Meaning Representation Parsing

A Literature Review

Jane Imrie
IMRJAN001@myuct.ac.za
University of Cape Town

ABSTRACT

This literature review aims to investigate the feasibility of using neural networks for node and edge prediction in semantic graph parsing, with a focus on concept identification. The current machine learning paradigms and their applicability to predicting concepts for spans was examined. Dependency parsing, and the viability of extending its algorithms to semantic graph parsing was also explored. There is evidence to suggest the pretrained encoder models, such as BERT, can be used for graph-based parsing, but further research is required.

KEYWORDS

Semantic Graph Parsing, Natural Language Processing, Semantic Parsing

1 INTRODUCTION

Semantic parsing is a Natural Language Processing (NLP) task that can briefly be described as the process of taking text and parsing it into a machine interpretable structure [35]. Semantic graph parsing is thus a kind of semantic parsing, where the text is interpreted into a graph structure. It uses ideas and concepts from dependency parsing in order to generate and analyse these structures. Many parsers have been developed in order to perform this task, with the aim of maximising accuracy, efficiency and outperforming other state-of-the art models. [19] [23] [5] [43]. Semantic graph parsing, like other NLP tasks - such as syntactic parsing or sentiment analysis - makes use of various different kinds of machine learning (ML) architectures [16]. Many of the previously mentioned parsers use LSTMs, however, the newer transformer architecture (created in 2017) is slowly starting to see use in these models, particularly as encoders in the form of BERT. However, current research on this is limited.

This paper aims to provide a brief introduction to some of the ML models that have been used in the field of NLP - such as recurrent neural networks and transformers; take a shallow, subject-constrained dive into some more general NLP concepts such as word embeddings and part-of-speech tagging, provide a somewhat detailed explanation of dependency parsing (transition and graph-baed) and finally elaborate on semantic parsing, and the relevant concepts that apply to this project.

2 MACHINE LEARNING

2.1 Word Embeddings

Word embeddings refers to the process of representing words in a format that allows them to be easily analysed by computers. Most commonly words are represented as vectors, a form which enables them to be used with machine learning models and algorithms [1]. Given a word, w, from some vocabulary V and vector $\vec{w}$ that has a real number value $\mathbb{R}$, we can say that mathematically, a word embedding is a mapping: $V \rightarrow \mathbb{R}^D : w \mapsto \vec{w}$, in an embedding space that has dimensionality D [47]. Two of the more well-known and reportedly most efficient models for word embeddings are word2vec and GloVe (Global Vectors) [36].

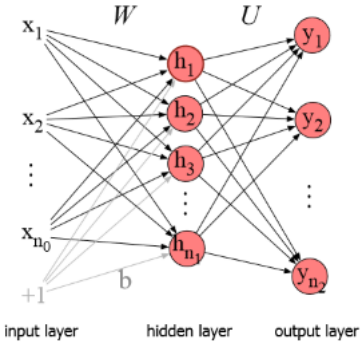


Figure 1: Simple feed-forward neural network with one hidden layer

2.2 Neural Networks

A neural network (NN) can be thought of as a learning model. Modelled after neurons in the human brain, these networks consist of small computing units. Each unit requires a vector as an input, and outputs a single value [24]. The use of neural networks have become quite prevalent in the field of Natural Language processing, and have been used for a wide variety of tasks within the field. [12][43][3] [5] show that NN's can be used for semantic parsing, the focus of this review. [29] [23] show that NNs can also be used for other kinds of NLP tasks, such as POS tagging and NER.

2.3 Feed-Forward Neural Networks

A feed-forward neural network consists of an input layer, at least one hidden layer and finally an output layer. In figure 1, W represents a matrix of weights from the input layer to the hidden layer, and U represents a weight matrix from the hidden to the output layer. Quite simply put, each neuron multiplies each input by its weight and then sums them. This result then has a non-linear activation function applied to it, which gets applied to the next layer. This process continues until the output layer is reached. Mathematically, the simplest neural net, the perceptron, can be represented in terms of vector-matrix operations:

$$NN_{perceptron}(x) = xW + b \tag{1}$$

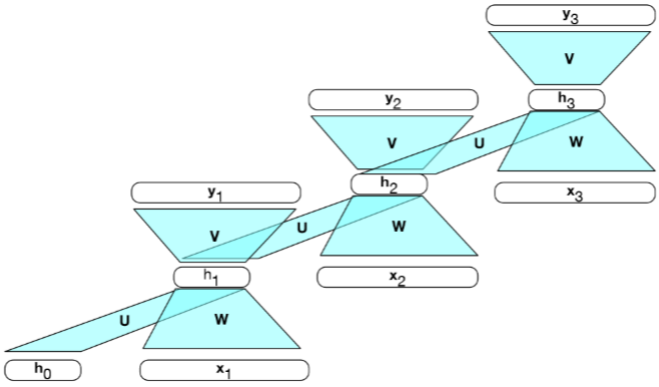


Figure 2: A simple RNN, unfolded over time [24]

