Meaning Representation Parsing

Project Proposal
Computer Science Honours Project 2022

Chase Ting Chong
tngcha001@myuct.ac.za
Department of Computer Science
University of Cape Town
South Africa

Claudia Greenberg
grncla009@myuct.ac.za
Department of Computer Science
University of Cape Town
South Africa

Jane Imrie
imrjan001@myuct.ac.za
Department of Computer Science
University of Cape Town
South Africa

1 PROJECT DESCRIPTION

Meaning Representation Parsing is an area of computational linguistics. Within this area is Semantic Graph Parsing, which is the task of creating graphical meaning representation for natural language, that can be interpreted by computers [11, 28]. This topic is crucial within natural language processing (a field of computer science which combines artificial intelligence and linguistics [39]). It has, therefore, been studied extensively over recent years [8, 19, 36, 42, 48].

Semantic parsers are systems that formally derive the semantic meaning of sentences. They need to be efficient and accurate at understanding and predicting the meaning of sentences. The resulting graphical representations are used within various fields such as robotics (understanding commands) and data exploration (understanding varying amounts of data) [28]. The overall goal of these parsers is to correctly process sentences of the many languages without manual (human) processing. Many studies, such as [2, 4, 7, 9, 11, 12, 17, 29, 31, 33, 37, 46, 51], have been published, whereby researchers create, analyse, and compare different semantic parsers. Each one intends to use a novel approach or outperform existing, similar models using slight methodological variations.

This project, too, attempts to combat the problem of creating some formal representation for a given English sentence that is interpretable by computers. We propose two different approaches to predicting the interpretation of English sentences using neural networks. We will be focusing on a computational framework called Elementary Dependency Structures (EDS). This task deals with constructing meaning representation graphs for (solely English, in our scope) sentences. Each graph consists of labelled nodes and edges. The nodes represent the tokenised input and can consist of either a single token, multiple tokens or sub-tokens. The edges represent the relationships between the different nodes.

This Project Proposal introduces the topic that we are researching and developing within. We begin with our motivation for undertaking this project, reviewing the problems we are focusing on. We then delve into various sections of related work. Subsequently, we discuss how we aim to address our project's problems. We then explain the methodologies that we adopt and provide a section on the project's ethics. We conclude this proposal with our expected outcomes and our project plan.

2 RESEARCH MOTIVATION

2.1 Problem Statement

Semantic parsing is a broad field with many approaches aiming to provide differing insights into the semantic meanings of sentences and represent these meanings in various ways [28]. Our research focuses on two main approaches of semantic graph parsing, transition-based [4] and graph-based [24]. Recent implementations [4, 24] of these approaches make use of neural networks such as Long Short-Term Memory or general Recurrent Neural networks, each having their own advantages and disadvantages.

Transformers [50] are a newer kind of neural network that solve the problems faced by RNNs (vanishing gradients, limited context) and also provide better performance as they are easily parallelizable and require significantly less time to train. Additionally pre-training transformers (§3.2) has shown benefits in many NLP processing tasks [13] but have yet to be fully explored for semantic graph parsing. Our research aims to address this gap in the literature.

2.2 Transition-Based Approach

The transition-based approach to semantic graph parsing transforms the task of predicting a graph to predicting a sequence of actions that when run on a stack-based transition system will construct the graph [40]. The advantage of this approach is its efficient implementation and linear run time [21]. Additionally, an end-to-end sequence-to-sequence model removes the need for a complex pipeline found in traditional models and has been found to outperform these models in many NLU tasks [4, 54]. As a testament to this, with the emergence of pre-trained transformers, the current state-of-the-art parsers for EDS and Abstract Meaning Representation (AMR) are [44] and [3] respectively both of which are sequence-to-sequence parsers. Furthermore [18] achieves state-of-the-art results for AMR parsing making use of [54] which combines a sequence-to-sequence pre-trained transformer encoder-decoder model and a transition-based system.

2.3 Graph-Based Approach

A secondary approach to semantic graph parsing is the graph-based approach, whereby a graph is constructed to represent the semantic meaning of a given sentence. There are two main sections to this: node prediction and edge prediction. The nodes either directly or indirectly represent the words of the sentence. The edges represent the relations between these nodes.

The process of predicting the nodes and edges can either be integrated, as seen in [24, 25], or separated, as seen in [7, 8, 10, 53]. Our proposal aims to use the latter strategy, separating node and

edge prediction in a pipeline format. This means that the output of the node prediction module will serve as the input to the edge prediction module.

2.3.1 Node Prediction. Node prediction can be formulated as as a sequence labelling i.e. tagging problem. The labels for each of the nodes can be encoded as tags, with the optional addition of encoding the spans as tags too.

There is some existing literature for semantic graph parsing which separates the task of predicting the nodes and edges [7] [10]. Both of these sources were able to predict the labels for spans and tokens with a high degree of accuracy, using either BERT [10] or ELMO [7] contextualized word embeddings, and BiLSTMS with a softmax layer for classification. However, there is a gap in the literature for using the statistical modelling method of conditional random fields, which is a sequence labeller that takes context into account when predicting labels [32], in combination with contextualised word embeddings and pre-trained transformer models. Additionally, the existing research has not done controlled evaluations comparing BERT, ELMO and biLSTMs.

