# Meaning Representation Parsing

**Literature Review**

**Computer Science Honours Project 2022**

Claudia Greenberg
GRNCLA009
grncla009@myuct.ac.za
University of Cape Town
Cape Town, WC, South Africa

## ABSTRACT

This Honours project literature review presents an analysis of the work that has been developed within the field of meaning representation parsing, with emphasis on semantic graph parsing. This is to assess the viability of existing work and whether there exists a gap for further study. Various subtopics are discussed below which relate to the project's context, such as natural language processing, neural networks and dependency parsing. It is concluded that, whilst there has already been a relatively vast amount of studies on this topic, there is still an opportunity for further studies to be executed.

## KEYWORDS

Natural Language Processing, Neural Networks, Dependency Parsing, Semantic Graph Parsing, Edge Prediction

## 1 INTRODUCTION

Semantic Graph Parsing is the process of generating graphical meaning representation (MR) for natural language, that can be interpreted by machines [10, 26]. This is a crucial aspect of natural language processing (NLP) and has been studied extensively over recent years [7, 16, 33, 37, 41].

There is a growing need for accurate and efficient semantic parsers. They are used within fields such as robotics (understanding commands) and data exploration (understanding varying amounts of data) [26]. The ultimate end goal of these parsers is to correctly and quickly process sentences of different languages without manual (human) processing. Several studies, such as [3, 4, 6, 8–10, 12, 15, 28, 30, 31, 34, 40, 46], have been released, where researchers create, analyse and compare various semantic parsers. Each one intends to use a novel approach or outperform existing, similar models using slight methodological variations.

Our project proposes another attempt at efficiently predicting the correct interpretation of English sentences through the use of neural networks (NNs).

Our project resides in the field of MR Parsing. We will be focusing on a framework called Minimal Recursion Semantics. This field deals with constructing MR graphs for (English, in our case) sentences. Each graph consists of nodes - representing the words directly or more complex entities - and edges - representing the relationships between the nodes.

This literature review attempts to explore the core aspects of the project and analyse several sources related to these aspects. First, a brief overview of the different NNs is discussed, as well as a concise explanation of the related NLP topics. Subsequently, dependency parsing is discussed, detailing the two dependency methods - transition-based and graph-based. Semantic graph parsing is then discussed in detail, particularly focusing on edge prediction. A brief and simple explanation of the meaning representation frameworks and of the performance evaluation metrics are discussed for additional context. Finally, this review concludes with a section on where our project fits into the field of meaning representation parsing.

## 2 NEURAL NETWORKS

As described in [24], a neural network (NN) is a web of computational nodes which work together to compute a complex calculation. Each node takes in some form of input, computes a calculation, and sends its output as input to other nodes. They are constructed using a varying number of layers with a varying number of different nodes, each serving its unique purpose. How these nodes and layers are linked together depends on the type of NN. The calculations are connected using node-specific weights and layer-specific biases. The outputs of these nodes are then often transformed using a non-linear function such as the sigmoid function [42]. The construction and implementation of these NNs allow for parallelism, which greatly increases the speed of the computations [1].

A NN is a type of Machine Learning model. Machine Learning is the process of learning and predicting using historical data. It can be supervised, semi-supervised or unsupervised, whereby supervised learning contains training data to teach the rules [1, 36]. Using machine learning techniques on provided data, the weights of the NN can be learnt.

Semantic graph parsing can use NNs as a tool to train computational models. They have shown to be a primary tool in training parsers to adequately predict natural language sentences' MRs. The array of papers cited in the introduction, such as [10, 28, 46], have shown accuracy scores that outperform pre-existing parsers.

There are various kinds of NNs that can be used depending on the problem's context, as well as variations *within* variations. Three particular types are used extensively in *deep learning* (referring to the many layers of NNs, as expressed in [18]): feedforward, recurrent and transformer. We consider all three but decide on one which is ideal for our project's requirements. Others include convolution NNs, which [49] investigated and concluded to be generally less suitable for NLP than other NN types. Depending on the context, each could be considered the ideal form for the problem at hand. Our project delves into these three as potential candidates for our deep learning strategy.

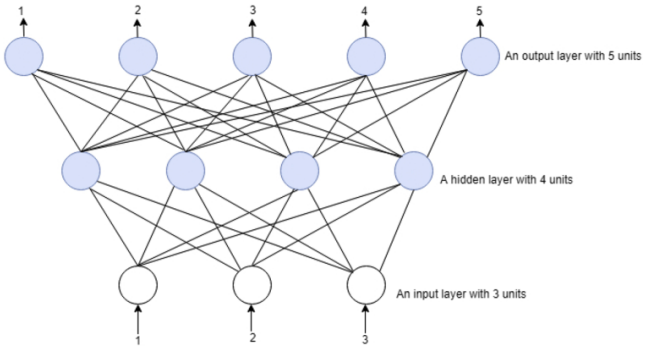


**Figure 1: Two-layered feedforward neural network [1]**

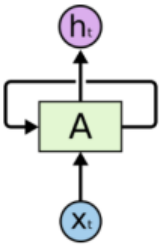


**Figure 2: A Rolled Up Recurrent Neural Network [1]**

## 2.1 Feedforward Neural Networks

A feedforward neural network (FFNN) is a more straightforward type, whereby within each layer of nodes, each node is linked to all other nodes within the adjacent layers. It is originally based on human neuron processing, as explained in [1, 18].

