

Lin, Z.; Kan, M.-Y.; and Ng, H. T. 2009. Recognizing Implicit Discourse Relations in the Penn Discourse Treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, 343–351. USA: Association for Computational Linguistics. ISBN 9781932432596.

Liu, X.; Ou, J.; Song, Y.; and Jiang, X. 2020. On the Importance of Word and Sentence Representation Learning in Implicit Discourse Relation Classification. In Bessiere, C., ed., *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, 3830–3836. International Joint Conferences on Artificial Intelligence Organization. doi:10.24963/ijcai.2020/530. URL https://doi.org/10.24963/ijcai.2020/530. Main track.

Liu, Y.; Li, S.; Zhang, X.; and Sui, Z. 2016. Implicit Discourse Relation Classification via Multi-Task Neural Networks. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/11831.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *ArXiv* abs/1907.11692.

Lukasik, M.; Dadachev, B.; Simoes, G.; and Papineni, K. 2020. Text Segmentation by Cross Segment Attention. *ArXiv* abs/2004.14535.

Mabona, A.; Rimell, L.; Clark, S.; and Vlachos, A. 2019. Neural Generative Rhetorical Structure Parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2284–2295. Hong Kong, China: Association for Computational Linguistics. doi:10.18653/v1/D19-1233. URL https://www.aclweb.org/anthology/D19-1233.

Mann, W.; and Thompson, S. 1988. Rhetorical Structure Theory: Toward a functional theory of text organization. *Text & Talk* 8: 243 – 281.

Muller, P.; Braud, C.; and Morey, M. 2019. ToNy: Contextual embeddings for accurate multilingual discourse segmentation of full documents. In *Proceedings of the Workshop on Discourse Relation Parsing and Treebanking 2019*, 115–124. Minneapolis, MN: Association for Computational Linguistics. doi:10.18653/v1/W19-2715. URL https://www.aclweb.org/anthology/W19-2715.

Nye, B.; Graesser, A.; and Hu, X. 2014. AutoTutor and Family: A Review of 17 Years of Natural Language Tutoring. *International Journal of Artificial Intelligence in Education* 24. doi:10.1007/s40593-014-0029-5.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12: 2825–2830.

Pitler, E.; Louis, A.; and Nenkova, A. 2009. Automatic Sense Prediction for Implicit Discourse Relations in Text. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, 683–691. USA: Association for Computational Linguistics. ISBN 9781932432466.

Prasad, R.; Dinesh, N.; Lee, A.; Miltsakaki, E.; Robaldo, L.; Joshi, A.; and Webber, B. 2008. The Penn Discourse TreeBank 2.0. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*. Marrakech, Morocco: European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2008/pdf/754_paper.pdf.

Ramachandran, S.; Jensen, R.; Ludwig, J.; Domeshek, E.; and Haines, T. 2018. ITADS: A Real-World Intelligent Tutor to Train Troubleshooting Skills. In Penstein Rosé, C.; Martínez-Maldonado, R.; Hoppe, H. U.; Luckin, R.; Mavrikis, M.; Porayska-Pomsta, K.; McLaren, B.; and du Boulay, B., eds., *Artificial Intelligence in Education*, 463–468. Cham: Springer International Publishing. ISBN 978-3-319-93846-2.

Reimers, N.; and Gurevych, I. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 3982–3992. Hong Kong, China: Association for Computational Linguistics. doi:10.18653/v1/D19-1410. URL https://www.aclweb.org/anthology/D19-1410.

Rus, V.; Niraula, N. B.; and Banjade, R. 2015. DeepTutor: An Effective, Online Intelligent Tutoring System That Promotes Deep Learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, 4294–4295. AAAI Press. ISBN 0262511290.

Rutherford, A.; and Xue, N. 2015. Improving the Inference of Implicit Discourse Relations via Classifying Explicit Discourse Connectives. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 799–808. Denver, Colorado: Association for Computational Linguistics. doi:10.3115/v1/N15-1081. URL https://www.aclweb.org/anthology/N15-1081.

Serban, I. V.; Gupta, V.; Kochmar, E.; Vu, D. D.; Belfer, R.; Pineau, J.; Courville, A.; Charlin, L.; and Bengio, Y. 2020. A Large-Scale, Open-Domain, Mixed-Interface Dialogue-Based ITS for STEM. *Artificial Intelligence in Education* 387–392. ISSN 1611-3349. doi:10.1007/978-3-030-52240-7_70. URL http://dx.doi.org/10.1007/978-3-030-52240-7_70.

Shah, R.; Shah, D.; and Kurup, L. 2017. Automatic question generation for intelligent tutoring systems. In *2017 2nd International Conference on Communication Systems, Computing and IT Applications (CSCITA)*, 127–132. IEEE.

Tamura, Y.; Takase, Y.; Hayashi, Y.; and Nakano, Y. I. 2015. Generating quizzes for history learning based on Wikipedia articles. In *International Conference on Learning and Collaboration Technologies*, 337–346. Springer.

Ventura, M.; Chang, M.; Foltz, P.; Mukhi, N.; Yarbro, J.; Salverda, A. P.; Behrens, J.; Ahn, J.-w.; Ma, T.; Dhamecha, T. I.; Marvaniya, S.; Watson, P.; D'helon, C.; Tejwani, R.; and Afzal, S. 2018. Preliminary Evaluations of a Dialogue-Based Digital Tutor. In Penstein Rosé, C.; Martínez-Maldonado, R.; Hoppe, H. U.; Luckin, R.; Mavrikis, M.; Porayska-Pomsta, K.; McLaren, B.; and du Boulay, B., eds., *Artificial Intelligence in Education*, 480–483. Cham: Springer International Publishing. ISBN 978-3-319-93846-2.

Wang, Y.; Li, S.; and Yang, J. 2018. Toward Fast and Accurate Neural Discourse Segmentation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 962–967. Brussels, Belgium: Association for Computational Linguistics. doi:10.18653/v1/D18-1116. URL https://www.aclweb.org/anthology/D18-1116.

Williams, A.; Nangia, N.; and Bowman, S. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 1112–1122. New Orleans, Louisiana: Association for Computational Linguistics. doi:10.18653/v1/N18-1101. URL https://www.aclweb.org/anthology/N18-1101.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; Davison, J.; Shleifer, S.; von Platen, P.; Ma, C.; Jernite, Y.; Plu, J.; Xu, C.; Scao, T. L.; Gugger, S.; Drame, M.; Lhoest, Q.; and Rush, A. M. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv* abs/1910.03771.