2.3.2 Edge Prediction. Given the nodes of the graph, the second stage of the pipeline is to understand how the nodes relate to one another. This provides crucial understanding and context.

A variation within edge prediction modules is the choice of training objectives. Two training objectives, which have been used in existing studies, are outlined below.

The first type is a maximum entropy objective, whereby the model is optimised to produce the most accurate graphs with minimal error (loss). This has been widely used in various areas of natural language processing (NLP) [47]. Papers which have used this for edge prediction modules include [30, 36, 51].

The second objective is maximum-margin. Cao et al. [7], for example, use a maximum-margin-based objective for training their data for their dependency identification module of their data-intensive parser. They first solve an optimisation problem to calculate their best graph prediction. They then analyse similarity by calculating the loss between the "gold" (actual) graph and the parser's most accurate predicted graph. This loss is then optimised by updating the model's parameters.

Another variation within the design of edge prediction modules is the encoder used during the process. Our focus is on the encoding which occurs during the biaffine network process (an edge prediction scoring function, explained §5.3.2).

Pre-trained BERT (introduced by Devlin et. al [13]), used by papers such as [44], has been used as an attractive encoder choice for both node and edge prediction modules, as opposed to other encoders such as ELMo and LSTMs [52]. Additionally, papers such as [38] have combined different encoders. The related gap in the literature has been discussed in the previous section (§2.3.1).

3 RELATED WORK

3.1 Neural Networks

A neural network is modelled after neurons and their function in the human brain. These networks are learning models that consist of small computing units, each of which takes in a vector as an input and provides a single value as an output [27]. Neural networks are used across a wide variety of Natural Language Processing tasks. Previous research has shown they can be effectively utilized for semantic parsing [15][46][5] [7] and other types of NLP tasks, such as sequence labelling in the form of part-of-speech (POS) tagging and named-entity recognition [35] [25] .

3.2 Pre-Trained Transformer

A transformer is a type of neural network [26]. Pre-training is a technique used to increase the detail and precision of a parser. The transformer is first trained with a text corpus using a self-supervised approach. Following this, fine-tuning takes place using a linear or feedforward neural network. A popular pre-trained transformer is the Bidirectional Encoder Representations from Transformers (BERT). Pre-training allows for more detailed analyses of the parsers' languages as it begins the process with pre-existing context and understanding.

3.3 Semantic Parsing

3.3.1 Semantic Dependency Parsing. Semantic graph parsing draws a lot of inspiration, concepts and algorithms from semantic dependency parsing. Dependency relations consist of a head (the word upon which the relation depends) and the dependent – these are also known as arguments and predicates [41]. These relations are represented as labelled arcs from head to dependent. Semantic dependency parsing tries to approximate semantic relationships by means of these dependency relations [27]. Semantic parsers are can be graph-based or transition based, amongst others [41].

3.3.2 Semantic Graph Parsing. Semantic graph parsing considers the problem of semantic parsing but models it as a graph prediction problem. While semantic dependency parsing aims to construct trees, here, the aim is to have meaning representations in a graph form [20].

3.3.3 Span Prediction. A span can be thought of as a series of words that can be grouped together under one label [27]. For example, "Cape Town" can be thought of as a span that has the label GEO (because it is a geographical location). For English semantic parsing, a sentence can be broken up (using the whitespaces as delimiters) into a set of tokens, where each token corresponds to a word. Then, a neural network can be used to predict the label for each of these tokens [10]. However, it is also possible to split a sentence into a set of spans, and then assigning labels to these spans rather. This is known as span prediction and one tagging method for these spans is BIO tagging [27]. Some research has been done to address some of the problems that arise when trying to perform span prediction, such as what happens when there are multiple overlapping spans in a sentence [24]. Furthermore, there has been research that shows that many NLP tasks can be performed with a generalised, task-agnostic model (as opposed to a task-specific architecture) if that task can be framed as a span-relation prediction problem [25] .

4 RESEARCH OBJECTIVES

4.1 General

The main goal of our research is to address the gap presented in §2.1 we thus aim to answer the following research question:

- How does the F1 score of pre-trained transformers compare to the F1 score of RNN-based approaches to semantic graph parsing?

4.2 Transition-Based System

Recently pre-trained transformer encoders have been widely explored in many semantic graph parsers [54]. However new research on full pre-trained transformer encoder-decoder models has shown state-of-the-art results in AMR parsing [3]. We aim to apply this model to EDS parsing by making use of BART [34], a pre-trained encoder-decoder transformer, to answer the following research questions:

(1) How does the F1 score of modelling the output sequence as a top-down linearisation of the graph compare to the F1 score of modelling it as a transition sequence on the BART model?
(2) How does the F1 score of the BART model compare to the F1 score of RNN based encoder-decoder models?

4.3 Graph-Based: Node Prediction

Graph-based semantic parsing can be split up into two parts: node prediction, and edge prediction. When predicting nodes, one can either include or elect not to encode the span information in the tags. For this research, we will attempt to include the span information in the tags - this approach can then be compared to previous attempts which do not do this. This notion can be further qualified by means of the following research questions:

(1) How does a BERT encoder with an added CRF layer compare to an LSTM with a standard word-embedding layer in terms of accuracy of node prediction i.e. how does BERT affect the precision, recall and SMATCH scores of the model?
(2) How does encoded span information for each node compare against no span information in terms of accuracy of the predicted labels for a set of tokens?