For example, a node in layer 4 will receive, as input, the outputs of all nodes in layer 3 and send its output to all nodes in layer 5. Figure 1 presents an example of a simple FFNN.

FFNNs are a relatively common tool used for NLP. He et al. [20] chose to use an FFN in their model for the factor scores.

According to [24], there are some limitations with FFNNs such as a limited context. This means that if a related word is outside of the context, because it is too far away from the current word, it will not be considered in the decision-making. There is no feedback presented between nodes [1] which means that information is lost during the input-output information transfer. This could also impact the parser's accuracy. Lawrence et al. [32] claim that an FFNN is not the best for learning a grammar, particularly in comparison to an RNN (as discussed in the following subsection). Ultimately, an FFNN could be the best tool for a particular model, and could also be a weaker option in comparison to other approaches.

## 2.2 Recurrent Neural Networks

While FFNNs do not take previous stages of the network into account (i.e., there is no feedback or backtracking), recurrent neural networks (RNNs) do. As defined by [24, 43], an RNN is a network where a node also depends on its own outputs as input. In other words, the network contains cycles. It is able to process sequences of any length. Figures 2 and 3 present a diagrammatic representation of this. Please note that $x_t$ is the input and $h_t$ is the output of the node A.

According to [18], using a recurrent neural model is a strong choice for various NLP tasks [49] such as dependency parsing [4]. Lawrence et al. [32] claim that this type of neural network is a more suitable option for learning a grammar. Tarwani and Edem [43] states that it is more powerful than feedforward. The network's nodes store internal memory. This is another reason why RNNs are often preferred over feedforward for certain semantic graph parsing tasks, as nodes should not be treated as independent units [43]. Many of the papers reviewed for this literature review stated using an RNN (or a sub-type such as bidirectional long short-term memory (LSTM)) in their parsers and they have yielded satisfactory results.

For additional information on the meaning representation frameworks (DM, PSD, EDS, UCCA and AMR), performance evaluation metrics (F1 and Smatch) used, and parsing (dependency and semantic graph), please consult their respective sections within this review.

Buys and Blunsom [4] used a bidirectional RNN for their sentence encoding and decoding. They created a competitive neural transition-based semantic graph parser. They report outperforming the attention-based baselines on minimal recursion semantics as well as the abstract MR benchmark, with their Smatch score being higher than the upper bound.

Wang et al. [47] created a graph-based parser and also presented an RNN for their encoder and decoder. They report higher F1 scores than their competing models, both state-of-the-art and baseline models.

Peng et al. [40] used a bidirectional LSTM network (a recurrent variation) for their basic semantic dependency model. Using F1 scores, they report that their one model - which uses this network - was one of the two best models, outperforming their other models and pre-existing models. Yet, they also report potential design improvements, unrelated to their use of RNNs. They propose the potential for future opportunities and extensions.

Other papers that have stated using some form of RNN include [3, 8, 13, 21].

While this type of neural network sounds like a strong option to consider for our project, the model has its weaknesses. RNNs perform many calculations sequentially, as the nodes are not completely independent of one another. This slows down the process. Another reason is that information is lost during the recurrent process, and this complicates the training [17, 24]. Therefore, the longer the inputted sentence, the lower the accuracy of the model. These issues inspire the next type, which our project has chosen out of these three neural networks.

## 2.3 Transformer Neural Networks

In 2017, Vaswani et al. [45] proposed a new type of neural network to combat the limitations of pre-existing neural networks: the Transformer. This recently-developed neural network can have record-high translation quality and allows for more parallelisation, which is becoming increasingly important with the introduction

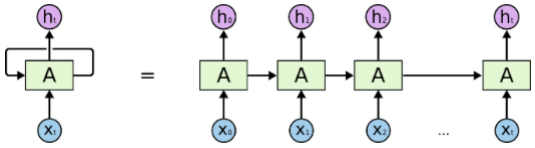


**Figure 3: An Unrolled Recurrent Neural Network [1]**

of multi-core computer architectures. Vaswani et al. [45] reported very positive results, outperforming all other models evaluated, and expressed the potential for future, record-breaking work using this neural network.