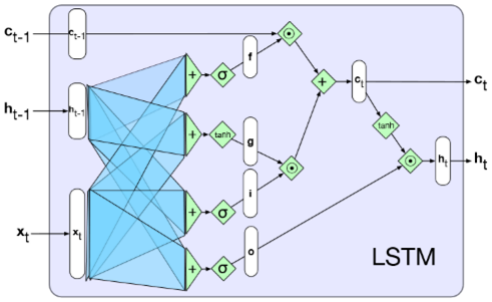


Figure 3: A single LSTM unit depicted graphically [24]

Where b represents a bias term and W the standard weight matrix. A Multi Layer Perceptron (MLP) can be created by adding a non-linear hidden layer. Following on from the previous equation, the MLP1 with one hidden layer is presented in this way:

$$NN_{MLP1}(x) = g(xW^1 + b^1)W^2 + b^2 \tag{2}$$

The first linear transformation has W^1 and b^1 stand for a matric and bias term, with W^2 and b^2 having the same function for the second. g is the non-linear activation function. Common non-linear functions for g are sigmoid, hyperbolic tangent (TanH) and rectified linear units (ReLu). Frequently, the output layers gets transformed as well. One of the most popular transformations is the softmax function:

$$x = x_1, \ldots, x_k$$
$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}} \tag{3}$$

[16]

2.4 Recurrent Neural Networks

While feedforward neural networks can be used for NLP, it's greatest limitation is that is uses limited context. A feedforward neural network has a context window and anything that falls outside of that window how no impact on the calculations it runs and choices it makes. However, some NLP tasks may require one to use information that is indiscriminately distant from the current word – and feedforward NN's have no mechanism to achieve this. This is where recurrent neural networks (RNNs) can be utilised. Though there are different kinds of RNNs, this paper will focus on Elman networks (otherwise known as simple RNNs). For each time step, they record an internal state, made possible by the looped connections between lower and higher layer neurons. In short, earlier outputs form part of the input to a unit. The hidden layer at time $t + 1$ and output at time t are both influenced by the hidden layer at time t. RNNs need to be trained differently than feed-forward neural networks because of this added temporal dimension. Backpropagation though time is one of these learning algorithms, and it involves unfolding the RNN through time, which results in the construction of a feedforward NN – shown in figure 2. The weight matrices U, V and W are shared across all time steps, but at each time step the network layers must be recalculated. Despite its advantages over FFNN's, the simple RNN suffers from the problem of vanishing gradients – a phenomenon whereby the gradient becomes lower as progress is made through the network, which increases the difficulty of training the weights [48]. Practically, it has also proven difficult to train RNNS for tasks where the network has to make use of information that is very distant from the current point being processed. The hidden states usually end up with more local encoded information that has higher relevance to the latest parts of the input sequence – notwithstanding the fact that the network has access to the whole preceding sequence [49]. This is where LSTMs enter the picture.

2.5 Long Short Term Memory Neural Networks

Long Short-Term Memory neural network is an extension of the RNN. While simple RNN's cannot juggle both the task of giving the current decision useful information and "remembering" information which must be carried forward for future decisions, this context management problem is something the LSTM can handle. It does so by splitting the over-arching problem into two sub-parts: taking the current context and removing unnecessary information, as well as retaining information which has a high probability of being needed for later decision making. Without going into too much detail, the LSTM accomplishes this through the use of gates – neural units which simulate logical gates. These gates control information flow between the network layers. Moreover, the LSTM modifies the RNN architecture through the addition of an explicit context layer [48]. The 3 shows a single LSTM unit – where x is the current input, h_{t-1} is the hidden state for the previous layer and c_{t-1} is the previous context. These three components form the input to this unit, which outputs h_t and c_t are a new hidden state and updated context, respectively.LSTMS have proven to be a popular choice for NLP tasks, and has been used by [19][12][43][51][23] - a more detailed explanation of the specific use cases for this model is explained in later secions of this paper.

2.6 Transformers

A transformer is a kind of neural network that has become widespread for use in various kinds of NLP tasks [52]. The transformer model has eclipsed the previously popular recurrent and long-short term memory neural networks, due to the fact that it takes less

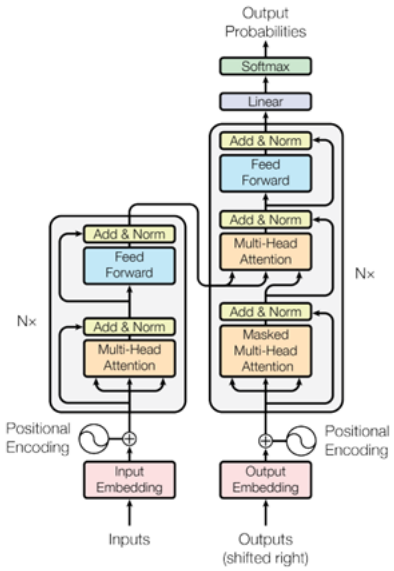


Figure 4: The Transformer Architecture [50]

time to train than these models whilst also more effectively parallelizing the learning process. The transformer has an underlying encoder-decoder structure, where an input sequence of symbol representations $(x_1, .., x_n)$ is mapped by the encoder to z, a sequence of continuous representations $(z_1, \ldots, z_n)$. The decoder generates an output sequence of symbols, $(y_1, \ldots, y_n)$, one element at a time, given z as input. When generating the next sequence, the model consumes additional input in the form of the previously generated symbols. Despite this structure, the transformer can be used for just encoding or decoding. Figure 4 shows the general architecture of the transformer model [50]. The transformer model consists of transformer blocks. These blocks in turn, are composed of a self-attention layer, feedforward layers, normalising layers and residual connections. The residual connection allows information to be passed from a lower to higher layer without going through the intermediate layer. Positional embeddings are used to model the position of each token in the input sentence. These are necessary because transformers do not consider the position of the tokens that are input. For this very reason, all tokens from the input can be submitted simultaneously, which is why the model's learning can be parallelised so easily.