4.4 Graph-Based: Edge Prediction

Given the nodes of the semantic graph, and their corresponding labels, the parser must now predict the edges by scoring them individually and then choosing the most suitable combination to complete the graph. There have been various attempts at this. We aim to produce an accurate edge prediction module to form the second pipelined stage of our graph-based semantic parser. We aim to answer the following questions:

(1) How does the accuracy of an edge prediction module with a maximum entropy training objective compare to pre-existing edge prediction modules that utilise a maximum margin training objective?
(2) How does the accuracy of this edge prediction module, developed using the pre-trained BERT, compare to the pre-existing parsers' edge prediction modules developed with non-trained LSTMs?

5 PROCEDURES AND METHODS

5.1 Text Corpus

Data will be sourced from the LinGo Redwoods corpus [43]. This corpus contains English sentences that have been annotated for use in both syntactic and semantic parsing. This data will be pre-processed using the PyDelphin library in order to get it into the correct form for use with the parser. PyDelphin is a Python module that can be used for modelling Minimal Recursion Semantics [23] and will extract the EDS graphs from the data. Each team member may then further process the data into a form that is compatible with their parser.

5.2 Transition-Based Approach

In this approach we will build on the work of [4] by replacing the RNN encoder-decoder with BART [34] and fine tuning it to meet our needs.

5.2.1 Architecture. We will use the Hugging Face implementation of the pre-trained transformer encoder-decoder model BART as a base for a top-down graph linearisation approach as well as a transition-based system approach.

For the top-down approach the graph data from the corpus will be linearised using PyDelphin. PyDelphin has a built-in algorithm to convert EDS graphs into PENMAN notation. This notation will then be pre-processed, similar to the work of [4], to remove unnecessary data and be used as a linear sequence to train the transformer model.

For the transition-based approach an oracle based on [4] will be used to derive the gold transition sequence from the corpus data and used to train the transformer. However unlike in [4] the parser state will not be directly encoded into the model but will be learned implicitly from the gold sequence.

To predict the alignments, we will follow the method used by [4] in their top-down approach, a token corresponding to the alignment will be generated in the output sequence. This should be compatible with transformers as they use positional embeddings.

5.3 Graph-Based Approach

5.3.1 Concept Identification. The methods and procedures for this project are drawn strongly from [7], although there are some differences in terms of the use of BERT (as opposed to ELMO) and the employment of a controlled evaluation of the contextualised embeddings against standard embeddings (which act as a baseline in this research).

If a node's label has a conceptual meaning, then a node can be referred to as a concept [10] and the concept identification problem can be framed as a word tagging problem. Thus, given an input sentence that has been split into tokens or spans, node prediction can be seen as predicting the labels that correspond to these tokens/spans. This process can then be formulated as a tagging problem - each token derived from the sentence can be associated with a tag.

In practice for this project, once a sentence has been split up into tokens, the transformer model will receive this as input and predict which tokens correspond to a node, as well as what tag that token should receive. Nodes can correspond to sub-tokens, multiple tokens or single tokens. In order to achieve all of this, the span information for each token must be encoded. The Flair library [1] will be used to produce the tags.

For neural networks, both the words and the tags need to be represented as continuous and dense features. Thus, a pre-trained BERT encoder will be used, together with a CRF layer and a softmax

layer for classification, resulting in the output of a node. A conditional random field (CRF) is a sequence labeller which uses posterior probabilities in order to compute the probability of an entire sequence of tags, i.e. given $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$, where X is the set of input words and Y the set of output tags, a CRF computes $p(Y|X)$, which can be computed as [27]:

$$p(Y|X) = \frac{exp(\sum_{k=1}^{K} w_k F_k(X, Y))}{\sum_{Y' \in \mathcal{Y}} exp(\sum_{k=1}^{K} w_k F_k(X, Y'))} \quad (1)$$

The softmax classification layer will take input from all the other layers and output a vector of probabilities. Mathematically, this can be formulated as :

$$z = z_1, \ldots, z_k$$
$$softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \quad (2)$$

Where z represents the output from each node in the final hidden layer, which is passed to the output layer and transformed using the softmax function to obtain a discrete probability distribution over all possible outcomes [22].

Architecture: A machine learning pipeline will be created to perform the node prediction, otherwise viewed as concept identification. This pipeline will consist of several stages, namely – a data preprocessing stage, a traini datang stage, evaluation stage, an testing stage and finally an output stage, which will be used to create output data that can then be fed into the edge prediction component of the parser. A predefined split will be used to separate the data into three subsets for the training, test and evaluation stages. PyTorch will be used to create the neural network [45].

Since this is a supervised learning problem, the model will first be given labelled training data. During training, the cross-entropy loss function will be used. For the baseline, this loss is independently computed for each of the tags, but for the CRF, the joint probability of the entire tag sequence is first computed and then the cross-entropy loss is calculated on that. Once it has been trained, this model will then be evaluated. During this stage, some hyperparameter tuning will be done to ensure that the optimal parameters for the model have been chosen. Finally, the testing set will be fed to the model. This stage is needed to check that the model can perform well on unseen data and is somewhat generalisable.

5.3.2 Relation Identification. In order to train the module, we will obtain the nodes of the graph from the text corpus. We will need to ensure that they are in the appropriate form, identical to what would be produced by our node prediction module (which will be developed in parallel to this module).