Figure 4 depicts the overall structure of the transformer neural network. Each word in a sentence is now able to be processed independently of one another, allowing for all to be inputted simultaneously (rather than sequentially, as with RNNs). Because the model inherently does not take word order into account, positional encoding is combined with the embeddings to add some "context" to the words [45]. The architecture includes three main layers: a self-attention, feedforward and linear layer [24]. A self-attention layer is one that takes in a sequence as input and returns a sequence of the same length. For each word, it is compared to the other words in the sentence that precede that word and a score is calculated where the larger the value, the more compatible it is. For the decoding process, the scores of all words after the current one are automatically set to zero. The linear and softmax transformations during the decoding phase are done to predict the probabilities of the subsequent tokens. Please consult [45] for a full explanation of how the transformer neural network works.

This model was only introduced in 2017, making this a new model. However, it has attracted attention from researchers such as [25, 39]. Desai and Aly [12] evaluate transformers in more detail.

A popular type of transformer, for encoding particularly, is Bidirectional Encoder Representations from Transformers (BERT). Tenney et al. [44] investigate BERT in detail, expressing some concerns that BERT lacks enough interpretability.

Kitaev and Klein [28] proved that using a self-attentive architecture yields benefits over LSTM. This inspired [7] to choose to use a transformer encoder in their model. They present satisfactory performance results, ranking first in AMR, sixth in PSD, seventh in DM and fifth in UCCA.

Zhang et al. [51] created five graph-based parsing systems for the different meaning representation frameworks. They reported using BERT for most of their chosen frameworks. It proved to be beneficial for them. They ranked third overall, notably first in EDS.

Che et al. [8] adopted BERT as well for word representation for their transition-based parsing model. Their model showed a noticeable improvement, using this rather than GloVe. Their model ranked first in a metric called ALL-F1 with a score of 86.20 and they ranked first in UCCA with a score of 81.67.

Ferraro and Suominen [17], a relatively recent paper, proved the benefits of using a Transformer model for semantic graph parsing. They calculated the accuracy of their developed parsing model and compared it to pre-existing models. They outperformed their

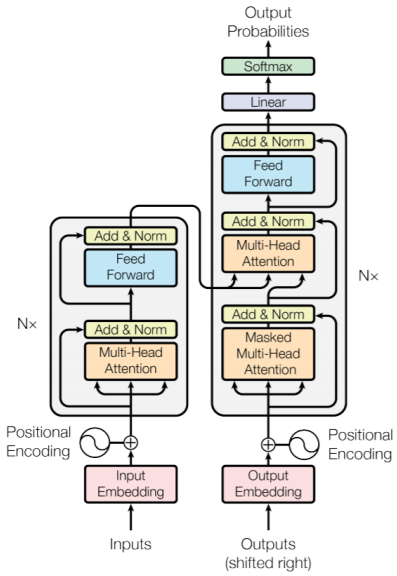


**Figure 4: The Transformer - Model Architecture [45]**

baselines. They note the transformer's powerful potential for learning long-distance relationships, yet also note their specific model's upper limit on detecting these. Finally, they note the potential for using transformers in future work.

Transformer neural networks are currently showing to be a very attractive neural network for semantic graph parsing and other NLP tasks. It combats the RNN sequential limitations and loss of information [17, 24]. This is why using a transformer neural network is the chosen method for this project.

## 3 NATURAL LANGUAGE PROCESSING

Natural language processing (NLP) is a combination of linguistics (language) and artificial intelligence, and was first studied around seventy years ago [36]. NLP is the process of understanding the meaning of, learning, and producing human sentences of any language using various computational techniques [22]. According to [22], four key reasons drive this sector's rapid development: new machine learning developments, increasing computational power, a better understanding of linguistics, and access to large amounts of data. Researchers are constantly striving to develop more within NLP and outperform what already exists. Languages hold complex meanings and interpretations, depending on many aspects such as context, time and place. It is also constantly changing, with new words being added to the dictionary every year. Goldberg [19] explains that humans are not very good at formally defining the languages that we have developed, despite being very good at using and understanding these languages. Various researchers, such as [22], note the complexity and dynamic nature of linguistics, yet also express confidence in the development of NLP within the near future.

NLP has many uses. Many people do not realise how much of the world around us utilise the advances within this field. Virtual assistants on devices - such as Apple's Siri [22] - utilise NLP to

understand and carry out the users' commands. Other uses include translations, summarisation and speech recognition [43].

NLP is the essence of the project being developed, as dependency parsing falls under this. As stated previously, this is a rapidly developing sector which we intend to research as well.

This literature review delves deeper into certain natural language sections to decide whether to include them in our project. First, parts-of-speech tagging is explored as well as word embeddings. Pretrained transformers are then explored.

### 3.1 Part-of-Speech Tagging

To understand a sentence, understanding the words' parts of speech is imperative. Part-of-Speech (POS) Tagging is the assignment of POS tags to each word in a sentence. It allows the model to understand how a particular word fits in the sentence, in relation to the other words. It provides additional context (especially since many words fit into different parts of speech) and allows the model to predict dependencies between words.