What differentiates the transformer from the other kinds of networks is this self-attention mechanism. One of the key concepts of the attention-based approach is that, given a collection of items and some item of interest, these two things can be compared, such that their relevance in the current context is revealed. The current input uses the result of these comparisons to create output. This concept can be applied to self-attention, where, given a particular input sequence, comparisons are made between the elements of that sequence. Comparing elements in a self-attention layer can be done with a dot-product - a softmax is then used to normalise these scores. This results in a vector of weights which specifies proportional relevance of each of the inputs to the current component that is the focus of the attention. This mechanism allows the model to capture dependencies between words at any distance, with great efficiency [50]. The self-attention takes input in the form of matrices that represent queries, keys of dimension d_k and values, that are mapped to their representations Q, K and V. The attention is then computed on these representations:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_K}})V \tag{4}$$

[29]

The transformer can be used to successfully perform, with a high degree of accuracy, POS tagging [29] and Named Entity Recognition [52]. BERT (Bidirectional Encoder Representations from Transformers) is a relatively recent language representation model, which has become quite prevalent in the literature. A deep bidirectional transformer is trained by being given unlabeled text and using both the left and right context in all layers [11]. A pre-trained BERT model can be fine tuned through the addition of an output layer to the architecture – and this allows for a highly accurate model which can be applied to a diverse number of natural language problems [26]. [18] has shown that for part-of-speech tagging tasks, as well as syntactic and semantic parsing, BERT can be successfully used for token-level embeddings.

3 NATURAL LANGUAGE PROCESSING

Natural language Processing can be considered to be a field under the discipline of artificial intelligence. It utilizes numerous computational techniques in order to perform linguistic analysis on different kinds of texts. The over-arching goal is to achieve human-like language processing over a variety of tasks and applications [31].

3.1 Part of Speech Tagging and Named Entity Recognition

Natural Language processing have various tasks which are used to analyse aspects of human language. A subset of these tasks are semantic and syntactic parsing of various texts. Part-of-speech tagging and Named Entity Recognition are two such tools which aid these parsing processes [23].

3.1.1 Part of Speech (POS) Tagging. For many languages, we use the part of speech (POS) system to categorise words in a sentence. For example, English would generally label "smoking" as a verb, though this can change depending on the context and other words in the sentence. For example – "he was smoking outside" vs "that's a no-smoking sign". Part of speech tagging is thus the process of assigning labels to words, based on their contextual information [46]. Furthermore, there are a variety of schemes used for tagging, such as Universal POS and Penn Treebank. The following example sentence has been annotated using the UD tagset.

Example: There/PRO/ are/VERB/ 2/NUM/ children/NOUN/ there/ABV/./PUNC/ .

3.1.2 Named Entity Recognition (NER). Proper nouns, like "Cape Town" or "Takealot", are semantically viewed as different kinds of entities. Cape Town is a geographical location (LOC) and Takealot is an organisation (ORG). We refer to these generally as named

entities, with there being different labels for the different kinds. Named entities are not strictly proper nouns and include other attributes like currency, dates, times – and these can be extended to incorporate field-specific types like proteins, chemicals, etc. Named entity recognition (NER) can therefore be defined as the process of identifying spans of text that can be classified as named entities, and then tagging them [24]. As with POS tagging, there are also tagging methods for entities, but one of the main ones is BIO tagging [45]. BIO tagging does NER by treating it like as a word-by-word sequence labelling tasks, using tags that capture the boundaries of the span, as well as it's type. A token that begins with the span of interest is labelled with a "B". Any tokens that occur inside the span are labelled with an "I", and any tokens outside the span are labelled with "O".

Example: Cyril/B-PER/ Ramaphosa/I-PER/ is/O/ the/O/ president/O/ of/O/ South/B-GEO/ Africa/I-GEO/.

3.2 Sequence Labelling

Both NER and POS tagging fall under the category of "sequence labelling". There are a number of sequence labelling algorithms available, but two of the most popular are the Hidden Markov Model (HMM) and Conditional Random Fields (CRFs). HMM is based on Markov chains and assumes that a word is conditioned on its tag. It aims to maximise the joint probability of a paired observation and a label sequence [44]. For example: what is the probability of "run" coming after the word "will" i.e. $P(verb\ in\ base\ form|modal\ verb)$, and what is the probability of "will" being a modal verb i.e.

