UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS/IT Honours Project Final Paper 2022

Title: Node Prediction in Meaning Representation Parsing

Author: Imrie, Jane (IMRJAN001)

Project Abbreviation: MRP

Supervisor(s): Buys, Jan (jbuys@cs.uct.ac.za)

Category	*Min*	*Max*	Chosen
Requirement Analysis and Design	*0*	*20*	0
Theoretical Analysis	*0*	*25*	5
Experiment Design and Execution	*0*	*20*	20
System Development and Implementation	*0*	*20*	0
Results, Findings and Conclusions	*10*	*20*	20
Aim Formulation and Background Work	*10*	*15*	15
Quality of Paper Writing and Presentation	*10*		10
Quality of Deliverables	*10*		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	*0*	*10*	
Total marks		**80**	

Node Prediction in Meaning Representation Parsing

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

ABSTRACT

Semantic Graph Parsing involves parsing the semantic meaning of a sentence and representing it in graph form. The process is comprised of concept identification, otherwise known as node prediction, and relation identification. In the paper we investigate the use of pre-trained transformers, namely SpanBERT, on concept identification. We also compare these models to the combination of BiLSTM and regular Glove embeddings. Furthermore, we explore whether an additional and costly Conditional Random Fields "layer" is worth including in a model i.e. how the layer affects the metrics of tag accuracy, node prediction precision, recall and the F1 score. Our research shows that SpanBERT was able to substantially outpeform Glove across all categories. We also show that a CRF layer does not have a significant effect on model accuracy.

KEYWORDS

Semantic Graph Parsing, Natural Language Processing, Semantic Parsing

1 INTRODUCTION

Natural language Processing uses many different types of computational techniques in order to perform linguistic analysis on a variety of texts across languages, with the broad goal of trying to achieve human-like language processing over a variety of applications[34]. One way this can be achieved is through syntactic and semantic parsing of sentences into various representations. These representations can then be used to perform a number of downstream NLP tasks, such as machine translation, question-answering and sentiment classification.

Given the state of technology and how it as advancing, it has become feasible to leverage deep learning technologies and techniques for tasks within the field of NLP. Semantic meaning representation parsing, that is, the process of converting some form of input sentence into a semantic meaning representation, has especially benefited from the use of BiLSTMS, as seen in [10] and [7], amongst others. Two of the most popular parsing systems in the literature is transition-based and semantic graph-based parsing [36]. In the latter, the parser outputs a graph structure - a form which is most advantageous due to how flexibly it allows for these representations to be made [39]. This process can be split into two parts (1) node prediction (otherwise known as concept identification) and (2) edge prediction.

A number of frameworks exist which provide rules for the parsing process, such as Dependency-based Minimal Recursion Semantics [13], Elementary Dependency Structures [38], Prague Semantic Dependencies, Universal Conceptual Cognitive Annotation [1] and Abstract Meaning Representations [4].

Presented with the above, this research attempts to pivot off the work done by [7].Their work focused on comparing different kinds of semantic parsers - chiefly, they created a new factorisation-based parser which created conceptual graphs for sentences. This parser used pre-trained embeddings and operated in a pipeline fashion. The first stage, concept identification, was optimised and the best predictions were selected for the following phase, dependency detection (edge prediction). This parser was then pitted against more conventional, knowledge intensive parsers.

In this paper, we investigate and compare the use of the BiLSTMs and pre-trained transformer model on node prediction accuracy (the first part of [7]'s work), which is measured in terms of precision, recall and the F1 score. Furthermore, we determine to the effect the addition of a probabilistic Condition Random Fields layer has on model accuracy [33] and assess if it would thus be worth including given its cost. This report is comprised of a background information section, a brief section on work relevant to our research, followed by details of the experiments, their results, the conclusions drawn from them, limitations and a small mention of future work.

2 BACKGROUND

2.1 Natural Language Processing and Computational Linguistics

2.1.1 Sequence Labelling. Natural Language processing contains 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 [27]. Many languages 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. E.g – "she's busy smoking" vs "that's a no-smoking sign". Part of speech tagging is the process of assigning labels to words, based on their contextual information [44].

Proper nouns, like "Cape Town" are semantically viewed as different types of entities, referred to as named entities. These are not strictly proper nouns and include other attributes like currency, dates, field-specific types like chemicals, etc. Named entity recognition (NER) can therefore be defined as the process of identifying and tagging spans of text that can be classified as named entities[29]. BIO tagging is one of the NER tagging methods[43]. BIO tagging performs NER by treating it as a word-by-word sequence labelling task, using tags that capture the boundaries of the span, as well as it's type. A token that begins with the span 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". An illustrative example:

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

"Sequence labelling" can thus be thought of in the general sense as a pattern recognition task. Given a sequence of values, an algorithm is used to assign a categorical label to each value. It includes POS tagging and NER.

2.1.2 Conditional Random Fields. Two of the most popular sequence labelling algorithms are the Hidden Markov Model (HMM) and Conditional Random Fields (CRFs). HMM uses Markov chains - it assumes a word is conditioned on its tag, and seeks to maximise the joint probability of a paired observation and a label sequence [42]. E.g. finding the probability of "high" coming after the word "jump" i.e. $P(adverb\ |verb)$, and the probability of "jump" being a verb i.e. $P(jump|verb)$ [29].

X is a sequence of input words $(x_1, \ldots, x_n)$, and the aim is to compute Y a sequence of output tags $(y_1, \ldots, y_n)$. Bayes' rules and the likelihood $P(X|Y)$ is used by HMMs in order to compute the tag sequence that maximises $P(Y|X)$. POS tagging uses Bayes', as shown in the next equation. Given $w_1 \ldots w_n$, a sequence of n words, the goal is to find the most probable tag sequence $t_1 \ldots t_n$:

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

Conversely, CRFS directly calculates the posterior $p(Y|X)$. It computes log-linear functions at each time step, over a set of applicable features. Given X,a probability is assigned to Y, out of (all possible output sequences), $\mathscr{Y}$. X and Y is mapped by F, a function, 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 using this method [29]:

$$p(Y|X) = \frac{exp(\sum_{k=1}^{K} w_k F_k(X, Y))}{\sum_{Y' \in \mathscr{Y}} exp(\sum_{k=1}^{K} w_k F_k(X, Y'))} \tag{2}$$

The Viterbi algorithm is used for inference and training the CRF and HMMs [29] and uses dynamic programming to find the optimal sequence of tags. [35] and [29] provides a more in-depth explanation for implementing the algorithm, and the mathematics underpinning it.

HMMs, are able to (in quite a limited sense) indirectly consider the context of a word. However, CRFs are comparatively far more flexible in how it can use a word's context. This is favourable, as often a word's context can play an important role in what it's correct label is. That is, 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 [29]. CRFS have been shown to outperform HMMs for POS tagging because of their flexibility [32].

2.1.3 Meaning Representation Frameworks. The meaning of a sentence can be depicted using a semantic representation. This representation can be in the form of a tree or graph structure. In the case of the latter, there are a number of formalisms which allow for effective encoding of rich semantic information for given language inputs [9]. These meaning representation frameworks act as blueprint, defining how these semantic graph structures can be generated, as well as enforcing general assumptions about sentence-node relationships in the context of the graph itself. Two of these,

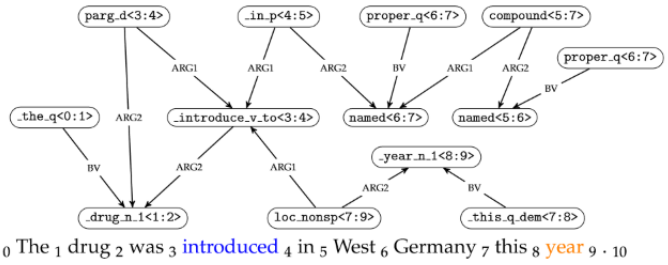

Figure 1: An EDS graph with labelled nodes and edges. Both surface and abstract nodes are present. *compound* is an example for an abstract node and *_introduce_v_1* is the label for a surface node [7].

which are pertinent to this research, are DELPHI-IN MRS (DM) and Elementary Dependency Structures (EDM) [38]:

(1) DM: a bi-lexical dependency describes asymmetric lexical relations between heads and dependents [25]. Under DM, the edges largely signify semantic argument positions e.g. ARG2. Node labels consist of the lemmatized word, a POS label and a frame identifier. [26]
(2) EDS: based on DM and creates a semantic dependency graph that (unlike DM), isn't restricted to bi-lexical dependencies. Logical predications form the nodes of the graph and labelled argument positions form the edges. There is no assumption of 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 [38].