POS tagging is a popular way to deal with this, as it often has very high accuracy rates [24]. Each word in the sentence is assigned a POS tag, such as **NOUN**, **VERB** or **AUX**.

In conjunction with POS tagging, one could also adopt named entity recognition to understand proper nouns in more detail. See [24] for more information.

POS Tagging, or simply utilising the tags, is a popular choice in NLP, as stated above.

Peng et al. [40] chose this method to compute a tag sequence.

Cao et al. [6], too, used this in their data-intensive parser to compute an embedding for "richer contextual information". In this paper, they compared knowledge-intensive and data-intensive parsers, which are different approaches to parsing.

Zhang et al. [51] used an embedding of words and POS tags.

Droganova et al. [15] were provided with pre-processed data that included automatic tagging. In this paper, they utilised external parsers for their parsing system.

Other papers include [29, 31, 37].

One shortfall, noted by [24], is that the tagger often runs into unknown words. They are then required to adapt to fit in these words, which can be a challenging task. As stated in the previous section, languages are dynamic entities, often adding new words and popular acronyms.

While POS tagging seems like a popular and successful choice, our project will not be using this. Instead, we will be using a different approach, whereby node label prediction is at the forefront.

### 3.2 Sequence Modelling Taggers

Sequence Modelling Tagging is another important aspect of a semantic parser. These taggers make sure that each node has some kind of label [24]. There are two main kinds to be considered for this project: Hidden Markov Model and Conditional Random Fields.

The Hidden Markov Model (HMM) is a generative approach. It calculates a probability distribution over every combination of labels in a sequence and then using those values, picks the best sequence [24]. It is especially useful for speech recognition [36].

The alternative to the HMM is a discriminative approach called Conditional Random Fields (CRF). This differs from the above as it *discriminates* against certain tags depending on their posteriors $p(Y|X)$. It computes a probability to an output sequence as a whole, as opposed to each label individually [24]. According to [36], they are more appropriate than an HMM for sequential multivariate data. Wang et al. [46] used the CRF approach during edge prediction.

Our project will be using CRF for labelling.

### 3.3 Word Embeddings

Word Embeddings have uses in various fields. Here, we focus on its vital place in NLP. Embeddings allow for words of similar meaning to be placed nearer to each other in a low dimensional space [18, 43]. They are represented in the form of vectors. Using embeddings allow for easier interpretation by the parser. According to [43, 49], using the appropriate embedding procedure can impact the accuracy levels of the parser.

Many models adopt different kinds of embeddings. Two common approaches are *Word2Vec* and *GloVe* [24]. To name a few examples seen in papers, [6] used both character- and word-based embeddings in their parser. He et al. [20] used a pretrained embedding procedure. Bai and Zhao [3], Cao et al. [7] and Lai et al. [31] used pre-trained GloVe.

### 3.4 Pretraining

Pretraining is a technique often used in conjunction with a transformer. The transformer is first trained in a self-supervised fashion using a corpus of text. Once this has been completed, an additional layer is placed (either feedforward or linear) for finetuning [24]. This allows for a more intricate analysis of a language.

Pretraining is a popular technique. Some examples are highlighted below.

Che et al. [8] used a pretrained BERT and then switched to a pretrained GloVe to improve certain conditions for a speed test. They report ranking first in two separate frameworks.

Koreeda et al. [29] used a pretrained GloVe, BERT and ELMo (another word embedding procedure) as well for their MR parsing network (where they integrate all five frameworks). They report mixed results, ranking fifth overall, competing against twelve other submissions.

Zhang et al. [51] used a pretrained word embedding technique for edge prediction as well as a pretrained GloVe. They report effective results, ranking third overall using the F1-score metric system.

On the other hand, Cao et al. [6] expressed concerns that pretrained embeddings may not have a positive impact on semantic parsers. Yet, they use a pretrained ELMo embedding and reported positive results with a 95.05 Smatch accuracy level in EDS.

It has been decided that our project will be using pretraining to improve the accuracy of our parsers. We will first pretrain our transformer. We believe that there will be benefits to using this technique in this project's context.

## 4 DEPENDENCY PARSING

Dependency Parsing, which falls under NLP, is the process of understanding the *syntactic* meaning of a natural language text [2]. Oepen et al. [38] emphasised the use of graph-based rather than tree-based, because of its assumption that every node may be reached from the