$P(will|modal\ verb)$ [24]. HMMS, through their Markov chains for POS tags, are able to (in quite a limited sense) indirectly consider the context of a word. However, CRFS are far more flexible are how it can use a word's context. But,frequently the context of a word can play a significant role in what it's correct label is i.e. knowing a lot about the preceding or following words for the one under consideration is a feature that would be needed to achieve high accuracy for tagging tasks [24].Because of their flexibility, CRFS have been shown to outperform HMMs for POS tagging [28].

Given a sequence of input words $(x_1, \ldots, x_n)$, labelled X – the aim is to compute a sequence of output tags $(y_1, \ldots, y_n)$, labelled Y. HMMs use Bayes' rule and the likelihood $P(X|Y)$ in order to compute the tag sequence that maximises $P(Y|X)$. The following equation shows how Bayes' rule is used for POS tagging - given $w_1 \ldots w_n$, i.e. a sequence of n words, the aim is to find the most probable tag sequence $t_1 \ldots t_n$:

$$\hat{t_{1:n}} =_{t_1 \ldots t_n} \frac{P(w_1 \ldots w_n|t_1 \ldots t_n)(t_1 \ldots t_n)}{P(w_1 \ldots w_n)} \tag{5}$$

Conversely, CRFS compute the posterior $p(Y|X)$ directly.The CRF computes log-linear functions at each time step, over a set of applicable features. These are known as local features, and will be aggregated and normalized, resulting in a global probability for the whole sequence. It assigns a probability to Y, out of $\mathscr{Y}$ (all possible output sequences), given X. A function F maps the entirety of X and Y to a feature vector. Assuming K features, with each feature labelled F_k, and being assigned a weight w_k, $p(Y|X)$ can be computed in the following way [24]:

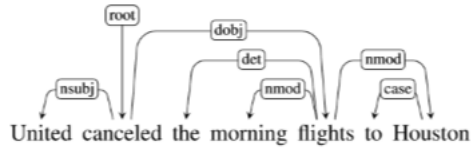


Figure 5: Dependency relations for an English sentence [24]

$$p(Y|X) = \frac{exp(\sum_{k=1}^{K} w_k F_k(X, Y))}{\sum_{Y' \in \mathscr{Y}} exp(\sum_{k=1}^{K} w_k F_k(X, Y'))} \tag{6}$$

As with HMMS, we can use the Viterbi algorithm for inference and training the CRF [24]. The Viterbi algorithm uses the principles of dynamic programming to find the optimal sequence of tags. [32] and [24] provides a more in-depth explanation for implementing the algorithm, as well as some of the mathematics underpinning it.

3.2.1 Evaluation of NER. Taggers for NER can be evaluated using the metrics of precision, recall and the F1 measure, which is the harmonic mean between precision and recall [17].

$$\text{Recall} = \frac{\text{number of correctly labelled items}}{\text{total number of items that should have been labelled}} \tag{7}$$

$$\text{Precision} = \frac{\text{number of correctly labelled items}}{\text{total number of correctly labelled items}} \tag{8}$$

Finally, the F1 score can be computed as

$$\text{F}_1 = \frac{(2 * \text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})} \tag{9}$$

4 DEPENDENCY PARSING

Dependency parsing is a type of syntactic parsing based on the concept of the dependency grammar. There are four main approaches to dependency parsing: a transition based, a graph based, constraint dependency parsing and context-free dependency parsing [39]. The transition and graph based approaches are covered in later sections.

4.1 Dependency relations

One of the fundamental ideas that underpins the dependency grammar is the notion of a dependency relation, shown by figure 5. A dependency relation consists of a dependant and a head – the word upon which it depends. The syntactic structure is thus comprised of words linked by these asymmetric, binary relations [39]. These relations are drawn as labelled arcs from head to dependent. The labels represent the grammatical function that the dependent has in relation to it's head e.g. passive subject or direct object [24].

A dependency-based approach to parsing has many advantages: it abstracts word order information away – the only information that is kept is the information that's necessary for the parse. Secondly, for semantic parsing, the semantic relationship between arguments and predicates can be approximated by the dependency relation – which allows for the dependency parsing approach to be useful for tasks such as information extraction and question

answering [24]. For syntactic parsing, it is able to efficiently extract low-level relationships between words in an input sequence [12].

4.2 Dependency structure

A directed, acyclic graph can be used to embody this structure. The edges/arcs represent the dependency relation, and the nodes represent the words in a given sentence [37]. There are certain constraints, which should be noted:

(1) There is only one root node and it must not have incoming arcs.
(2) Each node has exactly one incoming arc, except for the root.
(3) There is a unique path from the root to each node in the graph.

A directed graph that satisfies these constraints is known as a dependency tree, which is the structure that is compatible with the graph-based and transition-based approaches [24]. This restriction to a tree structure only applies to syntactic analysis, and therefore a graph construction can be used for semantic analysis [12].