2.1.4 Semantic Parsing. In the context of NLP, a parser is tasked with analysing a natural language utterance, such as a sentence, and breaking it up into components based on some criterion. These may be semantic or syntactic, for example. A factorisation-based parser, such as seen in [10], has a score function, which it uses to evaluate candidate graphs that are derived from the input sentence. Since a sentence can have any number of possible graphs, the parser is also tasked with retrieving graphs with the highest scores from the set of all possible graphs for the given input. This type of parser essentially needs to model the expected elements for the target semantic structure. These elements are (1) labelled nodes, (2) node edges and (3) node characteristics. Sometimes, a label for a given node has a conceptual meaning and thus a node can then be called a "concept". These concepts can be divided into two classes [10]:

(1) Surface: these are derived from the words themselves - their orthography provides the crucial part of the label
(2) Abstract: these are used to symbolise more specialised labels or the influence of grammatical constructions

Concept identification is thus the process of receiving tokenised input and establishing under which class (if any) each token falls. However, it should be noted that a surjection exists between the input and surface nodes. Furthermore, the connection between some abstract concepts and all surface concepts to nodes in the candidate graph is usually straightforward and easy to determine.

In the literature, these are known as *lexicalised concepts* and the task of their identification can be formalised as a tagging problem on the token level.

The identification of the other abstract concepts is far less trivial and there are differing approaches on how to achieve it. [10] proposed using a word-level tagger and some heuristics rules, and was able to achieve F1 scores of between 95 and 96 for their non-lexicalised concept identification.

2.1.5 Semantic Graph Parsing. Semantic graph parsing is the procedure of determining the semantic meaning of a sentence and converting it to a graph - therefore, it can be modelled as a graph prediction problem [18]. Many dependency parsing concepts and algorithms, which have tree structures, can be extended to graphs. Graph-based dependency 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 [11]. Analogously, for a semantic graph-based parser, the aim is to find the maximum subgraph i.e. a subgraph where the sum of the scores of the semantic structures is highest. SMATCH is a metric that can perform this, as it assesses whole-sentence semantic structures. These structures are the output of semantic parsers, which attempt to produce all semantic relationships present in a given input. SMATCH takes two semantic feature structures and calculates the degree of overlap between them [6].

2.2 Machine Learning

"Words" as they are understood in language, need to be represented in a numerical form in order to be used for machine learning. This representation is known as word embeddings, commonly calculated as vectors [3].

A neural network consists of computing units that require vectors as input and outputs a single value. The most basic neural network is the feed-forward (FF) single layer perceptron, which consists of an input layer, at least one hidden layer, a non-linear layer and an output layer. This can be represented mathematically in terms of vector-matrix operation, where b represents a bias term and W the standard weight matrix:

$$NN_{perceptron}(x) = xW + b \tag{3}$$

However, these kinds of networks cannot use sequential context effectively when making predictions. This can be problematic, as some NLP tasks may require the use of information that is indiscriminately distant from the current word being processed. Recurrent Neural Networks (RNNs), however, do possess this mechanism. This added temporal dimension means Simple RNNs (a type of recurrent network) use earlier outputs from previous timesteps to form part of the input to a neuron (computing unit) [29].

Despite their advantanges, RNNs suffer from the phenomenon of vanishing gradients - the gradient becomes lower as the network is trained on long sequences. This increases the difficulty of training the weights [47]. Furthermore, if the information is very distant from where the current point is being processed, in practice it makes training the network very challenging. The local encoded information is more relevant to the newer parts of the the input series, even though the network has access to the entire preceding series [48].

Long Short Term Memory (LSTM) neural networks address the aforementioned problem and serve as an extension to the simple RNN. The "context management problem" characterised in the RNN, that is, the inability to provide helpful information to the current decision whilst also "remembering" information which needs to be brought forward for future decisions, is not present in the LSTMs. LSTMs prevent this by: (1) removing unessential information from the current context and (2), keeping possession of only the information with a high likelihood of being required in future decision making [47]. An in-depth explanation of how the network realises this is beyond the scope of this paper. However, LSTMs, due to this useful characteristic, have proven to be a favoured choice for various NLP tasks, and are used in [23], [16],[39], [50] and [27]. More details of the specific use cases for some of these can be found in later sections of this paper.

The next advancement in neural networks which has proved applicable to NLP tasks are transformers. Their use has become somewhat widespread across various types of NLP activities [52]. Having an underlying encoder-decoder structure, a transformer accepts an input sequence, $(x_1, .., x_n)$. This gets mapped to continuous representations by the encoder labelled z: $(z_1, \ldots, z_n)$ and an output sequence, $(y_1, \ldots, y_n)$, is generated element-wise by the decoder. However, due to its modular nature, the transformer can be used as either an encoder or decoder.

The model is composed of blocks, which are themselves made up of feed-forward layers, normalising layers, a self-attention layer and residual connections. What differentiates and elevates the transformer above other kinds of neural networks is the "self-attention" mechanism. Under the attention-based approach, given both an item and collection of items of interest, they are compared in order to determine their relevance within the current context. Current input utilises the results of these comparisons to generate output. This notion allows for the model to efficiently capture dependencies of words, regardless of distance between them. Additionally, input tokens can be processed simultaneously, allowing the transformer to parallelise learning far more efficiently then other kinds of networks [49].

2.2.1 Deep contextualised word representations. The same word can mean different things in different contexts - “I eat an apple everyday” vs “Apple is a tech company” shows how the word “apple” drastically differs in meaning between the two contexts. Despite this, for static embeddings, the vectors in both cases for “apple” are the same. Contextualised word representations do not have this limitation - the vectors are sensitive to the contexts in which they appear [17].

Deep contextualised word representations are a step above this. A neural network is composed of many layers, and deep contextualised representations able to capture these contextualised representations over several layers. Additionally, they have the added advantage of being pre-trained on much larger corpora than the standard treebank [31]. BERT (Bidirectional Encoder Representations from Transformer) is a language model used for deep contextualised word representations that is becoming more widespread in its use throughout the literature. It is a bidirectional transformer

that uses left and right context in all its layers and unlabelled text is used to train it. A pre-trained BERT model can be placed into an existing pipeline, and through the addition of an output layer to the model's architecture it can be further fine-tuned to the specific task. BERT's versatility can be seen in[22], where it was used for POS tagging, semantic and syntactic parsing.

3 RELATED WORK

3.1 Comparing Knowledge-Intensive and Data-Intensive Models for English Resource Semantic Parsing

One of the key papers that influenced this experiment is [7]. In that paper, the researchers created two parsers for English Resource Semantics (ERS) graphs- one of which is a data-intensive model, the focus for this research. ERS is an extensive framework used for linguistic analysis [19].

This parser operated as a pipeline, whereby node and edge prediction was performed sequentially. Other research, such as [23] and [27], predict nodes and edges jointly. However this is not mandatory and others, such as [53], [8] and [10] have separated the process. [7] used Word2Vec, a basic type of word embedding as their baseline and pre-trained ELMo models for comparison.

ELMO (Embeddings from Language Models) is a type contextualised word representation model, like BERT. It is composed of a 2 layer BiLSTM that is trained on a bidirectional language modelling task. The concatenated internal states of the BiLSTM is used to generate a contextualised representation of each token. However, this concatenation can operate to the detriment of the model, as it hinders the ability of the representation to take advantage of both contexts simultaneously[41]. BERT builds upon some of the ideas put forward by ELMo, using transformers. It uses a masked-language objective, which essentially means that it masks words at random and replaces these with a mask. The unmasked words to the left and right are used to predict the encapsulated masked words. [15].

Their work showed that ELMo* was able to outperform all other models in terms of concept identification and tag accuracy. Though this research aims to employ similar techniques and heuristics, [7] endeavoured to compare knowledge-intensive and data-intensive parsers, rather than just focusing solely on data-intensive models. Knowledge-intensive parsers utilise lexical and syntactico-semantic rules, usually governed by a grammar, in order to derive semantic graphs. Conversely, data-intensive parsing involves modeling the target graph - each part of the graph may be associated with a score and the aim is to find the graph with the highest sum of these partial scores.

Additionally, the research does not examine the affects of adding a Conditional Random Fields layer on concept identification accuracy.