Two main tasks need to be completed to create an accurate edge prediction module for the graph-based parser: how to predict the individual edges (relations) and, subsequently, how to predict the overall graph.

An edge is represented in the form: $\mathbf{h}_i^{head} \rightarrow \mathbf{h}_i^{dep}$.

A common, effective approach to scoring the individual edges is a biaffine network architecture [14], used by papers including [30, 38, 52, 53]. Figure 1 depicts the network's structure and is explained below.

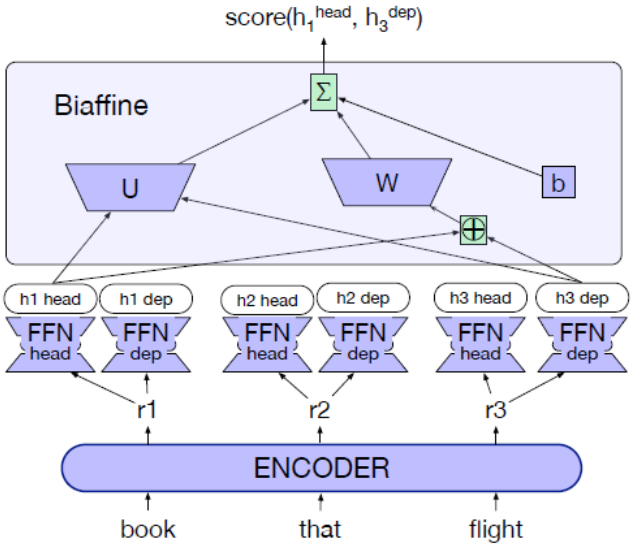

Figure 1: A Biaffine Network [26]

The inputted nodes are first sent through a BERT encoder. The start and end tokens of the nodes(' spans) are encoded and these are concatenated with the node label embeddings. As the model is trained, the BERT's weights can be fine-tuned to improve overall accuracy.

These representations are then sent through two feedforward neural networks (FFN), one to calculate the heads (edge and label), and the other to calculate the dependents (dep) (edge and label):

$$\mathbf{h}_i^{(\text{edge-head})} = \text{FFN}^{(\text{edge-head})}(\mathbf{r}_i) \quad (3)$$
$$\mathbf{h}_i^{(\text{edge-dep})} = \text{FFN}^{(\text{edge-dep})}(\mathbf{r}_i) \quad (4)$$
$$\mathbf{h}_i^{(\text{label-head})} = \text{FFN}^{(\text{label-head})}(\mathbf{r}_i) \quad (5)$$
$$\mathbf{h}_i^{(\text{label-dep})} = \text{FFN}^{(\text{label-dep})}(\mathbf{r}_i) \quad (6)$$

where $\mathbf{h}_i^{head}$ is the embedding representation of the head token $\mathbf{r}_i$ and $\mathbf{h}_i^{dep}$ is the embedding representation of the dependent token $\mathbf{r}_i$.

These units are then sent to a biaffine scoring function, which will calculate the edges' and edge labels' scores:

$$\mathbf{s}_{ij}^{(\text{edge})} = \text{Biaff}^{(\text{edge})}(\mathbf{h}_i^{(\text{edge-head})}, \mathbf{h}_j^{(\text{edge-dep})}) \quad (7)$$
$$\mathbf{s}_{ij}^{(\text{label})} = \text{Biaff}^{(\text{label})}(\mathbf{h}_i^{(\text{label-head})}, \mathbf{h}_j^{(\text{label-dep})}) \quad (8)$$

where $\mathbf{s}_{ij}$ is the score, *Biaff* is the biaffine scoring function, and

$$\text{Biaff}(\mathbf{x}, \mathbf{y}) = \mathbf{x}^{\text{T}}\mathbf{U}\mathbf{y} + \mathbf{W}(\mathbf{x} \bigoplus \mathbf{y}) + b \quad (9)$$

where $\mathbf{x}$, $\mathbf{y}$ are the representations, $\mathbf{U}$ and $\mathbf{W}$ are weight matrices, and b is the calculated bias term.

The edges' scores are then converted into a binary classification because an edge is either included or excluded.

If an edge is included, a softmax function is used to calculate which edge label is most likely [51].

$$P(y_{ij}^{(\text{label})}|\mathbf{w}) = \text{softmax}(\mathbf{s}_{ij}^{(\text{label})}) \quad (10)$$

where $P(y_{ij}^{(\text{label})}|\mathbf{w})$ is the probability score.

To train the above biaffine network, we will need to calculate the cross-entropy loss [26]. This measures the performance of the network by reviewing the network-assigned probability given to each edge in comparison to the correct edges. This training method can be used for various tasks, not just for edge prediction [36, 44, 51].