4.3 Transition Based Parsing

Transition-based parsing draws from the concept of shift-reduce parsing, a technique used in compilers. There is a buffer of words which need to be parsed, a stack on which the parse is built [39], a predictor, called an oracle and finally, a parser which uses the oracle. Supervised training is used for the parser, and the oracle generates the transition sequence used in training [40]. The input for the parser is some sequence of words, which it operates on from left to right. Items are shifted from the buffer onto the stack – and the top two elements are selected. The oracle is then consulted about what the optimal transition is to apply to these two items. While there are a number of different transition systems, here are three possible transitions for one of these systems[38]:

(1) Left arc: the word at the top of the stack is assigned as the head and the second word is assigned as the dependent. The dependent gets removed from the stack
(2) Right arc: the word at the top of the stack is assigned as the "dependent" and the second word is assigned as the head. Remove the dependent from the stack.
(3) Shift: remove the front word from the buffer and place it on the stack

Left and right arc can be thought of as reduce operations. Training the parser involves creating a training set and having the oracle predict an optimal transition sequence for each dependency tree. The oracle can be approximated by using a neural network that acts as a classifier [40].

Beam search and global learning have been applied to this arc standard approach to transition-based parsing, leading to performance that is comparable to graph-based methods [53].

4.4 Graph-Based Parsing

Graph-based parsing involves finding a global optimum – that is, the highest scoring spanning tree, from a complete graph. This stands in stark contrast to transition-based parsing, which relies on a greedy algorithm that tries to find the local optimum i.e. highest scoring transition [7]. Given a sentence S, we can construct a graph

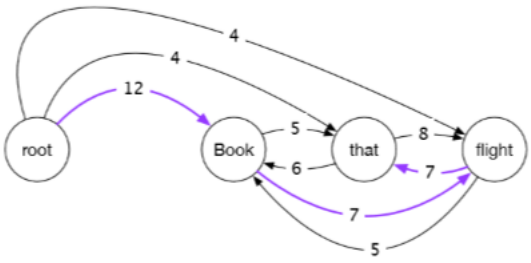


Figure 6: Directed graph for English sentence - maximum spanning tree labelled with purple [24]

G by using the words from S as input and the edges represent all possible head-dependent relations. This would thus be a weighted, directed graph, with a root node inserted that has an edge to every other node in the graph. The spanning tree of G is thus a subset of G that qualifies as a tree and contains all the nodes within G. A valid parse of S could therefore be defined as a spanning tree that starts at the root node [24].

In essence for a given sentence, a search is performed through some search space, which is comprised of directed graphs, for a spanning tree that maximises some score. An assumption is made that the score of a dependency tree can be derived as the sum of the scores for each edge that the tree is composed of. The parser has to assign scores to each edge and find the maximum spanning tree [34]. The Chu-Liu-Edmonds algorithm can be used to compute this tree [**?**]. The algorithm has a series of phases. The first phase involves iterating over each node in the graph and choosing the incoming arc with the highest score – this is a greedy selection. Then a check is performed to see if the resulting edges produces a spanning tree. If that is the case, then the process is finished. Otherwise, if the tree has any cycles in it then the edge weights need to be scaled and a recursive cleanup phases, which involves collapsing certain nodes, finishes off the process [8]. Figure 6 shows a maximum spanning tree for the sentence "Book that flight".

The edge score can also be computed using a weighted sum of the features that can be extracted from it. In this case, the relevant features must be identified and the weights then trained [51]. These features can include (but are not limited to): word embeddings, the dependency relation, parts of speech of the head word and its dependents [24]. Graph-based dependency parsers need to be evaluated by using finer-grained metrices: labelled attachment accuracy (LAS), unlabelled attachment accuracy (UAS) and label accuracy score (LS). LAS examines the appropriate assignment of a word to its head and checks that the dependency relation is correct. UAS ignores the dependency relation and checks the accuracy of the assigned head. Finally, LS is concerned with the percentage of tokens with the correct labels (and ignores the relations) [24].

5 SEMANTIC PARSING

Semantic parsing is concerned with the task of mapping natural language utterances into a semantic representation [35]. These representations can be referred to as meaning representations (MR), which can be more easily understood and processed by computers.

Minimal recursion semantics [10], abstract meaning representations [2] and dependency-based compositional semantics [30] are a subset of the possible formalisms that can be used to express these meaning representations. [25]. Semantic dependency parsing aims to capture the between words relationships, in order to determine the meaning of a sentence. This is done using a graph structured (as opposed to tree) representation, which allows for more linguistic information about a sentence to be captured [9].

5.1 Corpora

A number of corpora are available for use with neural networks. These corpora contain annotated semantic and syntactic text. Some examples of widely used corpora include DeepBank, PropBank and Penn treebank. These corpora need to be large, as the dataset will be split up into a training set, test set and evaluation set for the model.

5.2 Tokenisation and Lemmatisation

Once a corpus has been selected, the data has to be preprocessed before it can be fed into a neural network. The words are divided into units – a process known as tokenization [27] . For English, this is usually a straightforward process – spaces are the main mechanism used to split sentences [6]. These tokens can then be lemmatized. This is when a word and all its inflected forms are grouped together and viewed "as one" i.e. they can be analysed as a single word form. [27]

5.3 Meaning Representation Frameworks

Meaning Representation frameworks are distinct from the aforementioned corpora. Corpora simply provides annotations for syntactic and semantic structure. Meaning representation frameworks, however, provide a blueprint for how these graphs structures can be created and rendered - they each have assumptions about the relationships between a sentence and the nodes that form part of the graph. There are five main frameworks for meaning representation parsing.

