COLUMBIA ENGINEERING
The Fu Foundation School of Engineering and Applied Science

COLUMBIA UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

Neural Approaches for AMR Alignment and AMR-based Automatic Summary Evaluation

Author:
AadilMehdi Sanchawala

Supervisor:
Kathleen McKeown

December 2022

Abstract

Summarization of long dialogue transcripts poses two distinct challenges. First, the full self-attention mechanism in the transformer architecture used by some state-of-the-art large language models is in practice infeasible for long documents. Second, dialogue is not suited for extractive summarization and can require a deeper understanding of references and nuance. In order to improve summarization of long dialogue transcripts, we propose breaking the task of summarization into two discrete steps; content selection and summarization. We show that content selection as a prior task for abstractive summarization is beneficial, and we experiment with different variations of content selection methods in order to study which design decisions are most effective in improving downstream summarization.

Acknowledgments

I would to extend my appreciation to Dr. Kathleen McKeown for her invaluable guidance and support throughout the completion of my thesis. This research would not have been possible without the mentorship she provided and her advice and encouragement.

I would also like to thank Faisal and Fei-Tzin for their help and guidance. They advised me on ideas, research directions, and taught me how to be a better researcher. I would also like to thank Vishwesh for his constant support and all the time he helped me with my code and debugging.

Contents

1 Introduction

Determining salient parts of a document plays a vital role in the highly abstractive summarization of long creative texts like novel chapters. It is crucial to know which parts of the document are highly relevant and contain the gist of the chapter to be included in the summary. Representing documents as Abstract Meaning Representations (AMRs), we can describe a document as a graph where the nodes represent various entities and concepts present in the document, and the edges define the relations and interactions among them.

The primary step in picking out salient parts of a document AMR is to compare it with multiple reference summary AMRs and locate which document parts correspond to the reference summaries. This ability enables us to identify the centrality and importance of various concepts, entities, and relations. Thus there is a need for a tool that helps pick out important document sections using reference summaries. With the salient sections of the document identified, we can prune the document AMR and use the trimmed graph to generate semantically and conceptually sound summaries. This tool would also aid in automatically evaluating system-generated summaries by informing which parts of documents are considered important by the summarizer, enabling the use of multiple reference summaries on a fine-grained level for automatic evaluations.

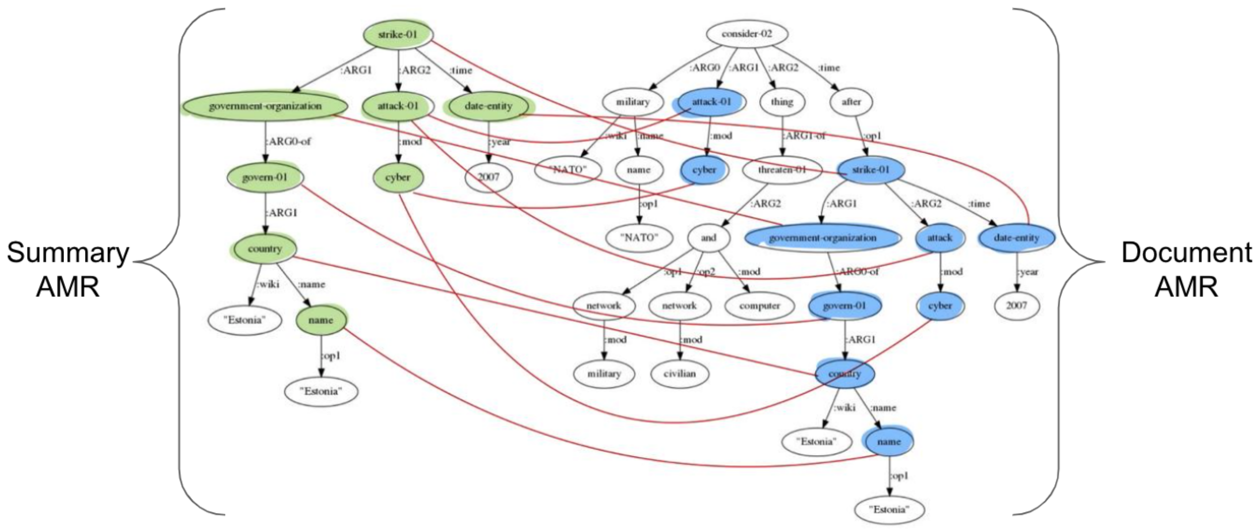
To achieve these downstream tasks of summary generation and automatic evaluation, we require a highly precise and robust method to perform AMR node alignments across document and summary AMR graphs. Since we rely on reference summaries to identify salient neighborhoods of a document AMR graph, finding alignments between the document AMR and summary AMR is paramount. Accurate alignments are also leveraged to prune the document graph to generate better summaries and evaluate system-generated summaries.