Subsequently, to find the most suitable graph for the inputted sentence, a common approach for utilising these individual edge scores is to find the maximum spanning tree, which can be found using an algorithm such as the Chu-Li Edmonds Algorithm [26]. A maximum spanning tree is a graph with a tree that includes all nodes and the edges connecting them yield the highest possible score. This recursive algorithm is shown in Figure 2. In this algorithm, a greedy approach is first taken by finding the best incoming edge for each node. If this combination is a spanning tree, the algorithm stops. Else, a cycle must be occurring. This cycle is deleted by recursively contracting the graph, finding the maximum spanning tree of this graph, and expanding it back while deleting the cycle-causing edge. Without cycles, the maximum spanning tree can be found.

```
function MaxSpanningTree(G=(V,E), root, score) returns spanning tree

    F ← []
    T' ← []
    score' ← []
    for each v ∈ V do
        bestInEdge ← argmax_{e=(u,v)∈ E} score[e]
        F ← F ∪ bestInEdge
        for each e=(u,v) ∈ E do
            score'[e] ← score[e] − score[bestInEdge]

    if T=(V,F) is a spanning tree then return it
    else
        C ← a cycle in F
        G' ← Contract(G, C)
        T' ← MaxSpanningTree(G', root, score')
        T ← Expand(T', C)
        return T

function Contract(G, C) returns contracted graph

function Expand(T, C) returns expanded graph
```

Figure 2: The Chu-Li Edmonds Algorithm [26]

Once the maximum spanning tree is found, additional edges are added by greedily predicting non-tree edges. This completes the full predicted semantic graph.

We aim to handle our edge prediction with a trained biaffine network for individual edges and finding the maximum spanning tree for the overall graph.

5.4 Evaluation

We require a careful evaluation of our parsers in order to deem its viability and accuracy levels. We conduct these evaluations during the training and testing phases. The overall aim is to compare our parsers' predictions to what is correct.

The above approaches (both transition-based and both stages of graph-based) will be evaluated using four main metrics: precision (ratio of correct to total labels), recall (ratio of correct to total expected labels), F1 score (the "harmonic mean of precision and recall)[26] and SMATCH [6].

Elementary Dependency Matching (EDM) [16] will be used to calculate the precision, recall and F1 score. EDM converts the gold and parser-generated graphs into node and edge tuples. A node tuple consists of the node label and character span whereas an edge tuple consists of the character spans of the head and dependant node, as well as the edge label. The tuples from the respective graphs can then be matched to compute the aforementioned metrics for the nodes, edges and overall graph. The node prediction will use the node EDM, the edge prediction the edge EDM and overall EDM, and the transition-based approach will use all three.

SMATCH compares the degree of overlap for whole sentence semantic structures in our case the output graph. The SMATCH score is calculated by performing a one-to-one mapping between the nodes of the output graph and gold graph to estimate a maximum obtainable F1 score. Since SMATCH requires the full graph it will only be applicable to the transition-based approach and edge prediction section of the graph-based approach.

6 ANTICIPATED OUTCOMES

We expect two semantic parsers to be fully designed, developed (predominantly using PyTorch) and analysed, one transition-based and one graph-based with node prediction and edge prediction treated as separate components. We expect both parsers to perform well in comparison to pre-existing models, outperforming the baseline models, in terms of precision, recall and F1 scores.

We expect that our research questions be answered to the best of our knowledge given the experience of building the parsers and performance results of such models. We also expect to make an impact within the meaning representation parsing research field by producing another two well-performing parsers using a unique combination of design factors.

6.1 Key Success Factors

To judge the success of the project, we need to consider the following conditions:

- We must successfully create both a transition-based and a graph-based semantic parser, the latter of which must be able to predict the nodes and edges of the graph.
- Both parsers must have a high degree of accuracy - measured in terms of various metrices, namely: precision, recall, F1 score and SMATCH.
- We must be able to generate results from these parsers that can be used to evaluate our models and compare them to pre-existing models.
- We must be able to answer the research questions posed and address the original problem statement.

7 ETHICAL, PROFESSIONAL AND LEGAL ISSUES

This project will not involve any user evaluation – thus there is no need to receive ethics clearance. The data that will be used is sourced from the LinGo Redwoods Treebank and is in the public domain. However, it should be noted that the parser may not be accurate across all domains, as the data is not sourced from one

particular place, but is rather a composition from various sources, including the Wall Street Journal. Care should also be taken not to encode any potential biases into the neural networks themselves. While some small bias is unavoidable – steps should be taken to reduce this as much as possible.

8 PROJECT PLAN

8.1 Risks

Please see Appendix A.

8.2 Timeline and Milestones

Please see Appendix B.

8.3 Deliverables

Deliverable	Due Date
Proposal	27-05-2022
Proposal Presentation	24 to 25-05-2022
Revised Proposal submission	15-07-2022
Initial Software Feasibility Demonstration	25 to 29-07-2022
Draft of Final paper	23-8-2022
Project Paper final submission	02-09-2022
Project Code final submission	05-09-2022
Final Project Demo	19 to 23-09-2022
Project Poster	03-10-2022
Project Website	10-10-2022
School of IT showcase	TBC

8.4 Required Resources

The data used for this project will be sourced from the LinGo Redwoods corpus [43]. This corpus was chosen due to the fact that it has rich and deep semantic and syntactic representations for English sentences [49].

In terms of the software required for this project, a number of Python libraries will be used. Python will be the primary programming language used to create the neural networks and a number of python libraries will be used. PyTorch is an open-source machine learning framework that contains many functions necessary for building deep neural networks [45] and will be used for both parsers. PyDelphin is another library that will be utilised - it can build semantic representations according to a number of frameworks [23], and this project will use the EDS framework. The data sourced from the corpus will be fed into PyDelphin in order to build the semantic graphs which will be evaluated against the one's created by the parsers. Futhermore, the Flair library will be needed in order to perform the tagging of the tokens and spans. Flair is a framework that can be used to perform various NLP tasks such as part-of-speech tagging [1]. As the project progresses, some more supplementary libraries may be needed in order to perform experiments and format the resulting data, but these are the aforementioned main libraries that have been identified as being essential to the project.