(1) DELPHI-IN MRS (DM): Asymmetric lexical relations between heads and dependents is what describes a bi-lexical dependency [21]. DM, like these other frameworks, can be used to generate graphs and under this framework, the edges largely signify semantic argument positions e.g. ARG1. Node labels consist of the lemmatized word, a POS label and a frame identifier [22].
(2) Elementary Dependency Structures: This framework is based on DM and creates a semantic dependency graph that (unlike DM, for example), is not restricted to bi-lexical dependencies. In this paradigm, logical predications form the nodes of the graph and labelled argument positions form the edges. EDS does not have a one-to-one correspondence between words in the sentence i.e. the tokens, and nodes in the graph. Nodes are rather anchored on spans, which can overlap with other nodes in the graph [42]. Figure 7 shows an EDS graph generated from an English sentence.

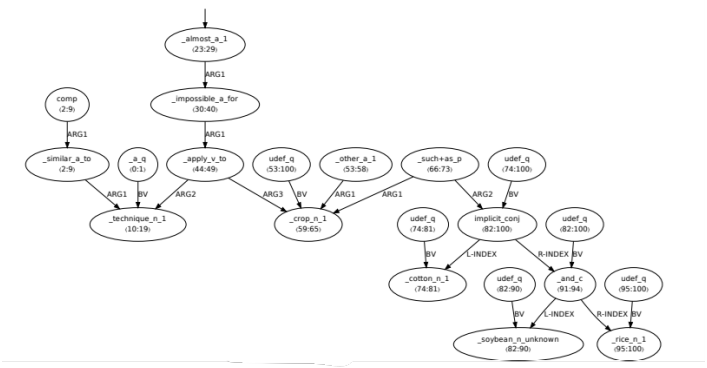


Figure 7: EDS graph for sentence: *A similar technique is almost impossible to apply to other crops, such as cotton, soybeans and rice* [41]

The Prague Semantic Dependencies, Universal Conceptual Cognitive Annotation and Abstract Meaning Representation frameworks will only be explained in brief, as they are not the focus of the project:

- Prague Semantic Dependencies (PSD): As with DM, PSD uses bi-lexical dependencies. However, edges labels with this framework are not usually functional i.e. multiple outgoing edges from one node with the same label is not permitted.
- Universal Conceptual Cognitive Annotation (UCCA): This framework was derived using cognitive linguistic and typological theories, and proposes the notion of unlabelled nodes called units and one or more labels per edge.
- Abstract Meaning Representation (AMR): Finally, this framework provides no explicit rules for how the words in the sentence must relate to the graph nodes. Thus, these nodes might be far more "abstracted" than the tokens that were given as input [41].

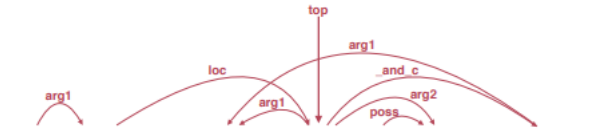
5.4 Semantic Role Labelling

Semantic role labelling is a particular kind of semantic parsing. Here, given a statement, we are trying to identify which groups of words can be linked as arguments to a given predicate. We also try to determine what an argument's role (with respect to the predicate) is [9]. The output of this will be a tree structure. Figure 8 shows an example of arguments, linked to predicates according to the DM framework.

Semantic role labelling can be performed using CRFs [9] or a BiLSTM [19].

5.5 Semantic Dependency Parsing

These semantic trees can be generated by means of taking tokenized input and producing semantic dependency schemes in the form of directed acyclic graphs. [12] concatenated word and POS embeddings, fed these through a multilayer bidirectional LSTM and finally used biaffine classifiers in order to predict arcs and labels. Biaffine classifiers are a method of modelling binary relations by using an attention mechanism [33], [43] uses concepts from



(a) DM

Figure 8: Semantic role labels for an English sentence, showing the arcs from head to dependents [43]

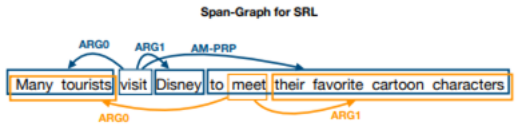


Figure 9: Span-Graph for Semantic Role Labelling [19]

graph-based dependency parsing to create a semantic dependency parser for parsing sentences into three semantic dependency graph formalisms – namely DELPHI-IN MRS (DM), Predicate-Argument Structures (PAS) and Prague Semantic Dependencies (PSD). This was achieved again through using a biLSTM composed with a multi-layer perceptron that scored arcs and predicates, for each of the formalisms.

6 SEMANTIC GRAPH PARSING

Semantic graph parsing is the process of determining the semantic meaning of a sentence and converting it to a graph representation. Semantic graph parsing can thus be modelled as a graph prediction problem [15]. Many of the parsing concepts and algorithms used for tree structures can be extended to graphs. [13] showed that a semantic graph could be converted to a tree, which could be used as input to a graph-based tree parser. The output of the parser was then converted back into a graph. [3] used the arc-eager transition system to create a semantic graph parser for the minimal recursion semantics formalism. This parser achieved a high SMATCH score of 86.69 and had a higher accuracy compared to the MRS baselines (attention-based).