In this report, we present our research on using neural-based approaches for AMR graph alignment. During this semester, we focused on the following:

- Examining various options for embeddings and conducting a detailed analysis of the trade-offs and behavior for the AMR alignment task.
- Improving the training of graph neural networks developed in the previous semester and learning rich, context-aware, and structure-aware graph embeddings.
- Developing an accurate and robust AMR alignment system with high precision and recall.
- Creating an AMR-based summary scoring mechanism and using the alignment system to evaluate system-generated summaries against multiple reference summaries.

We will begin by providing a brief overview of related works such as graph attention networks, entity alignment, and contrastive alignment for representation learning.

We will then describe the task of AMR graph alignment and the datasets used for our experimentation. We will also discuss the baseline we established for developing our models. Next, we will delve into the details of our embedding analysis, models, pre-training objectives, and performance using global word embeddings and contextual embeddings.



Document: *NATO considers cyber attacks a threat to military and civilian computer networks after the Estonian Government was struck by cyber attacks in 2007.*
Summary: *In 2007 the Estonian Government was struck by cyber attacks*

Figure 1: AMR Alignment for a Document Summary AMR Graph.

2 Related Work

2.1 Graph Neural Networks

Graph Neural Networks (GNNs) are a type of neural network that are specifically designed to analyze graph structures. They are particularly useful for modeling the dependencies between nodes in a graph, and have been successful in a variety of graph analysis tasks. GNNs can take several different forms, including Graph Convolutional Networks (GCNs) and Graph Attention Networks (GATs), which differ in the way they process and aggregate information from neighboring nodes. GATs, in particular, have proven to be effective thanks to their attention mechanism, which allows them to assign different weights to the contribution of each neighbor. Among the various versions of GATs that have been developed, such as GAT, GATv2, and SuperGAT, SuperGAT has consistently demonstrated the best performance, and is therefore the model of choice for our research.

2.2 Abstract Meaning Representations

Abstract Meaning Representation (AMR) is a method for representing the semantics of a sentence as a directed acyclic graph. The graph consists of concept nodes connected by named edges, and represents the relationships between the concepts in the sentence. They are rooted, labeled, directed acyclic graphs comprising whole sentences. AMR aims to preserve semantic relations. They are intended to abstract away from syntactic representations. AMR is designed to be independent of the specific wording of a sentence, so that sentences with the same meaning will be represented by the same AMR graph. AMR parsing is the process of converting a natural language sentence into an AMR graph, and there are several different methods for doing this, including two-stage parsers, dependency-based parsers, transition-based parsers, and neural parsers. AMR alignment is the process of mapping a span of words in a sentence to a corresponding concept in the AMR graph. This is important for a variety of tasks, such as training AMR parsers, cross-lingual AMR parsing, applying AMR to other tasks, and developing new methods for semantic parsing. In our work, we use an AMR parser and aligner to convert text into AMR graphs and compute contextualized embeddings for the nodes in the graph.

2.3 Entity Alignment

Entity Alignment, also known as Entity Matching or Entity Resolution (Fu et al., 2019; Nie et al., 2019), is one of the most basic and critical technologies in knowledge fusion. Entity alignment aims to identify entities from different knowledge graphs that describe the same real-world object. The knowledge graphs here can be in different languages and data sources, but there is overlapping and complementary knowledge. Recent works exploit graph structures and the use of attention to perform entity alignment between knowledge graphs. In addition, some works introduce distant neighbors into the attention mechanism to expand the overlap between neighborhood structures which helps mitigate the non-isomorphism of graph neighborhoods. Using attention mechanisms helps highlight helpful distant neighbors and reduce noises from nearby neighbors. This is a relevant issue since document and summary graphs often display non-isomorphism between their graphs, and identifying helpful distant neighbors can prove to be extremely helpful in the alignment task.

2.4 Automatic Summary Evaluation

Evaluation metrics play a crucial role in the development of summarization systems, as they are used to assess the quality of the generated summaries and compare them to human judgments. There are several types of evaluation metrics that can be used for natural language text generation, including n-gram matching metrics (such as BLEU and METEOR), edit distance metrics (such as Levenshtein distance), embedding matching metrics (such as BERTScore and MoverScore), and learned function metrics (such as S3 and BLANC). N-gram matching metrics measure the overlap between the generated text and a reference text using n-grams (sequences of n words), while edit distance metrics measure the number of changes needed to transform one text into another. Embedding matching metrics use word embeddings to compare the generated text to the reference text, and learned function metrics are trained on human judgment datasets to predict the evaluation score. These evaluation methods have some limitations, including their dependence on external resources, sensitivity to word order, and limited ability to capture complex linguistic phenomena.

3 Dataset

For the purposes of these exploration experiments and model development, we will use the following datasets,

3.1 The News AMR Dataset

This dataset consists of document-summary sentence pairs and corresponding AMR graphs for news articles, all of which have been manually annotated for the AMR alignment task. It includes a total of 646 data points, which are divided into 450 training samples, 50 validation and development samples, and 164 testing samples.

Most of our model development and exploration is conducted using this dataset. However, it is important to note that this dataset is not representative of a typical novel summarization dataset, as it is focused on news articles. Ideally, we would have preferred to work with a dataset from a novel domain, but due to the lack of human-annotated data in that domain, we were only able to use the news dataset for development.