Secondly, there are some hardware requirements. Training neural networks will require access to powerful hardware, and thus access will be needed to some GPUs. The University of Cape Town's High Performance Computing cluster represents a suitable candidate for this.

8.5 Work Allocation

The team members will all collaborate on shared deliverables – such as the project proposal and project poster. Chase Ting Chong will design and implement a transition-based semantic parser. Jane Imrie will create one half of a graph-based parser that utilises transformers, which is responsible for node prediction. Claudia Greenberg will create the second half of the parser, which will perform the edge prediction.

ACKNOWLEDGMENTS

We would like to acknowledge our project supervisor, Dr Jan Buys, for his continuous support and guidance.

REFERENCES

[1] Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual String Embeddings for Sequence Labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*. 1638–1649.

[2] Hongxiao Bai and Hai Zhao. 2019. SJTU at MRP 2019: A Transition-Based Multi-Task Parser for Cross-Framework Meaning Representation Parsing. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*. Association for Computational Linguistics, Hong Kong, 86–94. https://doi.org/10.18653/v1/K19-2008

[3] Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. 2021. One SPRING to Rule Them Both: Symmetric AMR Semantic Parsing and Generation without a Complex Pipeline. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 14 (May 2021), 12564–12573. https://ojs.aaai.org/index.php/AAAI/article/view/17489

[4] Jan Buys and Phil Blunsom. 2017. Robust Incremental Neural Semantic Graph Parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, 1215–1226. https://doi.org/10.18653/v1/P17-1112

[5] Jan Buys and Phil Blunsom. 2017. Robust incremental neural semantic graph parsing. *arXiv preprint arXiv:1704.07092* (2017).

[6] Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 748–752.

[7] Junjie Cao, Zi Lin, Weiwei Sun, and Xiaojun Wan. 2021. Comparing Knowledge-Intensive and Data-Intensive Models for English Resource Semantic Parsing. *Computational Linguistics* 47, 1 (2021), 43–68.

[8] Jie Cao, Yi Zhang, Adel Youssef, and Vivek Srikumar. 2019. Amazon at MRP 2019: Parsing Meaning Representations with Lexical and Phrasal Anchoring. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*. Association for Computational Linguistics, Hong Kong, 138–148. https://doi.org/10.18653/v1/K19-2013

[9] Wanxiang Che, Longxu Dou, Yang Xu, Yuxuan Wang, Yijia Liu, and Ting Liu. 2019. HIT-SCIR at MRP 2019: A Unified Pipeline for Meaning Representation Parsing via Efficient Training and Effective Encoding. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*. Association for Computational Linguistics, Hong Kong, 76–85. https://doi.org/10.18653/v1/K19-2007

[10] Yufei Chen, Yajie Ye, and Weiwei Sun. 2019. Peking at MRP 2019: Factorization- and composition-based parsing for elementary dependency structures. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*. 166–176.

[11] Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2017. Learning structured natural language representations for semantic parsing. *arXiv preprint arXiv:1704.08387* (2017).

[12] Shrey Desai and Ahmed Aly. 2021. Diagnosing Transformers in Task-Oriented Semantic Parsing. *arXiv preprint arXiv:2105.13496* (2021).

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[14] Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734* (2016).

[15] Timothy Dozat and Christopher D Manning. 2018. Simpler but more accurate semantic dependency parsing. *arXiv preprint arXiv:1807.01396* (2018).

[16] Rebecca Dridan and Stephan Oepen. 2011. Parser Evaluation Using Elementary Dependency Matching. In *Proceedings of the 12th International Conference on Parsing Technologies*. Association for Computational Linguistics, Dublin, Ireland, 225–230. https://aclanthology.org/W11-2927

[17] Kira Droganova, Andrey Kutuzov, Nikita Mediankin, and Daniel Zeman. 2019. ÚFAL-Oslo at MRP 2019: Garage Sale Semantic Parsing. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*. Association for Computational Linguistics, Hong Kong, 158–165. https://doi.org/10.18653/v1/K19-2015

[18] Andrew Drozdov, Jiawei Zhou, Radu Florian, Andrew McCallum, Tahira Naseem, Yoon Kim, and Ramon Fernandez Astudillo. 2022. Inducing and Using Alignments for Transition-based AMR Parsing. arXiv:2205.01464 [cs.CL]

[19] Yantao Du, Fan Zhang, Weiwei Sun, and Xiaojun Wan. 2014. Peking: Profiling syntactic tree parsing techniques for semantic graph parsing. In *Proceedings of the 8th international workshop on semantic evaluation (semeval 2014)*. 459–464.

[20] Federico Fancellu, Sorcha Gilroy, Adam Lopez, and Mirella Lapata. 2019. Semantic graph parsing with recurrent neural network DAG grammars. *arXiv preprint arXiv:1910.00051* (2019).

[21] Ramón Fernandez Astudillo, Miguel Ballesteros, Tahira Naseem, Austin Blodgett, and Radu Florian. 2020. Transition-based Parsing with Stack-Transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, Online, 1001–1007. https://doi.org/10.18653/v1/2020.findings-emnlp.89

[22] Yoav Goldberg. 2016. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research* 57 (2016), 345–420.