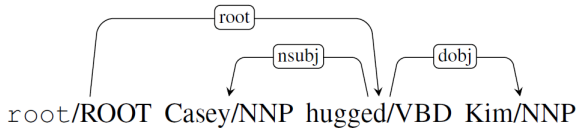


**Figure 5: A Dependency Tree Parse for "*Casey hugged Kim*" [14]**

| Step | Stack | Word List | Action | Relation Added |
|---|---|---|---|---|
| 0 | [root] | [book, me, the, morning, flight] | SHIFT | |
| 1 | [root, book] | [me, the, morning, flight] | SHIFT | |
| 2 | [root, book, me] | [the, morning, flight] | RIGHTARC | (book → me) |
| 3 | [root, book] | [the, morning, flight] | SHIFT | |
| 4 | [root, book, the] | [morning, flight] | SHIFT | |
| 5 | [root, book, the, morning] | [flight] | SHIFT | |
| 6 | [root, book, the, morning, flight] | [] | LEFTARC | (morning ← flight) |
| 7 | [root, book, the, flight] | [] | LEFTARC | (the ← flight) |
| 8 | [root, book, flight] | [] | RIGHTARC | (book → flight) |
| 9 | [root, book] | [] | RIGHTARC | (root → book) |
| 10 | [root] | [] | Done | |

**Figure 6: Trace of Transition-Based Parse [24]**

root node via one directed path. The goal is to correctly discover the words' syntactic relationships.

There are two different dependency parsing approaches, both of which will be studied in this project. The first is transition-based dependency parsing and the second is graph-based dependency parsing.

Figure 5 depicts a simple dependency parse tree. This diagram format is used extensively in dependency parsing. Depicted here is the sentence "*Casey hugged Kim*". A root node has been added, pointing to the word *hugged*. Each word is given a POS tag, where *NNP* is a singular proper noun and *VBD* is a past tense verb. The edges point from the head to the dependent, depicting the relationship between them, where *nsubj* is a nominal subject relationship and *dobj* is a direct object relationship [24].

## 4.1 Transition-Based Dependency Parsing

Transition-Based Dependency Parsing, introduced almost twenty years ago [30], is an approach based on shift-reduce parsing [24]. It consists of a stack, token buffer and an oracle (parser). The tokens get shifted from the buffer onto the stack one by one. At each step, the first two tokens of the stack are examined by the oracle and it decides whether to **LEFTARC** (*second* ← *first* dependency; remove second word), **RIGHTARC** (*second* → *first*; remove first word) or **SHIFT** (push a word from the buffer onto the stack) [4, 24]. It is a greedy algorithm, meaning that the best option at that current moment is taken and the future is not considered [24, 27]. Figure 6 provides a transition-based trace for a better understanding of LEFTARC, RIGHTARC and SHIFT.

To illustrate a few examples: Kiperwasser and Goldberg [27] adopts a transition-based parser for one of their approaches. They reported it to have either outperformed or matched competing models.

Bai and Zhao [3] trained a transition-based parser. They reported an F1 score of only 42%. Compared to their baseline, their model generally had higher precision but lower recall.

Buys and Blunsom [4] proposed a transition-based parser. As stated previously in this review, they reported high-performing results. According to their Smatch scores, they outperformed most of their competing models.

Other transition-based parsers researched during this review include [2, 31, 48].

Our project will include the creation of our own transition-based parser, as transition-based dependency parsing can extend to semantic graph parsing.

## 4.2 Graph-Based Dependency Parsing

According to [24], graph-based dependency parsing is a more accurate approach than transition-based when dealing with longer sentences. The output is a labelled, directed graph [40]. The edges, called arcs, each have a label defining the relationship between a head (parent) and dependent (child) node.

The main difference between the transition- and graph-based is that decisions in graph-based are made based on the entire output structure, as opposed to on individual entities. Another distinction is that graph-based is able to create non-projective structures [40]. A projective tree is one where all arcs have the projective property, whereby the head word may access all words that lie between it and its dependent word [24].

This approach attempts to find the best possible tree for the inputted sentence. How this is decided is by first assigning some calculated score for each arc, depicting its viability, and then finding the maximum spanning tree based on all arc scores [24, 27].

Kiperwasser and Goldberg [27], for example, created a graph-based dependency parser as part of their research on a bidirectional LSTM dependency parsing design. It managed to outperform all but two competing parsers when not using external embeddings.

The techniques for graph-based dependency parsing are similar to the techniques that will be used for edge prediction, as detailed in the subsequent section.

# 5 SEMANTIC GRAPH PARSING

Semantic Graph Parsing is the task of constructing a graph to represent the *semantic* meaning of natural language text. These graphs are constructed in a way that is easily understood by machines. This differs from dependency parsing, which is syntactic by nature. They have overlapping concepts and techniques, yet ultimately complete different tasks.

Du et al. [16] investigated how effective using syntactic parsing methods would be for semantic graph parsing. Given appropriate context, there is potential for the dependency parsing methods, as described in the previous section, to be used effectively for semantic graph parsing.

According to Johansson and Nugues [23], semantic role labelling is the task of learning "who does what to whom". It is a specific kind of semantic graph parsing that may be the most appropriate, depending on the problem's context.

Zhang et al. [51] created graph-based semantic parsers. As previously stated, they reported ranking third overall in their performance test.