4 Recapping Prior Work

4.1 Graph Attention Model

Last, we used a graph neural network architecture that consisted of three Super-GAT graph attention convolution layers and Leaky ReLU as the non-linear activation function. The purpose of this network was to transform the initial node embeddings of the document and summary AMR graphs into a common embedding space, such that nodes with similar neighbors and relationships had similar embeddings. The transformed embeddings from the document and summary AMRs were then used to compute the triplet margin loss and the attention loss, which served as the supervising criterion for our experiments. This process allowed us to ensure that the document and summary AMR graphs were aligned in a consistent and accurate manner. Figure 2 illustrates the architecture of this model.

4.2 Pre-training Approaches

The graph reconstruction and masked node reconstruction tasks were explored as pre-training objectives to improve the alignment between nodes in document-summary graph pairs. These tasks were performed using a graph attention model serving as the encoder. In the graph reconstruction task, the reconstruction loss between the input and reconstructed graph was used as the criterion, while in the masked node reconstruction task, the euclidean distance between the original and predicted node embeddings was used as the criterion. After training the encoder on these tasks, it was fine-tuned on the alignment task. However, these pre-training tasks did not result in any improvement in the accuracy and performance of the alignment model.

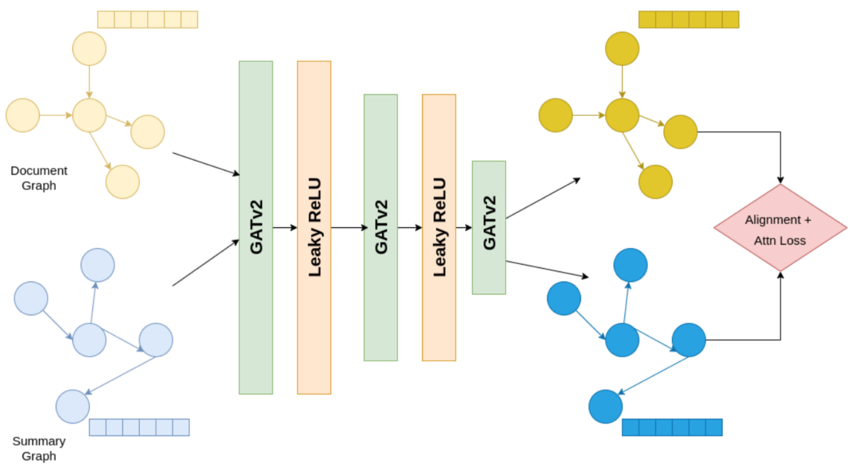


Figure 2: GNN Alignment Model developed in Spring'22. This model uses Super-GAT graph convolution layers. The GAT embeddings are initialized using pretrained word2vec embeddings.

Table 1: Results from prior work.

Model	Precision	Recall	F1-Score
Exact Match	0.31	0.86	0.45
Word2Vec Similarity	0.27	0.87	0.41
GNN-Encoder-V1	0.56	0.73	0.62

4.3 Results

In the previous research, we found that the GNN encoder gave us the best performance, resulting in an F1-score of 0.62. However, while the recall of the system was good, the precision was not as strong, resulting in a high number of false positive alignments that would negatively impact the downstream evaluation task. We will refer to the best-performing model from previous works as GNN-Encoder-V1 henceforth.

5 AMR Node Representation

To align AMR graphs accurately, it is necessary to represent the concept nodes in the AMR with rich embeddings that capture the node's meaning, structure, and concept. There are several options for creating these embedding representations, and it is important to choose ones that are contextual, consistent, and structure-aware.

Contextual embeddings are those that capture the context in which a concept appears, which is especially important for the task of abstractive summarization, as summaries may be paraphrased using different words and sentence structures compared to the input document. Consistent embeddings, on the other hand, ensure that similar concepts are represented in a similar way, regardless of the words used to express them. This is important because the goal of AMR is to abstract away from syntactic representations and preserve semantic relations, meaning that sentences with similar meanings should be assigned the same AMR regardless of how they are worded.

Structure-aware embeddings, capture the structure of the AMR graph, taking into account the relationships between different concepts and their relative importance. Following the distributional hypothesis, it is expected that a concept is heavily influenced by its neighboring nodes and relations, so it is important for the node embeddings to attend to essential neighbors and relations. By representing the concept nodes in the AMR with rich, contextual, consistent, and structure-aware embeddings, it is possible to align AMR graphs accurately and improve the accuracy of the summary.

5.1 Choices for AMR Node Embeddings

To represent the nodes in our AMR graphs, we have several options. These options can be broadly classified into three categories: static word2vec embeddings, transformer embeddings, and graph attention network (GAT) embeddings.

Static embeddings are obtained using a pre-trained word2vec model that maps each concept label to a 300-dimensional vector. These embeddings are global and do not change based on the context in which the concept label is used.