[23] Michael Wayne Goodman. 2019. A Python Library for Deep Linguistic Resources. In *2019 Pacific Neighborhood Consortium Annual Conference and Joint Meetings (PNC)*. Singapore.

[24] Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly predicting predicates and arguments in neural semantic role labeling. *arXiv preprint arXiv:1805.04787* (2018).

[25] Zhengbao Jiang, Wei Xu, Jun Araki, and Graham Neubig. 2019. Generalizing natural language analysis through span-relation representations. *arXiv preprint arXiv:1911.03822* (2019).

[26] Daniel Jurafsky and James H. Martin. 2021. *Speech & Language Processing*. Pearson Education India.

[27] Daniel Jurafsky and James H. Martin. 2021. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* (3rd ed.). Prentice Hall PTR, USA.

[28] Aishwarya Kamath and Rajarshi Das. 2018. A survey on semantic parsing. *arXiv preprint arXiv:1812.00978* (2018).

[29] Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. *arXiv preprint arXiv:1805.01052* (2018).

[30] Yuta Koreeda, Gaku Morio, Terufumi Morishita, Hiroaki Ozaki, and Kohsuke Yanai. 2019. Hitachi at MRP 2019: Unified Encoder-to-Biaffine Network for Cross-Framework Meaning Representation Parsing. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*. Association for Computational Linguistics, Hong Kong, 114–126. https://doi.org/10.18653/v1/K19-2011

[31] Artur Kulmizev, Miryam de Lhoneux, Johannes Gontrum, Elena Fano, and Joakim Nivre. 2019. Deep Contextualized Word Embeddings in Transition-Based and Graph-Based Dependency Parsing–A Tale of Two Parsers Revisited. *arXiv preprint arXiv:1908.07397* (2019).

[32] John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. (2001).

[33] Sunny Lai, Chun Hei Lo, Kwong Sak Leung, and Yee Leung. 2019. CUHK at MRP 2019: Transition-Based Parser with Cross-Framework Variable-Arity Resolve Action. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*. Association for Computational Linguistics, Hong Kong, 104–113. https://doi.org/10.18653/v1/K19-2010

[34] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 7871–7880. https://doi.org/10.18653/v1/2020.acl-main.703

[35] Hongwei Li, Hongyan Mao, and Jingzi Wang. 2021. Part-of-Speech Tagging with Rule-Based Data Preprocessing and Transformer. *Electronics* 11, 1 (2021), 56.

[36] Zuchao Li, Hai Zhao, Zhuosheng Zhang, Rui Wang, Masao Utiyama, and Eiichiro Sumita. 2019. SJTU-NICT at MRP 2019: Multi-task learning for end-to-end uniform semantic graph parsing. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*. 45–54.

[37] Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *11th Conference of the European Chapter of the Association for Computational Linguistics*. 81–88.

[38] Seung-Hoon Na, Jinwoon Min, Kwanghyeon Park, Jong-Hun Shin, and Young-Kil Kim. 2019. JBNU at MRP 2019: Multi-level Biaffine Attention for Semantic Dependency Parsing. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*. Association for Computational Linguistics, Hong Kong, 95–103. https://doi.org/10.18653/v1/K19-2009

[39] Prakash M Nadkarni, Lucila Ohno-Machado, and Wendy W Chapman. 2011. Natural language processing: an introduction. *Journal of the American Medical Informatics Association* 18, 5 (2011), 544–551.

[40] Joakim Nivre. 2008. Algorithms for Deterministic Incremental Dependency Parsing. *Computational Linguistics* 34, 4 (2008), 513–553. https://doi.org/10.1162/coli.07-056-R1-07-027

[41] Joakim Nivre. 2010. Dependency Parsing. *Language and Linguistics Compass* 4, 3 (2010), 138–152. https://doi.org/10.1111/j.1749-818X.2010.00187.x arXiv:https://compass.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1749-818X.2010.00187.x

[42] Stephan Oepen, Omri Abend, Jan Hajic, Daniel Hershcovich, Marco Kuhlmann, Tim O'Gorman, Nianwen Xue, Jayeol Chun, Milan Straka, and Zdenka Uresova. 2019. MRP 2019: Cross-Framework Meaning Representation Parsing. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*. Association for Computational Linguistics, Hong Kong, 1–27. https://doi.org/10.18653/v1/K19-2001

[43] Stephan Oepen, Dan Flickinger, Kristina Toutanova, and Christopher D Manning. 2004. Lingo redwoods. *Research on Language and Computation* 2, 4 (2004), 575–596.

[44] Hiroaki Ozaki, Gaku Morio, Yuta Koreeda, Terufumi Morishita, and Toshinori Miyoshi. 2020. Hitachi at MRP 2020: Text-to-Graph-Notation Transducer. In *Proceedings of the CoNLL 2020 Shared Task: Cross-Framework Meaning Representation Parsing*. Association for Computational Linguistics, Online, 40–52. https://doi.org/10.18653/v1/2020.conll-shared.4

[45] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[46] Hao Peng, Sam Thomson, and Noah A Smith. 2017. Deep multitask learning for semantic dependency parsing. *arXiv preprint arXiv:1704.06855* (2017).