Cao et al. [6] created a graph-based parser. This parser produced a high Smatch score of 95.5 for the EDS framework for MR.

Peking et al. [9] created a semantic parser using a factorisation-based approach, which reported F1 scores of over 95 for most evaluation sections.

For this project, two semantic parsers will be implemented, one transition-based and one graph-based. The latter of the two will be produced in a pipeline fashion with node prediction being separated from edge prediction.

## 5.1 Node Prediction

Predicting the nodes of the graph is the first step of the pipeline. The input is a sequence of words and the output is a sequence of nodes. The nodes represent the words of the sentence, or some indirect representation of the words, such as the **ROOT** node. There may be multiple nodes relating to one word, or vice versa. The model needs to be able to predict the correct sequence of nodes that map to an inputted sentence with a high level of *accuracy* and *speed* [24].

On top of predicting the nodes, they must also be labelled. Sometimes, it may be more appropriate, given the context of the words, to predict and label spans (i.e., multiple words) rather than singular words. This is another important task of node prediction.

There are various ways to predict the nodes and labels, depending on the context and parsing approach. For example, Zhang et al. [51] used a multi-head self-attention model for their node prediction. Some errors were noted during their analysis and these errors rolled over to affect the edge prediction performance. Using F1 scores, they reported ranking highly - notably third overall.

Node prediction may be treated as a labelled span prediction problem or, more specifically, a tagging problem. For this project, we will be undertaking this using a transformer encoder.

## 5.2 Edge Prediction

After node prediction, edge prediction is the second part of the pipeline. As previously explained, edge prediction can pull similar techniques from graph-based dependency parsing. It receives, as input, the appropriate nodes and labels for the graph. From there, the model must predict potential edges (arcs) using a neural network. These edges are then scored using a particular calculation. From there, the highest-scoring graph is chosen strategically (i.e., not necessarily in a greedy fashion), usually by finding the maximum spanning tree (more on this below). This means that the chosen edges form the best-scoring MR graph for the inputted text, even if some of the highest-scoring labels are not chosen because they do not fit in with the optimal combination [24].

Figure 7 depicts a semantic graph for the sentence "*Book that flight*". An additional root has been set in place to pinpoint the root node (in some cases, it is the main verb of the sentence). An algorithm is run to calculate which combination of edges yields the maximum spanning tree. In this case, the graph contains cycles and these can be eliminated using some other technique, such as [11] (explained below). If one were to use the greedy approach to choose the maximum spanning tree, the edge from *that* to *flight* would be chosen because it has a score of **8**. However, this restricts the other appropriate edges. What ends up being the most appropriate is the path highlighted in purple.

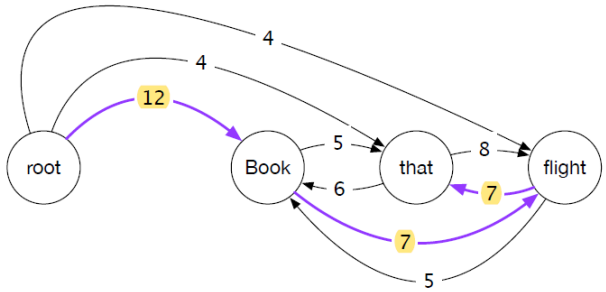


**Figure 7: A directed graph for "*Book that flight*" [24]**

One example of an algorithm for scoring these edges is a biaffine network [14]. To calculate this, the embedded nodes are first sent through two FFNNs. The first network creates a representation for the head of the edge and the second, the dependent. These representations are:

$$\mathbf{h}_i^{head} = \text{FFNN}^{head}(\mathbf{r}_i) \tag{1}$$

$$\mathbf{h}_i^{dep} = \text{FFNN}^{dep}(\mathbf{r}_i) \tag{2}$$

These representations are then sent to a biaffine scoring function that calculates the edge's score.

$$\text{Score}(i \rightarrow j) = \text{Biaff}(\mathbf{h}_i^{head}, \mathbf{h}_j^{dep}) \tag{3}$$

where

$$\text{Biaff}(\mathbf{x}, \mathbf{y}) = \mathbf{x}^{\text{T}}\mathbf{U}\mathbf{y} + \mathbf{W}(\mathbf{x} \bigoplus \mathbf{y}) + b \tag{4}$$

where U, W and b are learned weights.

Passing this onto a softmax function calculates a probability between 0 and 1. All equations above are from [24].

Once the individual scores have been calculated, the best combination of edges must be chosen to find the optimal (highest-scoring) graph. As stated above, this is often done by finding the maximum spanning tree, which is a tree within the graph that encompasses all of the nodes in that graph, and the edges within the tree yield the highest score when combining the edges [24].