Transformer embeddings are context-dependent embeddings obtained by using pre-trained large language models based on the transformer architecture. The transformer embeddings for a concept are obtained by taking the mean of the transformer embeddings of the sentence word span corresponding to the node. We use TAMR to obtain the node-to-sentence word span alignments.

GAT embeddings are context-dependent and structure-aware embeddings of the AMR graph. We use graph attention networks to pre-train a GNN encoder with self-attention on the masked concept-node label prediction task. The encoder is then fine-tuned on the node alignment task. The embeddings produced by the GNN encoder are contextual and structure-aware because the encoder learns to attend to the essential neighbor nodes and uses their interaction information to represent node embeddings. There are several considerations to keep in mind when training a GAT, and we will explore these in more detail in later sections.

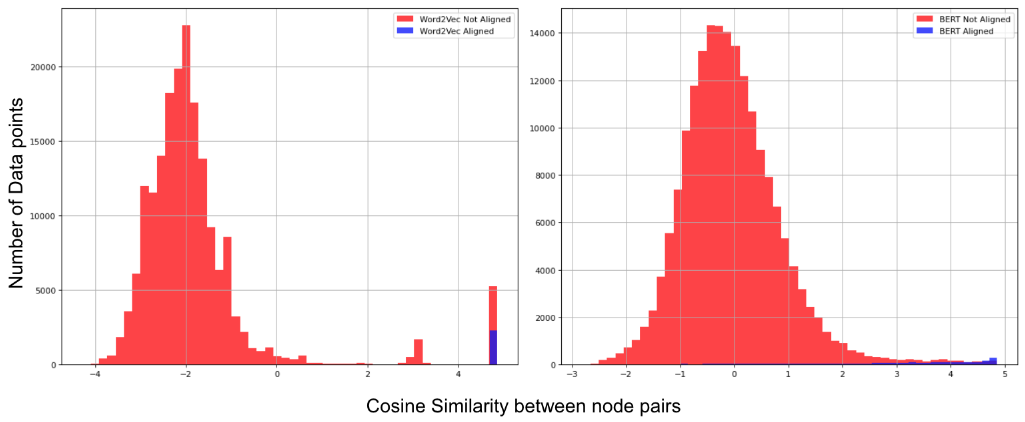


Figure 3: Histogram of cosine similarities for all node pair combinations for document-summary pairs from the training set.

5.2 Embedding Analysis

In order to analyze the performance of different embedding representations for the AMR alignment task, we closely examined the static embeddings (word2vec) and transformer embeddings (BERT). We plotted the histogram of cosine distances for all possible combinations of node pairs from document-summary pairs in the training set, using the gold star alignment annotations as a reference.

We observe the cosine similarity for aligned nodes is generally 1. This is because most aligned node pairs have the same concept label, hence the exact word2vec embeddings. However, a few node pairs do not have the same concept label but are still aligned. The cosine similarity using word2vec is not able to capture that. Another observation is that many unaligned nodes have a cosine similarity of 1. These are the cases where the nodes have the same concept label but are not aligned. Thus although word2vec captures about 87% of the gold star alignments of the training data, it is prone to false positives and negatives, resulting in high recall but inferior precision.

Unlike word2vec, the histogram for BERT embedding cosine distance for aligned node pairs is more distributed. Therefore, it does not exhibit an immediate clear distinction of cosine similarity between aligned and unaligned pairs. However, the histogram shows that the cosine similarity between BERT embeddings tends to be closer to 1 for aligned node pairs. This indicates that the BERT embeddings carry contextual signals that can aid the alignment task. We can use the information provided by BERT embeddings to rectify the errors made by word2vec embeddings by filtering out false positives and negatives, boosting the alignment model's precision while preserving the recall.

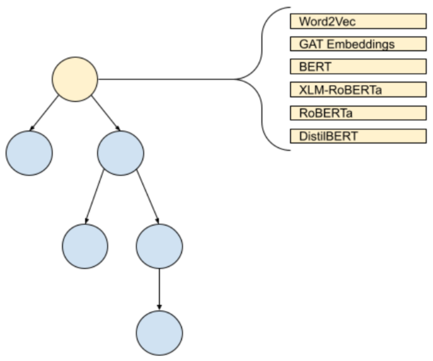


Figure 4: Representing AMR Graph Nodes with multiple embeddings including word2vec, transformer, and GAT embeddings. From our experiments we observe that using multiple embeddings helps improve the accuracy and performance of our alignment models.

The error analysis suggests we must employ a combination of the static, transformer, and GAT embeddings to represent the AMR nodes for the alignment task. Since a significant number of alignments are caught correctly by word2vec, and BERT provides a more contextual representation for the concept nodes, an ensembling classifier trained on the cosine similarities of corresponding embeddings of node pairs would give a strong baseline. We will discuss this baseline in more detail in the following section.

6 Baselines

To evaluate the effectiveness of our models, we need to establish a baseline for comparison. In this section, we will describe the various baselines that we will use to compare and test our alignment model. These baselines will provide a point of reference to help us gauge the performance of our model and determine if it is meeting our expectations. We will compare our model to these baselines in order to determine its strengths and weaknesses and identify areas for improvement.