3.2 Generalizing Natural Language Analysis through Span-relation Representations

The current standard for many NLP activities is to build a task-specific architecture for the charge. [27] thus investigated if (1) a task-independent model could be created and used across various different types of NLP tasks and (2), could that model provide comparable or increased accuracy than the task-specific architectures. The result of this was the task-agnostic SpanRel - which merely required that the task be framed as a span-relation prediction problem. SpanRel was able to compete with more task-specialised models for the same NLP tasks. The relevance of SpanRel to this research is that, SpanRel's architecture scores spans directly, instead of using BIO or tree2label to encode spans as tags.

3.3 Jointly Predicting Predicates and Arguments in Neural Semantic Role Labeling

This research has focused on predicting labelled spans. BIO-tagging based semantic parsers have some problems, one of which is that they can't integrate span-level features. [23] also notes that BIO-tagging models for semantic role labelling have some shortcomings - they cannot model overlapping spans that form part of multiple predicates within the same sentence [23]. Thus, these researchers set about to create an end-to-end model to tackle these limitations. The model they developed was able to incorporate span-level features and perform joint predicate identification. Their model produced state-of-the art results, beating the previous best system (at the time) for SRL created by [24].

3.4 Factorization- and Composition-Based Parsing for Elementary Dependency Structures

[10] created and contrasted factorisation-based and composition-based approaches to semantic parsing for EDS. They created a four stage pipeline, starting from tokenisation (at the word-level in their case), and continuing to concept identification, relation detection and property prediction (such as spans/anchors). Their factorisation parser out-performed their composition-based one across categories, including anchor, edge and label. Concept identification was performed using a multi-layer BiLSTM, as well as ELMo and BERT contextualised representation models.

4 RESEARCH QUESTIONS AND PROBLEM DEFINITION

4.0.1 ***RQ1****. How does a fine-tuned BERT encoder compare to an BiLSTM with a standard word-embedding layer in terms of accuracy of node prediction?*

Accuracy in this context, can be defined using the metrics of precision and recall and F1. Here, were are trying to determine the effect of using BERT when compared to the more popular BiLSTM, on the values of these metrics. In essence, we are trying to assess if the transformer, with its many useful characteristics, is a viable successor to the BiLSTM in the case of concept identification in semantic graph parsing. We hypothesise, due to the transformer's unique attention mechanism and the promise showed by BERT in other literature for other NLP tasks [22], that models that use BERT will outperform those that utilise Glove [40].

4.0.2 ***RQ2****. To what extent does the addition of a CRF layer affect model accuracy, measured in terms of precision, recall and F1 score?*

In a similar vein to the previous question, we are trying to investigate what effect the use of a CRF layer has on the metrics used to

evaluate accuracy. The CRF has a quadratic cost, and it therefore becomes necessary to ascertain if it has any substantial affect on these values, or whether it is better to exclude it from the models, which would reduce computational costs to some extent. Given how effectively the CRF is able to use context, we speculate that models that contain this additional layer will produce higher scores and therefore the inclusion is worthwhile.

4.1 Problem Definition

We cast concept identification as a sequence labelling problem of two parts, given that we are separately predicting surface/lexicalised and abstract/non-lexicalised concepts. The first is the lexicalised concept prediction. In the context of this research we are trying to predict nodes, which are composed of spans. Surface span prediction can thus be seen as being most similar to an NER problem. Second, given that we transformed the abstract node labels into a set of tags who structure itself encodes the span information, the prediction of these tags is somewhat analogous to a POS tagging problem.

5 EXPERIMENT METHODOLOGY

5.1 Evaluation Metrics

Generally speaking, sequence taggers can be evaluated using the metrics of precision, recall and the F1 measure, which is the harmonic mean between precision and recall [21].

$$\text{Recall} = \frac{\text{number of correctly labelled items}}{\text{total number of items that should have been labelled}} \tag{4}$$

$$\text{Precision} = \frac{\text{number of correctly labelled items}}{\text{total number of labelled items}} \tag{5}$$

Finally, the F1 score can be computed as:

$$F_1 = \frac{(2 * \text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})} \tag{6}$$

In the case of this research, the micro-averaged precision, recall and F1 score was used to calculate the node label accuracy. The micro-average treats the entire dataset as an aggregate, rather than averaging the metric of k classes. The gold and predicted tags were converted into a scalar corresponding to the index of the tag in the list of all possible tags, placed into a `pytorch tensor` and fed to the `Torch Metrics` [14] package for the final calculations. This general-purpose metrics package allows for easy replication for this part of the results given the appropriate data.

For concept identification, the gold (i.e. target) spans are extracted from the data. A labelled item in this context consists of a node label and span information. For both the gold and predicted spans, any unlabelled token is removed from the set before the above metrics are calculated, as per the method used in [7]. The `evl_edm.py` script used in [5] was modified in order to perform the evaluation.

5.2 Data Pre-Processing

Sourced from the Redwoods corpus [37], the data was extracted and split into training, development and test sets by means of a pre-made script [1]. The Deepbank EDS semantic annotations with span-level normalisations were selected. The script provides the tokenised input sentences, node labels, as well as edge labels, of which only the former two were needed for this research.

This resulted in a split of 35598 train, 1813 dev and 1482 test sentence sets. Some inspection was performed to check the tag distribution and this showed that most of the tags appeared in each set. If the distribution was too skewed i.e. a significant proportion of tag types only appeared in either the test or dev sets, this would be problematic. However, this was not the case for this corpus.

It should be noted that not all tokens were labelled - these were allocated a label of "None" for the purposes of prediction with the neural networks. Table 1 shows a labelled sentence extracted from the corpus. Note that “60%-held”, which is viewed as "one-word", has been split into two tokens according rules found in the EDS framework.

5.2.1 Tag Processing. The combination of surface and abstract labels resulted in a label set size of over 8000 items. This would prove infeasible in terms of memory requirements in order to train and thus it was decided that the surface and abstract labels would be predicted separately. This would also allow for a more finer-grained control over the parameters for each model, which we surmised would lead to better results.

Surface node labels may contain the lexical part of the token that aligns to it, for example in 1, the tag for “The” is `_the_q`. The can lead to a data-sparseness problem when training.[5] suggests delexicalising these predicates by replacing the lemma with a “*”. The labels for this corpus simply removed the lemmas.

We needed to establish what the labet set sizes for the different types of nodes would be. The expectation was that there would be minimal overlap between the spans for surface node labels and thus a smaller label set size. Some exploratory data analysis was performed and this assumption did hold. This can be seen visually in figure 2.Following that, BIO tags [29] were assigned to each token, in the form “B/I/O-surface label”. “B” was assigned to the beginning of a span for a surface concept, and an “I” for any following concepts that had the same tag. This can be seen in table 1, where “American Express” has the tags [B-named, I-named], indicating a span.“O” was assigned to tokens that had the “None” label, as is the case for “is” in that table. Some surface nodes had multiple labels (due to having membership in overlapping spans) and in those instances those labels were concatenated, using a “;” as a delimiter. After processing, the surface tag set size was 989 items, much smaller than the abstract set due to the small amount of overlapping (which led to less “unique” labels). Adding the BIO tags conflicted with some internal features of some of the python libraries used in the experiments and therefore we changed the letters to A/X/D before piping the data into the model.

Conversely, the abstract node labels had significant overlap. See figure 2 for an example of the typical amount of overlap. This lead to a substantially larger label set size, approximately 7100 - making running experiments infeasible without a sizeable increase in amount of resources allocated. Thus, some strategies were employed to reduce the label set size. Following the technique used by [20], an encoding function was used to transform an abstract label

[1] https://gitlab.cs.uct.ac.za/jbuys/mrs-processing

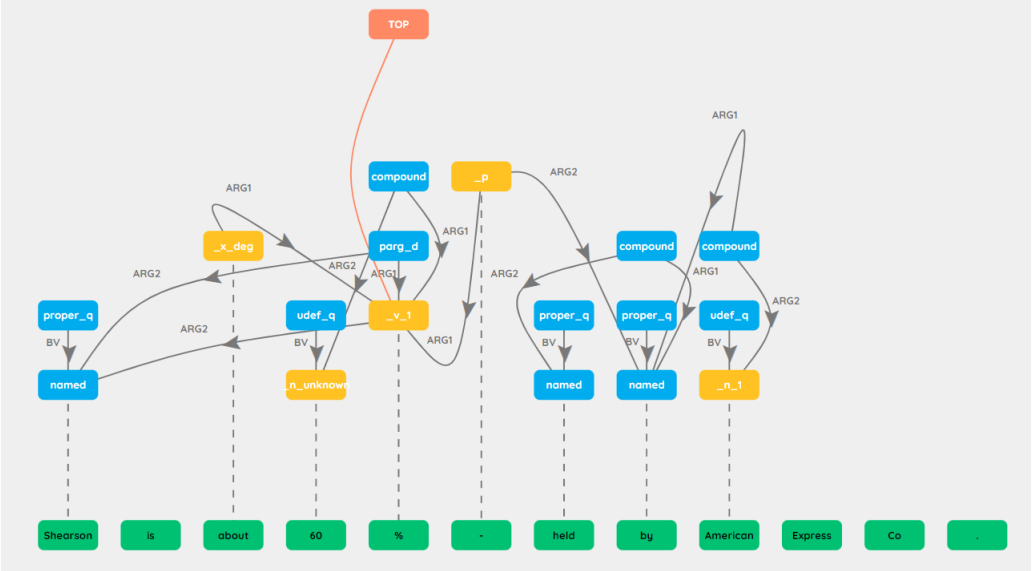

Figure 2: A visual representation of a sentence from the corpus - "Shearson is about 60%-held by American Express Co". This visualisation was created using RepGraph, a tool for visualising meaning representation graphs [12]. Abstract nodes are blue and surface nodes are yellow. Note how "*by*" is part of multiple spans - compound, proper_q and named. This overlap and multi-membership is typical for abstract nodes. Again, attention is drawn to the lack of overlap for surface nodes.