A common, potentially-recursive algorithm to find the maximum spanning tree for a weighted directed graph is the Chu-Li Edmonds algorithm [24], as shown in Figure 8. First, the algorithm uses the greedy approach by finding the highest-scoring incoming edge for each node in the graph. If the combination of those edges is a spanning tree, this is the maximum spanning tree. If not, a cycle is found and eliminated by contracting the graph, finding the maximum spanning tree of that new graph and then expanding it back to the original form with the deleted edge that deletes the cycle. This is the recursive step of the algorithm. From there, the maximum spanning tree can be found in this newly-acyclic graph.

Edge prediction is an imperative step in the semantic graph parsing process. Different parsers use different techniques.

Koreeda et al. [29] used a biaffine network to predict their node attributes and labelled edges. They reported positive results, ranking fifth against their competing parsers, and they outperformed their baseline parser.

```
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 8: The Chu-Li Edmonds Algorithm for Finding the Maximum Spanning Tree [24]**

Na et al. [35] also chose to use a biaffine network for their parsing model and investigated biaffine models and methods in detail. They do not present state-of-the-art performance results, outperforming baseline models for the DM and PSD frameworks but ranking relatively low for all frameworks evaluated (DM, PSD and UCCA).

Highlighting one parser in particular, used for SDP (DM and PSD), [51] was inspired by [14]'s biaffine network. The words are transformed into a vector which is sent to an LSTM for contextualised representations. Two modules are used, one to predict whether a particular edge exists and one which predicts the most suitable label for the edge. Using, overall F1 scores, they ranked eighth for DM (with a score of 91.64) and ninth for PSD (with a score of 81.84).

Wang et al. [47] used the method used by [46], which follows from [14]'s biaffine network for scoring. Wang et al.'s [46], however, differs slightly. Scores are done for first- and second-order parts and are then sent through recurrent inference layers. For context, first-order parts are the usual individual edges and second-order parts deal with relationships between more than two nodes (see [46] for more information). For edges specifically, [47] received the following F1 scores and ranks for each framework: 92.32 for DM (ranking first) and 79.50 for PSD (first).

Our project will be utilising the biaffine network for scoring predicted edges. We will be finding the maximum spanning tree to determine the most appropriate graph.

## 6 MEANING REPRESENTATION FRAMEWORKS

Oepen et al. [37] describe the five main frameworks (consisting of different expectations and assumptions) in detail for MR parsing, which are briefly described below. Each framework requires its own specific structure and has unique linguistic properties for the nodes, labels (tags) and edges[3].

DELPH-IN MRS Bi-Lexical Dependencies (DM) is the first framework. It was developed for the Semantic Dependency Parsing tasks in 2014 and 2015 at the SemEval campaigns. Its nodes (tokens) are only straight-forward, directly representing the words, and there is no designated root node.

Prague Semantic Dependencies (PSD) was also developed for SDP. Many papers, such as [51], refer to PSD and DM under SDP. PSD shares similarities with DM, also only utilising straight-forward nodes. However, its labelling system and dependency structure are noticeably distinct.

The third framework is Elementary Dependency Structures (EDS). It is a variable-free semantic graph and is based on DM. Its nodes are not necessarily straight-forward, i.e. it may mean more complex entities than just the word itself, with additional information attached.

The Universal Conceptual Cognitive Annotation (UCCA) framework is one that is tree-like (although need not have a root node) and contains unlabelled nodes, which correspond to the words and grammatical units (such as commas) in the sentence. There may be different types of edges and the edges are labelled with singular letters.

The final framework reviewed is Abstract Meaning Representation (AMR). The nodes contain more abstract entities, rather than a straight-forward relation to the words of the sentence. They may use inverted edges, mainly for tree-viewing convenience.

For all five frameworks, nodes may have more than one incoming edge.

Our project will only be using EDS.

## 7 EVALUATION

Parsers need to undergo performance evaluations to fully understand their usefulness and competitiveness. Metrics may be run on the entire parser and on the individual sections such as labels, nodes and edges. These metrics aim to understand how close a parser's resulting sentence representation is, to what was expected. Two metrics are discussed below in detail.

F1, precision and recall is a very common [50] accuracy scoring metric combination and is used in many papers, such as [4, 20, 28]. Precision is the ratio of **correct** to **total** labels. Recall is the ratio of **correct** to **total expected** labels. F1 is the "harmonic mean" of the previous two [24]. Whilst these calculations are centred around assessing labels (tags), they can be adapted to evaluate the nodes and edges of parsers. Parsers are fed natural language sentences and the resulting meaning representations are compared to what was expected and correct.

A slight variation to the above is Smatch, introduced by [5], which calculates overlap between different "feature structures". This has been used by papers such as [4, 6, 9].

Another evaluation metric pair, that could be used for dependency parsing only, is unlabelled attachment score and labelled attachment score. [27] has used this.

One final evaluation strategy is exact match. This marks an entire sentence as right or wrong. Within many contexts, this is an unsuitable option if used as the sole metric. This is because a more precise measure must be used to understand *how incorrect* a

sentence is and whether a future iteration of the parser is predicting better [24].

We will be using F1 scores to evaluate our parsers' nodes, labels and edges.

## 8 MOTIVATION

This Honours project is centred around producing two new semantic parsers - one transition-based and one graph-based. For the latter approach, we are separating the node and edge prediction tasks rather than combining them, as in [4, 10, 20].