[5] provides a thorough, detailed dissection of two approaches to semantic graph parsing: the knowledge-intensive, grammar rule driven approach and a data-intensive, maximum-score for a whole graph approach. This review focuses on the latter approach.

6.1 Evaluation

Parsing accuracy i.e. correct graph identification can be evaluated by the metric of SMATCH. It can be used to evaluate whole-sentence semantic structures. These whole-sentence semantic structures are the output of semantic parsing processes, which, given some input, attempt to produce all semantic relationships present in that input. SMATCH takes two semantic feature structures and calculates the degree of overlap between them [4].

6.2 Node Prediction

As mentioned earlier, an input sentence must be tokenised before it gets parsed. Following that, a sequence labeller, such as a CRF, may be used to predict the various concepts for each of the tokens[5]. If a node's label has a conceptual meaning, then a node can be referred to as a concept. There are two classes of node labels: surface and abstract [6]. Furthermore, there is no assumption that there is a one-to-one relationship between the tokens and the nodes in the graph. Nodes may in fact correspond to a sub-token, or multiple tokens. The sequence labeller would thus need to align the tags and concepts based on a word's span information – and a set of heuristic rules may also be necessary [5]. Should data sparseness become an issue, lexical predicates would need to be delexicalised [3]. This word tagging problem can be formulated mathematically as follows:

$$\sum_{n \in \text{NODE}(G)} \text{SC}_n(s,n) \approx \sum_{1 \leqq i \leqq m} \max_{st_i \in ST} SC_{st}(s,i,st_i) \qquad (10)$$

[5] used a BiLSTM with a softmax layer for classification. In the above equation, G represents a graph and $SC_{st}(s,i,st_i)$ is computed by using softmax on the output of the BiLSTM. They used character-based and word-based word-embeddings for their model. ELMo, a contextualized representation model similar to BERT, was used in place of static word embeddings [14]. Their ELMO* model was able to achieve a tag accuracy of 95.38% (compared to 94.51% for Word2Vec), and an F1 score of 97.04 (compared to 94.87% for W2V) [5].

6.3 Edge Prediction

Edge prediction forms the second stage of the data-intensive parser. Though predicting nodes and edges can be done at the same time, as seen with [19]. Output from the previous stage i.e. the predicted nodes, are used for this one. Identifying these dependencies can be framed as an optimization problem, represented mathematically as:

$$\hat{G} = arg \max_{G \in \mathcal{G}(\mathcal{N})} \sum_{(p,a) \in \text{ARC}(G)} \text{SC}_a(s,N,p,a) \qquad (11)$$

G = the set of all possible graphs that accept N as their node set. By employing a factorization-based approach, a graph can be measured using the scores of local subgraphs and summing them to find some maximum. A score is calculated for all directional arcs between two node in a graph. A multi-layer perceptron is used to get each arc's scores for all possible labels and the max one is selected as the final label. Dependency graphs can be measured using labelled and unlabelled, precision/recall as well as an F_1 score [5].

6.4 Span Prediction

Rather than predicting labels for single tokens as the papers mentioned in the previous section have done, some of the research has instead focused on predicting labelled spans. One of the problems with BIO-tagging based neural semantic parsers is that they can't integrate span-level features. Additionally, [19] notes that BIO-tagging and Markov based models for semantic role labelling have some limitations, in that they cannot model overlapping spans

that form part of multiple predicates within the same sentence [19]. This research aimed to create an end-to-end model that could address these aforementioned issues. They were able to develop a model that was able to incorporate span-level features and perform joint predicate identification with a high degree of accuracy - outperforming the previous best system (at the time) for SRL created by [20]. Figure 9 shows a graphical representation of the labelled spans and arcs that their model predicted. [23] examined the need for building task-specific architectures for 10 different NLP tasks. Some of these tasks included NER, POS tagging, SRL and dependency parsing. The main problem the research was trying to address was: rather than creating task-specific architectures to perform these NLP tasks, is it possible to devise a task-independent model that can be used across these NLP and provide a comparable, if not higher degree of accuracy compared to these task-specific architectures. They were able to formulate a model, called SpanRel, which is largely task agnostic. The one requirement was that a task needed to be formulated as a span-relation prediction problem in order for the model to operate on it. SpanRel was able to perform competitively with the more task specialized models.

7 MOTIVATION

There are several motivations for this project. Firstly, this project aims to use the newer approach of having pretrained transformers form the basis of the model. Much of the literature has used word embeddings and LSTSMs . Furthermore, for the graph-based parsing part, the prediction of nodes and edges will be done separately, as opposed to much of the literature, where this is done jointly [19] [23] [43]

8 CONCLUSION

Advances in machine learning, such as the development of the transformer model, have had a significant impact on the field of NLP. Small, initial research has shown that as an encoder in the form of BERT, it can be applied to a wide variety of NLP tasks. But, more research needs to be done to determine whether the transfomer can supersede the BiLSTM that is widely used in semantic and syntactic parsing, and whether contextualised representations models, such as BERT, can replace traditional, static word embeddings. Semantic parsing is applicable to question answering, machine translation, code generation and many other use cases. As demand for better systems increase, new parsers that are efficient and accurate will need to be developed. Therefore, there is a great need to take new ML techniques, such as transformers and test to see if and how it may improve semantic graph parsing.