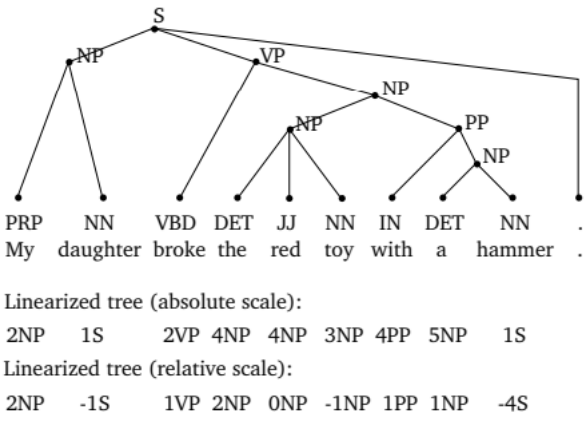

Figure 3: A sentence with labels is mapped to a tree format and then encoded to produce a set of labels [20]

for a given span into a function of the number of shared ancestors between it and another token. The input is transformed into a tree, then linearised into a sequence of labels. There were two types of possible encodings and we chose the relative scale as it allowed us to drastically reduce the label set size. Figure 3 shows how an input sentence with a set of non-EDS labels is encoded using both types of scales.

Though the authors of that paper created this approach for constituency parsing, it is still applicable in this use case. This allowed for the span information to be encoded into the tag itself. These labels are referred to as “tree2label” in table 1. This transformation resulted in a much smaller label set size, 3225 items.

5.3 Machine Learning Pipeline

A pipeline was implemented in order to facilitate the running and rerunning of experiments. Models were created and then trained, and checkpoints were implemented to ensure that model training could be paused and resumed if necessary. Once a model was trained with a minimum of 20 epochs, training vs validation loss plots were drawn up to determine if more epochs were necessary. These can be seen in the appendix. Once the loss plateaued, training was stopped. Thereafter, hyper-parameter tuning was conducted and the optimal learning rate was determined. Models with an additional CRF layer often required 5-8x as much time to train compared to their counterparts.

5.4 Data Post-Processing

Node prediction accuracy was tested using the development data set and a final evaluation was performed with the test set. The predictions of the model needed to be converted in order to determine what the predicted nodes were, as well as the predicted tags. For the predicted nodes, the tokens which were labelled "None" were removed from the set, however, they were retained in the predicted tag set.

In the case of the abstract models, the predicted node labels had to be converted into trees, and then back into a collection of spans with anchor (start and end) values and labels. A similar approach was used for the surface nodes, as the BIO tags were used to convert the model output and then removed before evaluation.Table 2 shows

Token	Surface	BIO Surface	Abstract	tree2label
Shearson	named	B-named	proper_q	1_S_proper_q
is	None	O-None	None	0_S
about	_x_deg	B-_x_deg	None	0_S
60%-	n_unknown	B-n_unknown	udef_q;compound	1_compound_udef_q
held	_v_1	B-_v_1	parg_d;compound	-1_S_parg_d
by	_p	B-_p	None	0_S
American	named	B-named	proper_q;compound;proper_q	2_compound_proper_q
Express	named	I-named	compound;proper_q	-1_proper_q
Co	_n_1	O-_n_1	udef_q;compound;proper_q	NONE_udef_q;compound

Table 1: A table showing how the sentence "Shearson is about 60%-held by American Express Co" is tokenised, as well as the surface and abstract node labels. Furthermore, it shows how those labels are transformed before they are fed into the neural network

Input	Consumers may want to move their telephones a little closer to the TV set.
Output	[0, 0, '_n_of', 'Consumers'], [1, 1, '_ˇmodal', 'may'], [2, 2, '_v_1', 'want'], [4, 4, '_v_cause', 'move'], [6, 6, '_n_1', 'telephones'], (7, 8, '_x_deg', 'a little'), [9, 9, '_a_to', 'closer'], [11, 11, '_q', 'the'], [12, 12, '_n_1', 'TV']

Table 2: Formatted I/O for a given input sentence. The first two digits represent the predicted start and end of a span, and the third element of the tuple is the label for the surface node.

an example of some formatted I/O from the Surface + BERT + No CRF model.

5.5 Experimental Setup

The Flair NLP [2] library `SequenceTagger` type was used to create all models. Initially, 8 models were envisioned:

(1) Abstract + Glove + No CRF
(2) Abstract + Glove + CRF
(3) Abstract + BERT + No CRF
(4) Abstract + BERT + CRF
(5) Surface + Glove + No CRF
(6) Surface + Glove + CRF
(7) Surface + BERT + No CRF
(8) Surface + BERT + CRF

All models were initially run with a nominal 5 epochs to determine basic parameters, such as batch size, batch chunk size (were necessary). Despite trialling various configurations, both Abstract models that included the CRF layer required more memory than was possible to provide. Thus, they were dropped from the experiment. All models employed a cross-entropy loss function and those that included a CRF used the Viterbi algorithm for decoding.

5.5.1 Baseline. The baseline for this experiment are the models with the standard Glove embeddings. Glove is an unsupervised learning algorithm used to produce vector representations of words [40]. Each node label is fed one at a time into this embedding layer before being passed to the BiLSTM layer. From there, it is passed through a softmax function and a predicted label is returned. When training the models with Glove, the embeddings were small enough to allow them to be stored on the GPU i.e. in CUDA memory. This made training with Glove substantially faster than BERT.

5.5.2 Computing Resources. All model training and hyper-parameter tuning was performed on the South African Department of Science and Innovations's Centre for High Performance Computing cluster. 1x Nvidia V100 16GB GPU and 20 CPUs were used for each model. Each node had 128 GiB of memory available.

5.5.3 Transformer Models. SpanBERT was the transformer selected for the models. This is the appropriate choice, as the method specialises in identifying relationships between two or multiple spans of text and out-performed BERT in terms of F1 scores [28]. The transformer was being fine-tuned during training - and due to this, the embeddings could not be stored. Instead, they were created on-the-fly in each mini-batch during training. This did keep the memory requirements lower but did increase training time.

Many transformers used subword tokenisation - for example "let's" would be split into $[let, ', s]$. This standards in contrast to typical tokenisation for English, which usually splits a character string based on whitespaces. To get a final token representation, there are a number of different pooling operations which are used on these subwords. For this experiment, *first_last* was chosen, which concatenates the embedding of the first and last subwords. Other options, such as first, last or mean are available and can potentially be explored in future work.

5.5.4 Hyper-Parameter Tuning. Hyper-parameter tuning was performed in order to find the optimal learning rate and dropout for each model. For the learning rate, the technique of cyclical learning rates, developed by [45] was employed. The learning rate is varied cyclically between an interval of values, using any optimiser of choice. In this research, AdamW [30] was chosen as it generally leads to faster convergence for models. The method is advantageous as it removes the requirement of "serially" tuning the learning rate by trialing a set of discrete values. A table with the final parameters, as well as learning rate curves for each model can be found in the appendix of this paper.

Dropout is a method used to prevent neural networks from overfitting[46]. Tuning for the optimal dropout is somewhat necessary

for this research, as it involves transformers, which are particularly prone to overfitting due to their over-parameterisation [51]. However, memory constraints prevented this from being performed. To counteract this, early stopping was employed once the loss became too small.