The goal is to perform a careful analysis of the effect of our algorithm design choices, such as which of our parsers are faster and more accurate, according to some quantitative measure, and whether using a pretrained transformer (which is a relatively novel approach [45]) has a positive effect on the performance results.

## 9 CONCLUSIONS

Semantic graph parsing, within the field of meaning representation parsing, is an extensively-studied and relatively new field. There have been various attempts at predicting the meaning of natural language sentences with 100% accuracy. As proven in previous sections, many have reported creating high-scoring parsers. There are different ways of representing the meaning of sentences and different techniques to get there, depending on the languages covered. This is a dynamic field, with new techniques and state-of-the-art parsers still being developed. We conclude that there have already been extensive studies within the field, yet there is still future work to be explored and higher-performing parsers to be developed. This project aims at contributing to the field with two new semantic graph parsers, whose performances will be evaluated against existing baseline and state-of-the-art parsers.

## REFERENCES


[1] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. 2018. State-of-the-art in artificial neural network applications: A survey. *Heliyon* 4, 11 (2018), e00938.

[2] Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. *arXiv preprint arXiv:1603.06042* (2016).

[3] 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

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

[5] 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.

[6] 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.

[7] 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

[8] 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

[9] 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.

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

[11] Yoeng-Jin Chu. 1965. On the shortest arborescence of a directed graph. *Scientia Sinica* 14 (1965), 1396–1400.

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

[13] Lucia Donatelli, Meaghan Fowlie, Jonas Groschwitz, Alexander Koller, Matthias Lindemann, Mario Mina, and Pia Weißenhorn. 2019. Saarland at MRP 2019: Compositional parsing across all graphbanks. 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, 66–75. https://doi.org/10.18653/v1/K19-2006

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

[15] 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

[16] 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.

[17] Gabriela Ferraro and Hanna Suominen. 2020. Transformer semantic parsing. In *Proceedings of the The 18th Annual Workshop of the Australasian Language Technology Association*. 121–126.

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

[19] Yoav Goldberg. 2017. Neural network methods for natural language processing. *Synthesis lectures on human language technologies* 10, 1 (2017), 1–309.

[20] 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).

[21] Daniel Hershcovich and Ofir Arviv. 2019. TUPA at MRP 2019: A Multi-Task Baseline System. 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, 28–39. https://doi.org/10.18653/v1/K19-2002

[22] Julia Hirschberg and Christopher D Manning. 2015. Advances in natural language processing. *Science* 349, 6245 (2015), 261–266.

[23] Richard Johansson and Pierre Nugues. 2008. Dependency-based semantic role labeling of PropBank. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. 69–78.

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

[25] Aditya Kalyanpur, Or Biran, Tom Breloff, Jennifer Chu-Carroll, Ariel Diertani, Owen Rambow, and Mark Sammons. 2020. Open-domain frame semantic parsing using transformers. *arXiv preprint arXiv:2010.10998* (2020).

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

[27] Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics* 4 (2016), 313–327.

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

[29] 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

[30] 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).

[31] 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

[32] Steve Lawrence, C Lee Giles, and Sandiway Fong. 1998. *On the applicability of neural network and machine learning methodologies to natural language processing.*

Technical Report.

[33] 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.

[34] 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.

[35] 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

[36] 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.

[37] 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

[38] Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Yi Zhang. 2014. SemEval 2014 Task 8: Broad-Coverage Semantic Dependency Parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Association for Computational Linguistics, Dublin, Ireland, 63–72. https://doi.org/10.3115/v1/S14-2008

[39] 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*. 40–52.

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

[41] 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.

[42] Milan Straka and Jana Straková. 2019. ÚFAL MRPipe at MRP 2019: UDPipe Goes Semantic in the Meaning Representation Parsing Shared Task. 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, 127–137. https://doi.org/10.18653/v1/K19-2012

[43] Kanchan M Tarwani and Swathi Edem. 2017. Survey on recurrent neural network in natural language processing. *Int. J. Eng. Trends Technol* 48 (2017), 301–304.

[44] Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. *arXiv preprint arXiv:1905.05950* (2019).

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[46] 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).

[47] 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

[48] David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. *arXiv preprint arXiv:1506.06158* (2015).

[49] Jiudong Yang and Jianping Li. 2017. Application of deep convolution neural network. In *2017 14th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*. IEEE, 229–232.

[50] Dell Zhang, Jun Wang, and Xiaoxue Zhao. 2015. Estimating the uncertainty of average F1 scores. In *Proceedings of the 2015 International conference on the theory of information retrieval*. 317–320.

[51] 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