[47] Ruhi Sarikaya, Geoffrey E Hinton, and Anoop Deoras. 2014. Application of deep belief networks for natural language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22, 4 (2014), 778–784.

[48] Bo Shao, Yeyun Gong, Weizhen Qi, Guihong Cao, Jianshu Ji, and Xiaola Lin. 2020. Graph-based transformer with cross-candidate verification for semantic parsing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 8807–8814.

[49] Kristina Toutanova, Christopher Manning, Stuart Shieber, Dan Flickinger, and Stephan Oepen. 2002. *Parse Disambiguation for a Rich HPSG Grammar*. Technical Report 2002-64. Stanford InfoLab. http://ilpubs.stanford.edu:8090/571/

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[51] Xinyu Wang, Jingxian Huang, and Kewei Tu. 2019. Second-order semantic dependency parsing with end-to-end neural networks. *arXiv preprint arXiv:1906.07880* (2019).

[52] Xinyu Wang, Yixian Liu, Zixia Jia, Chengyue Jiang, and Kewei Tu. 2019. ShanghaiTech at MRP 2019: Sequence-to-Graph Transduction with Second-Order Edge Inference for Cross-Framework Meaning Representation Parsing. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*. Association for Computational Linguistics, Hong Kong, 55–65. https://doi.org/10.18653/v1/K19-2005

[53] Yue Zhang, Wei Jiang, Qingrong Xia, Junjie Cao, Rui Wang, Zhenghua Li, and Min Zhang. 2019. SUDA-Alibaba at MRP 2019: Graph-Based Models with BERT. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*. Association for Computational Linguistics, Hong Kong, 149–157. https://doi.org/10.18653/v1/K19-2014

[54] Jiawei Zhou, Tahira Naseem, Ramón Fernandez Astudillo, Young-Suk Lee, Radu Florian, and Salim Roukos. 2021. Structure-aware Fine-tuning of Sequence-to-sequence Transformers for Transition-based AMR Parsing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 6279–6290. https://doi.org/10.18653/v1/2021.emnlp-main.507

A RISKS AND RISK MANAGEMENT STRATEGIES

Scale is 1 to 5 where 1 represents low impact or probability and 5 represents high impact or probability.

Risk Condition	Consequence	Category	Probability	Impact	Mitigation	Monitoring	Management
Project exceeds allocated time	Final project does not complete all requirements, loss of marks	Time	3	5	Track project progress with project planning software, such as Planner on MS Teams. Enforce a weekly meeting with team members and supervisor.	Use project planning software to monitor progress of project. Creating weekly deadlines for team members.	If necessary, speaking to supervisor to see where the scope can be redefined or narrowed.
Code and/or project data lost due to loadshedding	Delayed progress, less time to complete other features and aspects of the project	Resources	4	4	Enforcing frequent use of various backup services such as GitHub and Google Drive, for all team members.	Setup a system whereby a team member has to notify the team when they have made changes to the existing codebase. Monitor to ensure changes were made successfully.	Prioritise recoding critical features and trying to recover lost data.
Lack of knowledge needed to use frameworks and software	Project may not be completed, with leads to lower marks achieved and supervisor dissatisfaction	Skills	4	4	Discuss with supervisor to determine what technical skills each team member needs. Making it mandatory for members to take online courses or use online resources to upskill.	Project management software can be used to govern when the necessary skills are required. Team members will be required to gain these skills by the stated dates.	If team feels overwhelmed trying to find resources, then liase with supervisor to determine what resources are useful and accessible.
Problems with training models using CHPC cluster (e.g. unexpected downtime) or loadshedding disrupts model training	Time lost as models will need to be retrained, progress on project slows down	Resources	3	5	Testing CHPC cluster to ensure that it is working correctly. Also checking loadshedding schedule to ensure that we do not start training the models near a time when blackouts could potentially happen.	Communicating with the supervisors of the CHPC cluster to ensure that it's in a stable state.	If model training gets interrupted, then training will be restarted when the cluster has low traffic and can thus provide us with more processing power.

Poor Communication between stakeholders and project team	Project will lack a well-defined direction and end deliverable may not meet all requirements	Communications	3	3	Have meetings as often as possible between team and supervisor. Make use of designated communication platform i.e. Slack to ask for clarification when necessary.	Present progress to supervisor at weekly meetings. Ensure that the tasks are being done using the correct methodology.	Keep backups or logs of all communication between team members, as well as between the team and supervisor. Re-evaluate current communication chain and adjust if necessary.
Decrease in Team motivation and morale	Project not being completed because it ran over schedule	Quality, Human Resources	2	3	Set realistic workload and SMART goal setting for team members	Have weekly check-ins with team members to ensure that everyone is not feeling overwhelmed or overburdened.	Taking each member's workload and adjusting when and if necessary.
Misunderstanding the brief and purpose of the project.	Creating models that are not able to correctly perform semantic parsing and thus misrepresent sentence meaning representations.	Scope	2	3	Ensure that the team has an accurate understanding of the project's requirements and scope by having regular meetings with the supervisor. Additionally, making sure the team builds modular code that can easily be re-used should we decide to pursue a different direction with the research.	Elicit feedback from supervisor during and at the end of every iteration of the system design, in order to ensure we have understood and implemented requirements as intended.	Remove the code modules that are not appropriate to the new research goal and reusing those that are still applicable.

B GANTT CHART