6 RESULTS AND DISCUSSION

A final evaluation was done using the test set of sentences and the results of the experiments are shown in tables 3 and 4. We begin by examining the metrics for each individual model, shown in table 3. The BERT surface models scored the highest across all categories, substantially outperforming the best Glove surface model by over 10% for the F1 score. Additionally, we note that the Glove surface model without the CRF layer performed marginally better than the model that did have it. The converse is true for the surface transformer models - BERT with a CRF only scored slightly higher across all categories.

The abstract models performed far worse than the surface ones. As can be seen in table 5, these two models required 3 to 4x the number of epochs to train. This may be due to the label set size being approximately 3x larger for the abstract nodes (compared to the surface nodes). Despite the extra number of epochs, their performance is lower for the concept identification metrics and tag accuracy. Glove's scores were far lower than BERT, particularly for concept identification. We postulate that BERT's better performance is due to it's ability to use context far better than the static embedding, and that it was able to fine-tune as it was training.

Finally, we move to the combined scores of the models, table 4. We note that again, both the BERT models outperform Glove by a substantial gap. BERT + CRF performed best overall, but the margin between it and the second best model is minimal at 0.21. Moreover, precision scores were slightly higher for all models.

We use the above results in order to answer our initial research questions:

(1) *How does a fine-tuned BERT encoder compare to an BiLSTM with a standard word-embedding layer in terms of accuracy of node prediction?*
Using the aforementioned tables, we see that BERT models scored much higher across these 3 metrics, with precision being slightly higher than recall. The F1 scores show a substantial gain with transformer embeddings.

(2) *To what extent does the addition of a CRF layer affect model accuracy, measured in terms of precision, recall and F1 score?*
We note that we were unable to train all 4 models that were slated to have a CRF. However, of the 2 that were able to be trained for surface concept identification, the additional CRF layer did not have a sizeable effect on node prediction accuracy. Thus, we conclude that having a CRF layer is not worth the extra computational costs.

6.0.1 Comparison To Other Work. Some of our results are analogous to that of [7]. In both cases, the models that utilised the transformer embeddings outperformed the static embeddings. Whilst [7]'s concept identification F1 score was slightly higher than their tagging accuracy score, the opposite was observed in our experiments. This may be due to an insufficient number of training epochs or sub-optimal hyper parameters. Moreover, [7] had marginally higher F1 scores for each model, 94 or greater, whereas our highest was 91.34. This may be due to our abstract models, which overall had lower scores compared to the surface models.

Type	Model	Tag	Concept		
		Accuracy	Prec.	Recall	F1
Surface	Glove + No CRF	0.85	0.83	0.83	82.93
	Glove + CRF	0.85	0.83	0.83	82.83
	BERT + No CRF	0.95	0.93	0.93	93.29
	BERT + CRF	0.95	0.94	0.94	93.63
Abstract	Glove + No CRF	0.70	0.71	0.68	69.37
	BERT + No CRF	0.88	0.88	0.87	87.83

Table 3: Table cataloguing separate metrics for each type of model. These provide a more in-depth look into how the different types of neural networks handle the different types of concept classes

		Concept		
Embedding	CRF?	Precision	Recall	F1
Glove	No	0.78	0.77	77.65
Glove	Yes	0.78	0.76	77.04
BERT	No	0.91	0.91	91.13
BERT	Yes	0.92	0.91	91.34

Table 4: Results for the node prediction of the full graph. The CRF column refers to the use of a CRF layer in the prediction of surface nodes

6.1 Limitations and Future Work

As mentioned in previous sections, this experiment leaves scope for future work. The abstract models performed comparatively worse to the surface ones. We propose that in-depth hyper-parameter tuning is necessary. These can include different hidden layer sizes, dropout, mini-batch size or number of RNN layers, for example. Additional epochs for training are also a possibility. Furthermore, Flair allows for embeddings to be combined - these stacked embeddings allow for contextualised and traditional embeddings to be used together. These are especially useful for sequence labelling problems, which proves relevant, given that concept identification can be framed as sequence labelling problem.

Additionally, concept prediction is one of two phases in the semantic graph-based parsing process. The models from this research have the potential to become parser of a larger pipeline that incorporates edge prediction models and whole-graph evaluation, which could lead to the development of a full data-intensive parser.

Some limitations of this work is that there is no guarantee of 100% accuracy with the use of the code created by [20]. The trees created using their code are not fully accurate in terms of representing the abstract nodes. More work needs to be done on this codebase to ensure that the parsing from a sentence with labels to a tree is precise and representative.

7 CONCLUSION

These experiments have demonstrated that BERT, a contextualised word representation model and transformers, surpasses the more "standard" Glove embeddings. CRF layers add additional model complexity and require more computational resources but are not worthwhile, given the minimal effect they had on node prediction accuracy. There is a lot of scope for future work with these experiments - including additional hyper-parameter tuning and the integration of these models into a large pipeline for a factorisation-based data-drive parser.

REFERENCES

[1] Omri Abend and Ari Rappoport. 2013. Universal Conceptual Cognitive Annotation (UCCA). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, 228–238. https://aclanthology.org/P13-1023

[2] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. 54–59.

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

[4] 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*. Association for Computational Linguistics, Sofia, Bulgaria, 178–186. https://aclanthology.org/W13-2322

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

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

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

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

[9] Yufei Chen, Weiwei Sun, and Xiaojun Wan. 2018. Accurate SHRG-Based Semantic Parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, 408–418. https://doi.org/10.18653/v1/P18-1038

[10] Yufei Chen, Yajie Ye, and Weiwei Sun. 2019. Peking at MRP 2019: Factorization- and Composition-Based Parsing for Elementary Dependency Structures. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*. Association for Computational Linguistics, Hong Kong, 166–176. https://doi.org/10.18653/v1/K19-2016

[11] 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.

[12] Jaron Cohen, Roy Cohen, Edan Toledo, and Jan Buys. 2021. RepGraph: Visualising and Analysing Meaning Representation Graphs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 79–86. https://doi.org/10.18653/v1/2021.emnlp-demo.10

[13] Ann Copestake. 2009. **Invited Talk:** Slacker Semantics: Why Superficiality, Dependency and Avoidance of Commitment can be the Right Way to Go. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*. Association for Computational Linguistics, Athens, Greece, 1–9. https://aclanthology.org/E09-1001

[14] Nicki Detlefsen, Jiří Borovec, Justus Schock, Ananya Jha, Teddy Koker, Luca Di Liello, Daniel Štancl, Quan Changsheng, Maxim Grechkin, and William Falcon. 2022. TorchMetrics -Measuring Reproducibility in PyTorch. *The Journal of Open Source Software* 7 (02 2022). https://doi.org/10.21105/joss.04101

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[16] Timothy Dozat and Christopher D. Manning. 2018. Simpler but More Accurate Semantic Dependency Parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Melbourne, Australia, 484–490. https://doi.org/10.18653/v1/P18-2077

[17] Kawin Ethayarajh. 2019. How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 55–65. https://doi.org/10.18653/v1/D19-1006

[18] Federico Fancellu, Sorcha Gilroy, Adam Lopez, and Mirella Lapata. 2019. Semantic graph parsing with recurrent neural network DAG grammars. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 2769–2778. https://doi.org/10.18653/v1/D19-1278

[19] Dan Flickinger, Emily M. Bender, and Stephan Oepen. 2014. Towards an Encyclopedia of Compositional Semantics: Documenting the Interface of the English Resource Grammar. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. European Language Resources Association (ELRA), Reykjavik, Iceland, 875–881. http://www.lrec-conf.org/proceedings/lrec2014/pdf/562_Paper.pdf

[20] Carlos Gómez-Rodríguez and David Vilares. 2018. Constituent Parsing as Sequence Labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 1314–1324. https://doi.org/10.18653/v1/D18-1162

[21] 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.

[22] 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*.

[23] Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly Predicting Predicates and Arguments in Neural Semantic Role Labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Melbourne, Australia, 364–369. https://doi.org/10.18653/v1/P18-2058

[24] 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.

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

[26] 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.

[27] Zhengbao Jiang, Wei Xu, Jun Araki, and Graham Neubig. 2020. Generalizing Natural Language Analysis through Span-relation Representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 2120–2133. https://doi.org/10.18653/v1/2020.acl-main.192

[28] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving Pre-training by Representing and Predicting Spans. *Transactions of the Association for Computational Linguistics* 8 (2020), 64–77. https://doi.org/10.1162/tacl_a_00300

[29] 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.

[30] Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations* (12 2014).