6.1 Baseline Models

Exact Match on Concept Labels In this baseline, we perform exact matching on concept labels of the document-summary node pairs and declare the node pairs with the same concept labels as aligned. This baseline has a high recall since most of the alignment pairs have the same concept label. However, it is prone to false positives and negatives and has poor precision.

Word2Vec - Cosine distance thresholding baseline
In this baseline, we use the word2vec embeddings for representing nodes. We declare a node pair to be aligned if its cosine similarity is greater than an optimal threshold found by grid searching. This baseline provides slack by allowing nodes with different embeddings to be declared aligned if their cosine similarity is above the threshold. Like exact match, this baseline has a high recall but low precision. Since, we allow alignment of node pairs with different concept labels, we see a slight boost in precision as compared to exact match.

BERT Embeddings - Cosine distance thresholding baseline
In this baseline, we use the BERT embeddings for representing nodes. We declare a node pair to be aligned if its cosine similarity is greater than an optimal threshold found by grid searching. This baseline has a greater F1-score than the previous baselines. We see a drop in recall scores but a boost in precision suggesting we can leverage transformer-based embeddings along with word2vec embeddings to boost both precision and recall

Word2vec + BERT Ensemble Classifier
This baseline trains an ensemble classifier with word2vec and BERT cosine distances of node pairs as input features for the alignment task. The ensemble classifier gets a significant boost in the precision of the system while maintaining a high recall score. This baseline is also able to outperform the alignment model developed last semester. From the results on this baseline, we learn that combining multiple embeddings for node representations and using ensemble classifiers for node alignments is a promising approach.

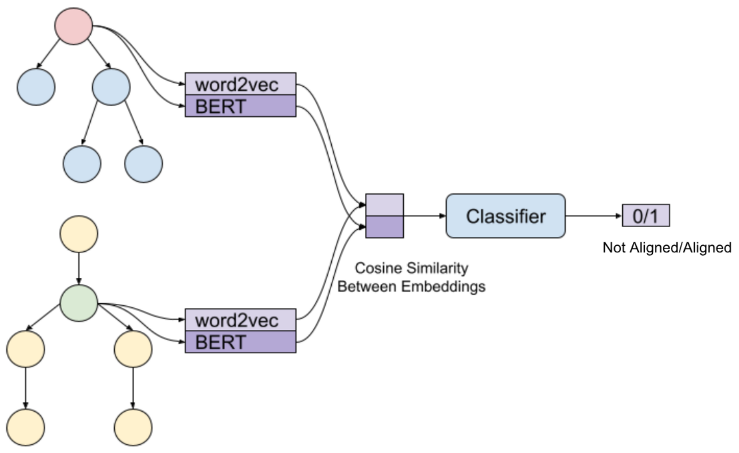


Figure 5: Word2Vec + BERT ensemble classifier for AMR graph alignment. The ensemble classifier gets a significant boost in the precision of the system while maintaining a high recall score.

Table 2: Baseline Performance.

Model	Precision	Recall	F1-Score
Exact Match	0.31	0.86	0.45
Word2Vec	0.27	0.87	0.41
BERT	0.41	0.57	0.48
GNN-Encoder-V1	0.56	0.73	0.62
Word2Vec + BERT	0.71	0.64	0.67
Word2Vec + Pre-trained LMs	0.84	0.72	0.77

6.2 Results and Analysis

In this analysis, the models with the highest recall include the exact match baseline and the word2vec baseline. However, both of these models suffer from a high rate of false positives, leading to a low precision score. The BERT embeddings improve the precision of the model by filtering out erroneous alignments, but this comes at the cost of a lower recall score. The ensemble classifier, on the other hand, achieves a higher recall than the BERT model and a significantly improved precision score compared to the other baselines. Additionally, the ensemble classifier outperforms the GNN-Encoder-V1 in both precision and F1-score.

When multiple transformer embeddings are used in addition to word2vec and BERT, the model achieves the highest performance. Adding more transformer embeddings significantly boosts the precision while maintaining a similar recall to the best-performing model from the previous semester. These results suggest that ensembling methods, especially when using multiple embeddings to represent AMR nodes, can significantly improve the performance of the alignment task.

7 Improved GAT Embeddings

To better understand the performance of the GAT encoder developed in the previous semester, we conducted error analysis and found that all of the models had high recall scores but poor precision scores. This resulted in many false positives, preventing us from accurately identifying the most important parts of the document AMR graph. Additionally, the pre-training approaches of graph reconstruction and masked node reconstruction did not improve the performance of the system, and there was a need for a better pre-training objective that could help the GNN encoder capture the structure and context of the graph.

Upon further analysis, we realized that not all of the predicted node alignments were equally important. It was crucial for the model to correctly predict the important alignments, as these enable us to locate key parts of the document AMR and identify subgraphs containing the main ideas of the document. On the other hand, we could afford to be more lenient with the less important alignments and allow the model to miss some of them.