REFERENCES

[1] Felipe Almeida and Geraldo Xexéo. 2019. Word embeddings: A survey. *arXiv preprint arXiv:1901.09069* (2019).

[2] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse*. 178–186.

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

[4] 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.

[5] 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.

[6] 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.

[7] Jinho D Choi and Martha Palmer. 2011. Getting the most out of transition-based dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. 687–692.

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

[9] Trevor Cohn and Phil Blunsom. 2005. Semantic role labelling with tree conditional random fields. (2005).

[10] Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A Sag. 2005. Minimal recursion semantics: An introduction. *Research on language and computation* 3, 2 (2005), 281–332.

[11] 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).

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

[13] 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.

[14] Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings. *arXiv preprint arXiv:1909.00512* (2019).

[15] 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).

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

[17] James Hammerton. 2003. Named entity recognition with long short-term memory. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*. 172–175.

[18] Han He and Jinho Choi. 2020. Establishing strong baselines for the new decade: Sequence tagging, syntactic and semantic parsing with BERT. In *The Thirty-Third International Flairs Conference*.

[19] 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).

[20] Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 473–483.

[21] Angelina Ivanova. 2015. Bilexical Dependencies as an Intermedium for Data-Driven and HPSG-Based Parsing. (2015).

[22] Angelina Ivanova, Stephan Oepen, Lilja Øvrelid, and Dan Flickinger. 2012. Who did what to whom? A contrastive study of syntacto-semantic dependencies. In *Proceedings of the sixth linguistic annotation workshop*. 2–11.

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

[24] 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.

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

[26] M. V. Koroteev. 2021. BERT: A Review of Applications in Natural Language Processing and Understanding. https://doi.org/10.48550/ARXIV.2103.11943

[27] Kristopher Kyle. 2021. Natural language processing for learner corpus research. *International Journal of Learner Corpus Research* 7, 1 (2021), 1–16.

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

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

[30] Percy Liang, Michael I Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics* 39, 2 (2013), 389–446.

[31] Elizabeth D Liddy. 2001. Natural language processing. (2001).

[32] H.-L. Lou. 1995. Implementing the Viterbi algorithm. *IEEE Signal Processing Magazine* 12, 5 (1995), 42–52. https://doi.org/10.1109/79.410439

[33] Tomoki Matsuno, Katsuhiko Hayashi, Takahiro Ishihara, Hitoshi Manabe, and Yuji Matsumoto. 2018. Reduction of parameter redundancy in biaffine classifiers with symmetric and circulant weight matrices. *arXiv preprint arXiv:1810.08307* (2018).

[34] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of human language technology conference and conference on empirical methods in natural language processing*. 523–530.

[35] Raymond J Mooney. 2007. Learning for semantic parsing. In *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, 311–324.

[36] Marwa Naili, Anja Habacha Chaibi, and Henda Hajjami Ben Ghezala. 2017. Comparative study of word embedding methods in topic segmentation. *Procedia computer science* 112 (2017), 340–349.

[37] Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the eighth international conference on parsing technologies*. 149–160.

[38] Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. 351–359.

[39] 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

[40] Joakim Nivre, Marco Kuhlmann, and Johan Hall. 2009. An improved oracle for dependency parsing with online reordering. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*. 73–76.

[41] 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*. 1–27.

[42] Stephan Oepen and Jan Tore Lønning. 2006. Discriminant-based MRS banking.. In *LREC*. Citeseer, 1250–1255.

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

[44] Natalia Ponomareva, Paolo Rosso, Ferrán Pla, and Antonio Molina. 2007. Conditional random fields vs. hidden markov models in a biomedical named entity recognition task. In *Proc. of Int. Conf. Recent Advances in Natural Language Processing, RANLP*. 479–483.

[45] Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*. Springer, 157–176.

[46] Helmut Schmid. 1994. Part-of-speech tagging with neural networks. *arXiv preprint cmp-lg/9410018* (1994).

[47] Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 conference on empirical methods in natural language processing*. 298–307.

[48] Ralf C Staudemeyer and Eric Rothstein Morris. 2019. Understanding LSTM–a tutorial into long short-term memory recurrent neural networks. *arXiv preprint arXiv:1909.09586* (2019).

[49] 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.

[50] 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).

[51] Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2306–2315.

[52] Hang Yan, Bocao Deng, Xiaonan Li, and Xipeng Qiu. 2019. TENER: adapting transformer encoder for named entity recognition. *arXiv preprint arXiv:1911.04474* (2019).

[53] Yue Zhang and Joakim Nivre. 2012. Analyzing the effect of global learning and beam-search on transition-based dependency parsing. In *Proceedings of COLING 2012: Posters*. 1391–1400.