[31] Artur Kulmizev, Miryam de Lhoneux, Johannes Gontrum, Elena Fano, and Joakim Nivre. 2019. Deep Contextualized Word Embeddings in Transition-Based and Graph-Based Dependency Parsing - A Tale of Two Parsers Revisited. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 2755–2768. https://doi.org/10.18653/v1/D19-1277

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

[33] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML '01)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA,

282–289.
[34] Elizabeth D Liddy. 2001. Natural language processing. (2001).
[35] H.-L. Lou. 1995. Implementing the Viterbi algorithm. *IEEE Signal Processing Magazine* 12, 5 (1995), 42–52. https://doi.org/10.1109/79.410439
[36] 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
[37] Stephan Oepen, Dan Flickinger, Kristina Toutanova, and Christopher D Manning. 2004. Lingo redwoods. *Research on Language and Computation* 2, 4 (2004), 575–596.
[38] Stephan Oepen and Jan Tore Lønning. 2006. Discriminant-based MRS banking.. In *LREC*. Citeseer, 1250–1255.
[39] Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep Multitask Learning for Semantic Dependency Parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, 2037–2048. https://doi.org/10.18653/v1/P17-1186
[40] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543. http://www.aclweb.org/anthology/D14-1162
[41] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, 2227–2237. https://doi.org/10.18653/v1/N18-1202
[42] 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.
[43] 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.
[44] Helmut Schmid. 1994. Part-of-Speech Tagging With Neural Networks. In *COLING 1994 Volume 1: The 15th International Conference on Computational Linguistics*. Kyoto, Japan. https://aclanthology.org/C94-1027
[45] Leslie N. Smith. 2017. Cyclical Learning Rates for Training Neural Networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 464–472. https://doi.org/10.1109/WACV.2017.58
[46] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.
[47] 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).
[48] 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.
[49] 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).
[50] 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.
[51] Zhen Wu, Lijun Wu, Qi Meng, Yingce Xia, Shufang Xie, Tao Qin, Xinyu Dai, and Tie-Yan Liu. 2021. UniDrop: A Simple yet Effective Technique to Improve Transformer without Extra Cost. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Online, 3865–3878. https://doi.org/10.18653/v1/2021.naacl-main.302
[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, 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