To focus on important nodes, we used the degree centrality of nodes to assign importance weights to the alignment predictions and penalized the network based on the weight of the mistakes made. We assumed that the importance of a node can be determined by the number of interactions it has with its neighboring nodes.

In the following sections, we will detail the steps taken to improve the richness of the GAT embeddings and enhance the performance of the alignment model.

7.1 Pre-Training with Masked Node Label Prediction

The goal of pre-training the GNN encoder on the masked node label prediction task is to help it learn the structure of the AMR graph and pay attention to important neighboring nodes while performing graph convolution operations. To do this, we use a series of SuperGAT layers followed by LeakyReLU, and initialize the concept labels of the nodes with transformer embeddings. Then, we pass the graph through the GNN encoder to obtain GAT embeddings for each node, which are fed through a feed-forward network that produces logits for the concept labels of each node. These logits are passed through a softmax layer to compute the probability of each possible prediction, and we use the prediction loss to optimize the network.

During this pre-training process, we randomly mask 15% of the nodes in the input AMR graph and try to predict the concept label for the masked node based on the information provided by the remaining, unmasked nodes and edges. This helps us learn the overall structure of the graph and create embeddings that are more sensitive to that structure and the context provided by the graph.

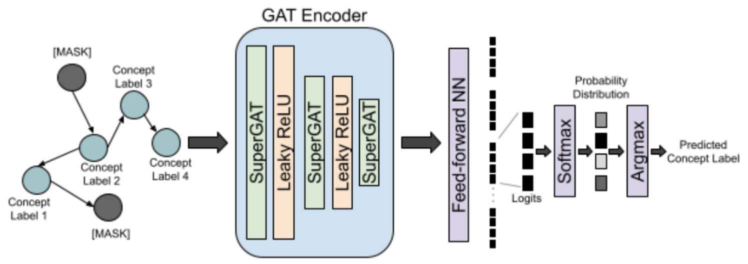


Figure 6: Masked node label prediction pre-training task. This pre-training objective helps the GNN Encoder to learn the graph structure and make the GAT embeddings more structure and context aware.

7.2 Centrality Error

To improve the performance of the GNN encoder, we introduce a new optimization criterion that focuses on getting the important node alignments correct and penalizes the network based on the severity of the errors made. This criterion is based on the concept of degree centrality, which is a measure of how important a node is in a graph based on the number of connections it has. Specifically, we define a centrality error that calculates the weighted average of false negative and false positive alignment predictions, with the weights being based on the degree centrality of the predicted alignments and normalized by the weighted sum of the degree centrality of all the document nodes. This helps guide the training of the GNN encoder to prioritize alignments that are more central to the structure of the graph.

Let G be the set of ground truth alignments and P be the set of predicted alignments. We define an underestimation error and an overestimation error.

Underestimation Error
Define $P_u = G - P$ to be the set of alignments present in ground truth but not predicted by the model. This is the set of false negatives predicted by the alignment model. The underestimation error is computed using the weighted sum of degree centrality for the document nodes in P_u. We normalize the sum by the weighted sum of degree centrality for the document nodes in the document AMR graph. The following equation shows the computation of the underestimation error.

Overestimation Error
Define $P_o = P - G$ as the set of alignments predicted by the model but not in the ground truth. This is the set of false positives predicted by the alignment model. The overestimation error is computed using the weighted sum of degree centrality for the document nodes in P_o. We normalize the sum by the weighted sum of degree centrality for the document nodes in the document AMR graph. The following equation shows the computation of the underestimation error.

The centrality error is a measure of how well the GNN encoder is aligning the important nodes in the graph. It is calculated by adding together the underestimation and overestimation errors, which represent the degree to which the GNN encoder is under- or over-predicting the alignments of these important nodes. If the GNN encoder is making absolutely correct alignment predictions, the centrality error will be 0. By using the centrality error as an optimization criterion, we can guide the GNN encoder to pay more attention to the important nodes and ensure that they are correctly aligned. This helps improve the overall performance of the network.

7.3 Learning Better GAT Embeddings

To create more structure-aware and context-aware GAT embeddings, we start by pre-training the GNN encoder on the masked node labeling task. In this task, we randomly mask out 15% of the nodes, sampling them based on the probability distribution given by their degree centrality. The GNN encoder is then trained in an unsupervised fashion to predict the concept labels of these masked nodes based on the information provided by the remaining, unmasked nodes and edges.

After pre-training, we fine-tune the GNN encoder on the alignment task. In addition to using the Triplet Margin Loss and Self Attention Loss, we also introduce the Centrality Loss as a regularization term to guide the optimization of the network. To address the issue of class imbalance in the data, we also apply oversampling techniques like SMOTE during training. Finally, we initialize the nodes with BERT embeddings to provide a strong foundation for the GAT encoder to build upon. Overall, these steps help improve the performance of the GNN encoder on the alignment task.