A LEARNING RATE CURVES

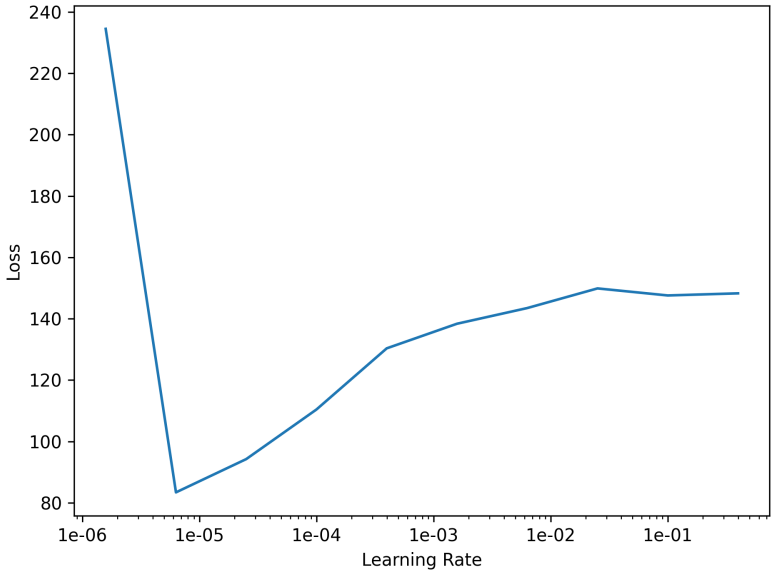

Figure 4: Abstract + Glove

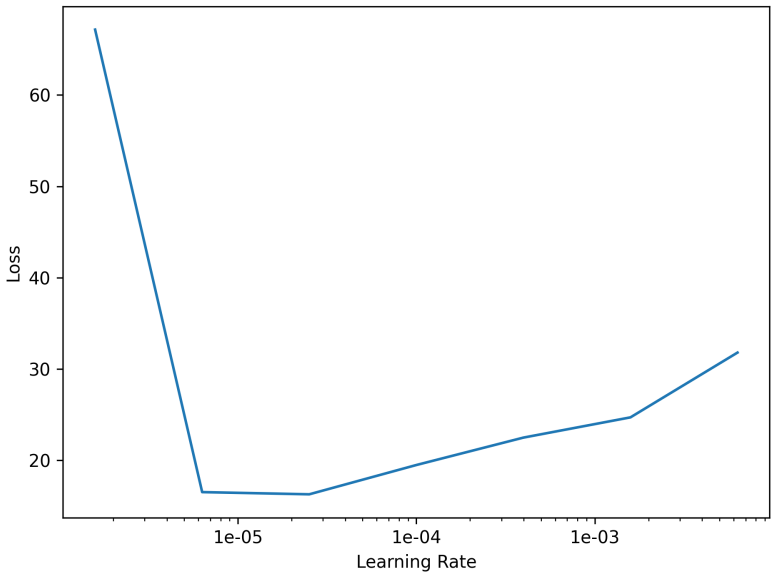

Figure 5: Abstract + BERT

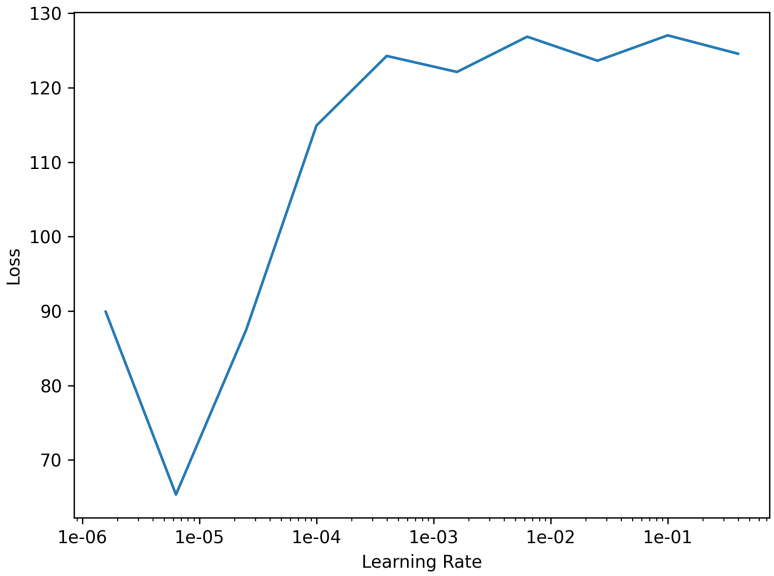

Figure 6: Surface + Glove

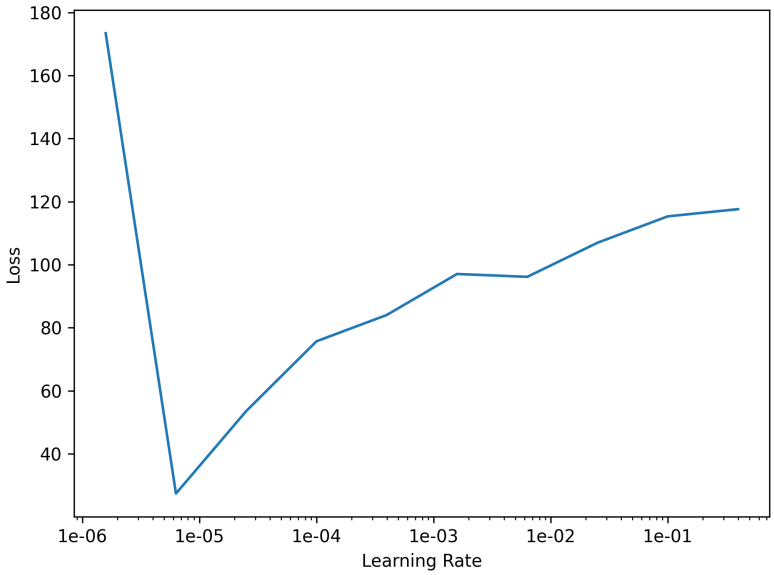

Figure 7: Surface + Glove + CRF

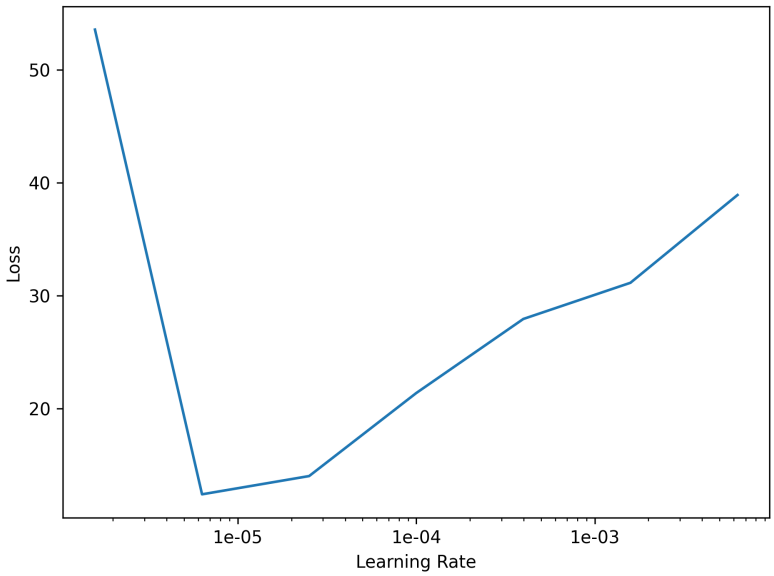

Figure 8: Surface + BERT