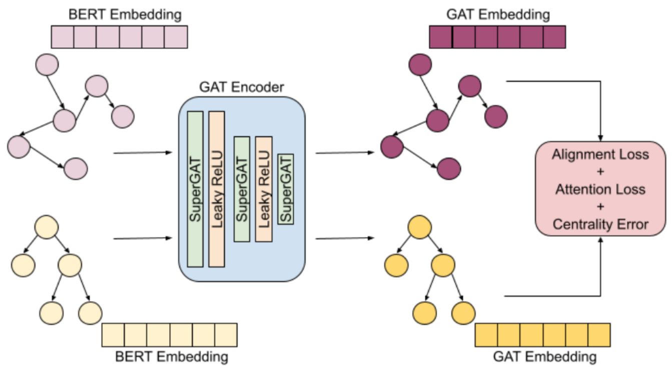


Figure 7: Learning context- and structure-aware GAT embeddings for AMR alignment.

7.4 Ablation Results

In our experiments, we conducted an ablation study to understand the impact of various strategies on the performance of the GNN encoder for the alignment task. The GNN encoder was pre-trained on a masked node labeling task and fine-tuned on the alignment task, using the Triplet Margin Loss, Self Attention Loss, and Centrality Loss as regularization terms. We also applied oversampling techniques like SMOTE

Table 3: Ablation Study for GAT Embeddings.

Model	Precision	Recall	F1-Score
GNN-Encoder-V1	0.56	0.73	0.62
GNN-Encoder-V2 (BERT initialized, Masked Label Pre-training, Centrality Error, SMOTE)	**0.64**	**0.67**	**0.70**
GNN-Encoder-V2 (w/o BERT initialization)	0.55	0.58	0.56
GNN-Encoder-V2 (w/o Masked Label Pre-training)	0.59	0.61	0.60
GNN-Encoder-V2 (w/o Centrality Error)	0.61	0.66	0.68

to address class imbalance in the data and initialized the nodes with BERT embeddings to provide a strong foundation for the GAT encoder.

We compared the performance of the GNN-Encoder-V2 model, which includes all of these strategies, to the performance of the GNN-Encoder-V1 model and three variations of the GNN-Encoder-V2 model that exclude one of these strategies. The results showed that the GNN-Encoder-V2 model performed the best in terms of F1-score, with a value of 0.70. The model without BERT initialization performed the worst, with an F1-score of 0.56. The model without masked label pre-training had an F1-score of 0.60, and the model without the centrality error criterion had an F1-score of 0.68.

These results suggest that pre-training the GNN encoder on a masked node labeling task and incorporating the Centrality Loss as a regularization term during fine-tuning can significantly improve the performance of the model on the alignment task. Initializing the nodes with BERT embeddings also appears to be an important factor in the model's performance. Our findings highlight the importance of considering both structure and context when learning GAT embeddings for alignment tasks.

8 AMR Alignment Model

8.1 Ensemble Model

In our final model, we use an ensemble approach to combine the predictions of multiple models and improve performance. This approach involves representing each node of the AMR graph with three types of embeddings: word2vec embeddings, improved GAT embeddings, and transformer-based embeddings. We use transformer embeddings from several different models, including BERT, RoBERTa, XLM-RoBERTa, and Distil-BERT.

To align a given pair of document summary AMR graphs, we consider all possible combinations of document-summary node pairs and calculate the pair-wise cosine distance between the constituent embeddings for each pair. We then use these pair-wise cosine distances as the input to a random forest classifier, which makes the final predictions for the ensemble model. This approach has proven to be effective in our experiments for aligning AMR graphs. By using an ensemble of multiple models and combining their predictions, we are able to achieve better performance than any single model alone.

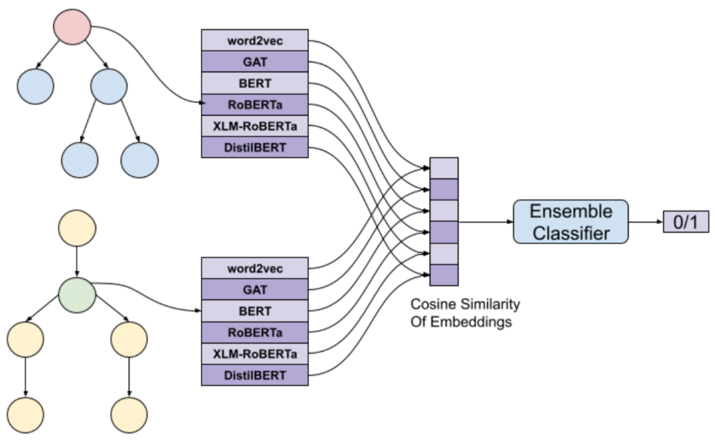


Figure 8: Ensemble AMR Alignment Model.

8.2 Results

We compared the performance of the ensemble model to several baselines and previous models. The results, shown in the table below, indicate that the ensemble model significantly outperforms the other models in terms of F1-score, while maintaining good recall performance.

The ensemble model achieved a precision of 0.91, a recall of 0.78, and an F1-score of 0.84. This represents a significant improvement over the GNN-Encoder-V1 model which had an F1-score of 0.62. The ensemble model also shows improvement over GNN-Encoder-V2 and Word2Vec + Pre-trained LMs models.

Table 4: AMR Alignment Model Performance.

Model	Precision	Recall	F1-Score
Exact Match	0.31	0.86	0.45
Word2Vec	0.27	0.87	0.41
GNN-Encoder-V1	0.56	0.73	0.62
GNN-Encoder-V2	0.64	0.67	0.70
Word2Vec + Pre-trained LMs	0.84	0.72	0.77
Ensemble Model (Word2Vec + GNN-Encoder-V2 + Pre-trained LMs)	**0.91**	**0.78**	**0.84**

One of the key benefits of the ensemble model is that it is able to significantly improve precision while maintaining good recall performance. This is particularly important in the AMR alignment task, as false positives and false negatives can have serious consequences. By focusing on getting the important node alignments correct and penalizing the network based on the severity of the errors made, the ensemble model is able to reduce the number of false positives and false negatives, resulting in a more accurate alignment of the AMR graphs.

Overall, our findings demonstrate the effectiveness of using an ensemble approach to combine multiple types of embeddings for AMR alignment. This approach allows us to capture different aspects of the AMR graph structure and context and make more accurate alignment predictions, while also reducing the number of false positives and false negatives.

9 Scoring System-Generated Summaries

9.1 Centrality-Based Score

To evaluate the quality of system-generated summaries, we use a centrality-based scoring metric that has two main steps:

Identification of salient nodes in the document graph: In this step, we identify the nodes in the document graph that are most important or relevant to the overall meaning of the document. To do this, we use multiple reference summaries and align them with the document graph. The aligned nodes of the document graph are then considered to be salient.

Aligning the candidate summary with the salient nodes and calculating the centrality-weighted graph cover: In this step, we align the generated candidate summary with the salient nodes identified in the previous step and calculate the centrality-weighted graph cover. This metric measures the extent to which the summary covers the most important nodes in the document graph and is used as a measure of the summary's quality. It is calculated as the weighted sum of the degree centrality of the document nodes in the predicted salient alignments, normalized by the weighted sum of the degree centrality of all the salient nodes in the document AMR.

This centrality-based scoring metric is a recall-based measure that tends to correlate with content-based criteria. It is used to evaluate the quality of system-generated summaries by considering both the relevance and comprehensiveness of the summary with respect to the document.

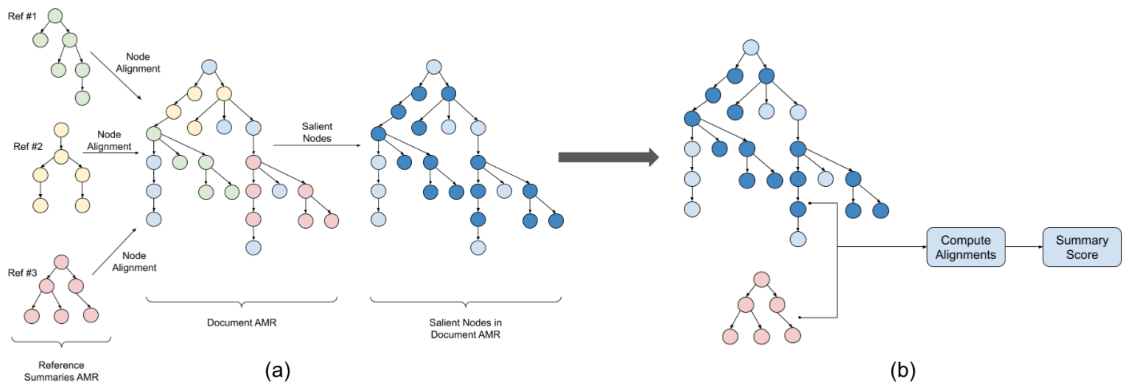


Figure 9: Centrality-based Scoring.

9.2 Results

We evaluated the correlation of this centrality-based scoring metric with several other automatic evaluation metrics, including ROUGE-1, ROUGE-2, ROUGE-L, and

Table 5: Correlation of Centrality-Score with other Automatic Metrics.

Metric	Correlation Coefficients
ROUGE-1	0.289
ROUGE-2	0.180
ROUGE-L	0.119
‘ BertScore	0.452

BertScore on a small subset of CNN/Daily Mail dataset. The results showed that the centrality-based scoring metric had moderate to low correlation with these other metrics, with the highest correlation being with BertScore (0.452). The correlation with ROUGE-1, ROUGE-2, and ROUGE-L was lower, with coefficients of 0.289, 0.180, and 0.119, respectively. Overall, these results suggest that the centrality-based scoring metric is a useful measure for evaluating the quality of system-generated summaries, but may not be highly correlated with other commonly used automatic evaluation metrics. More experiments and studies on different datasets are required to analyse the behavior and utility of the centrality-based score.

10 Conclusion and Future Work

References