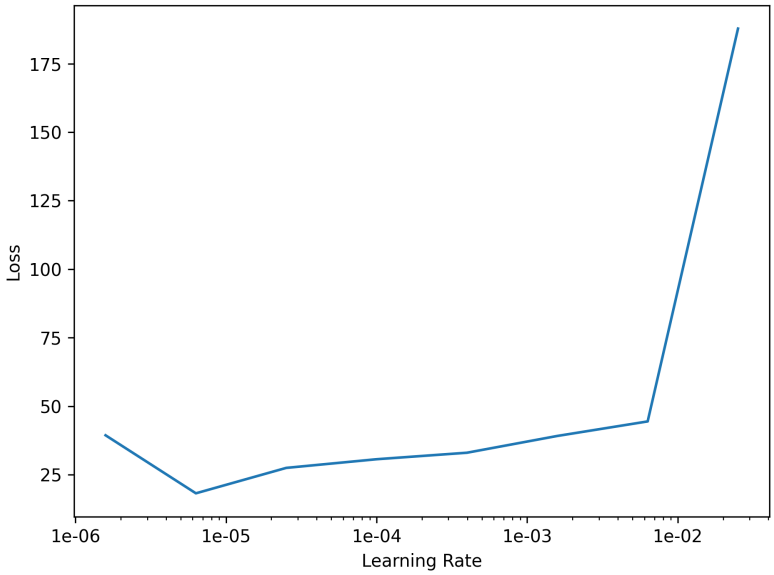

Figure 9: Surface + BERT + CRF

B TRAINING VS VALIDATION LOSS

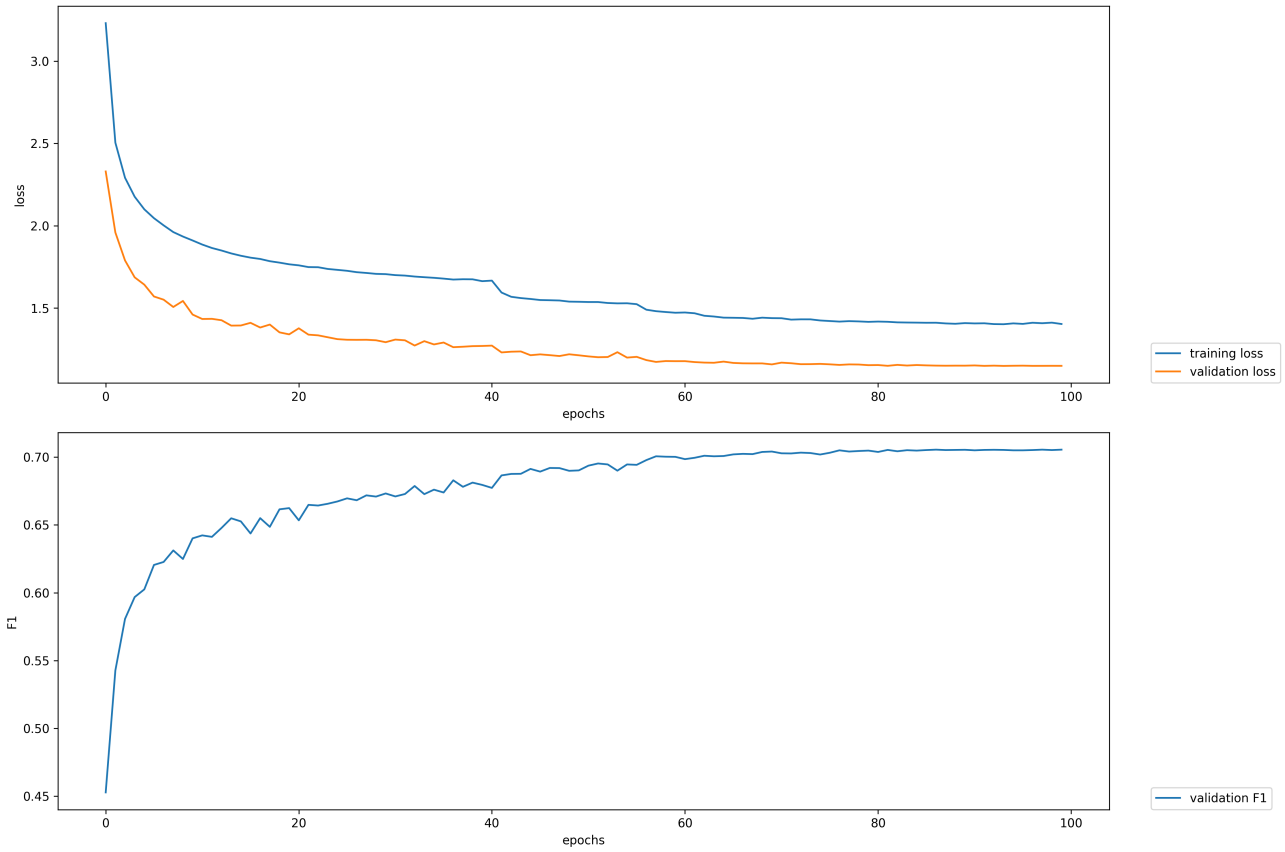

Figure 10: Abstract + Glove

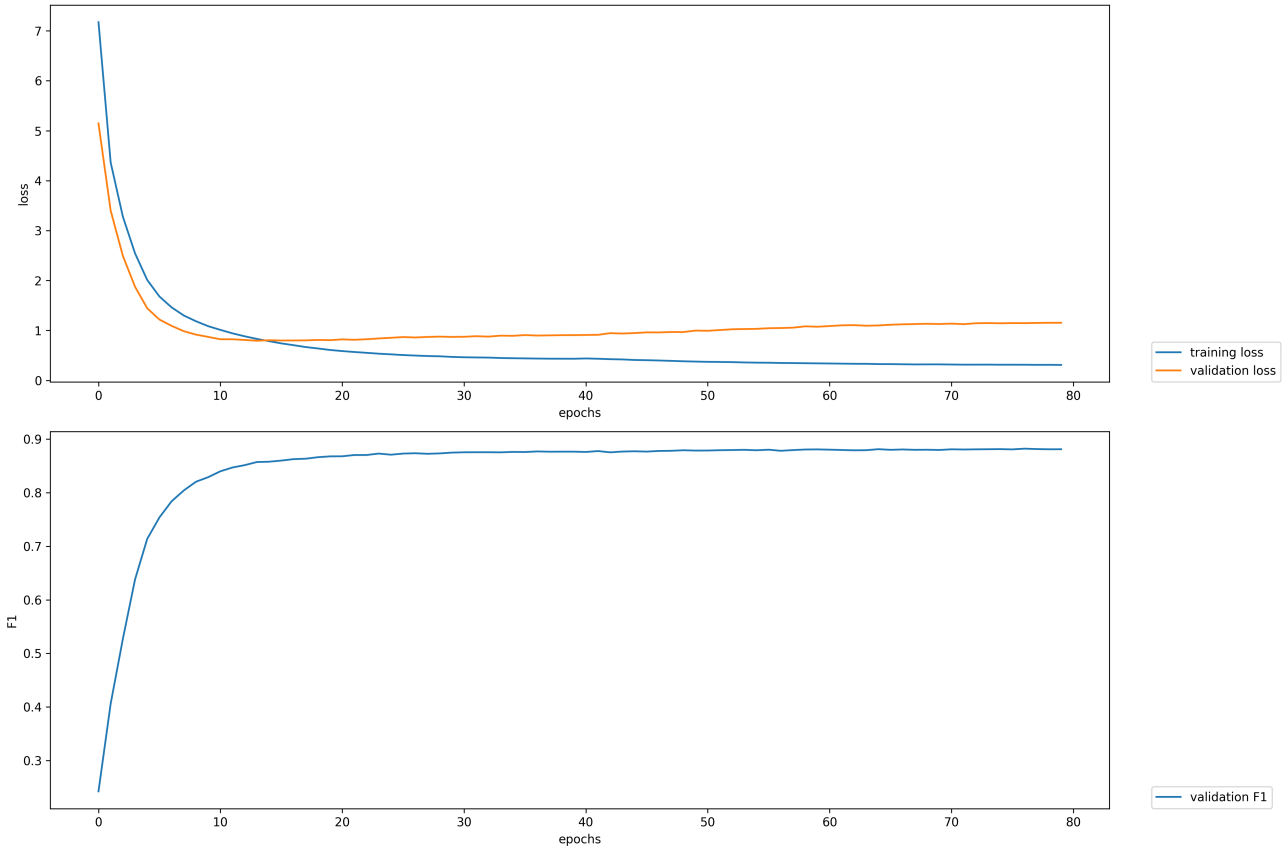

Figure 11: Abstract + BERT

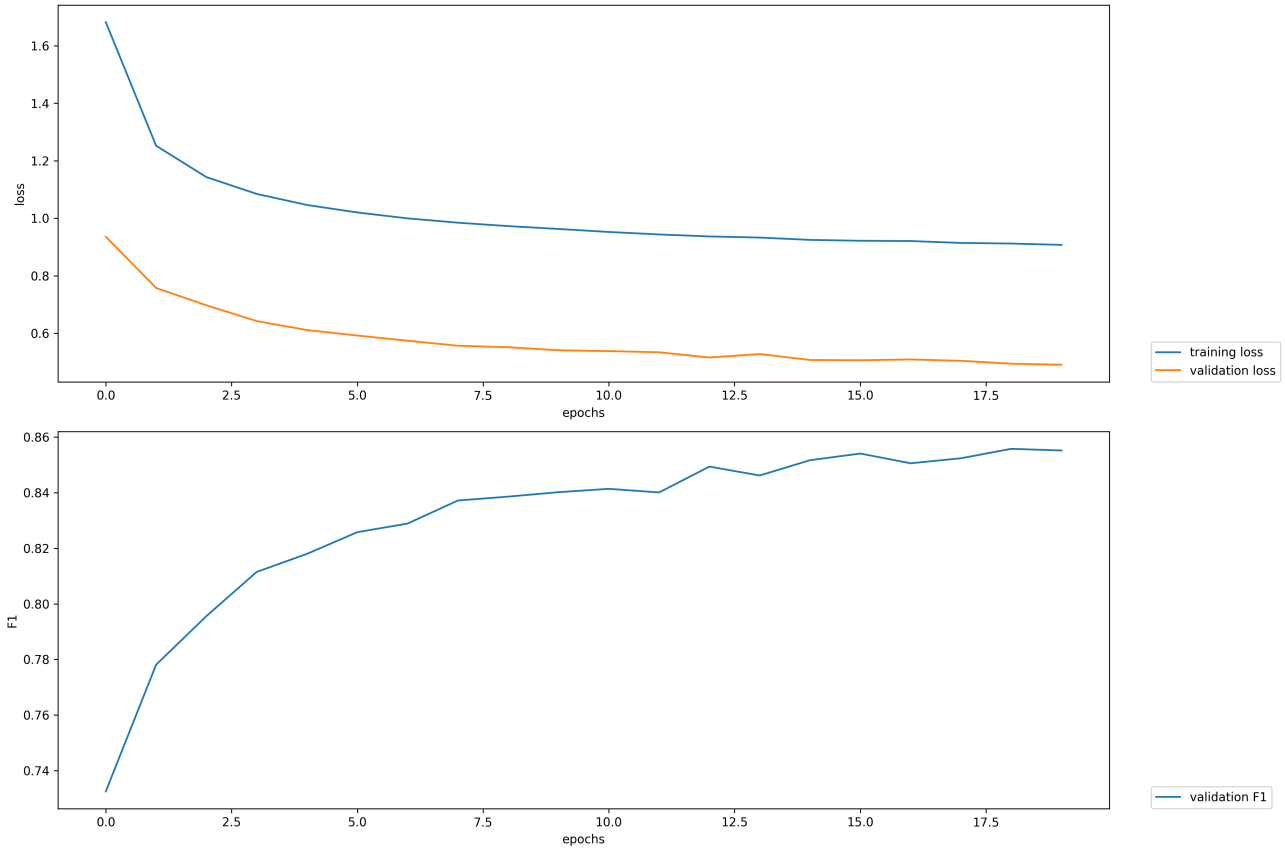

Figure 12: Surface + Glove

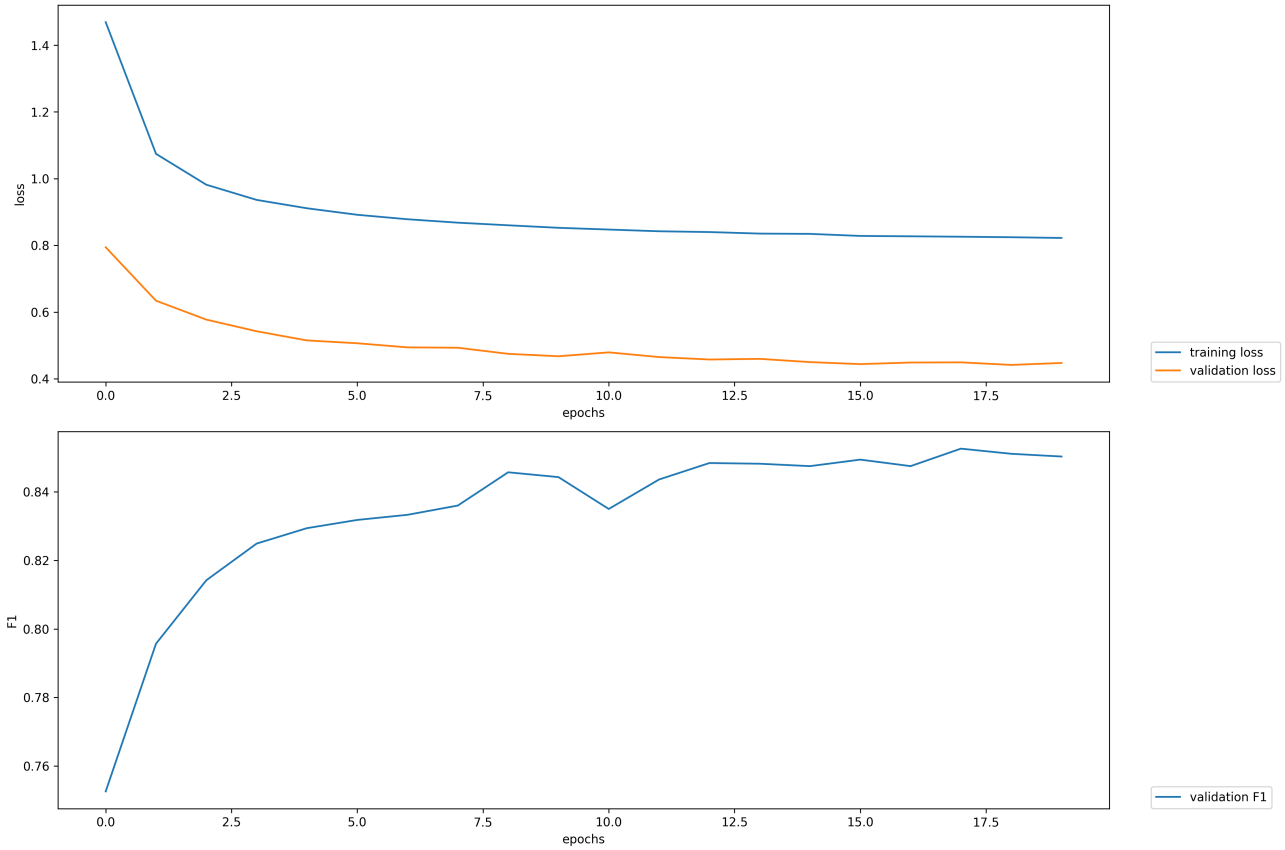

Figure 13: Surface + Glove + CRF

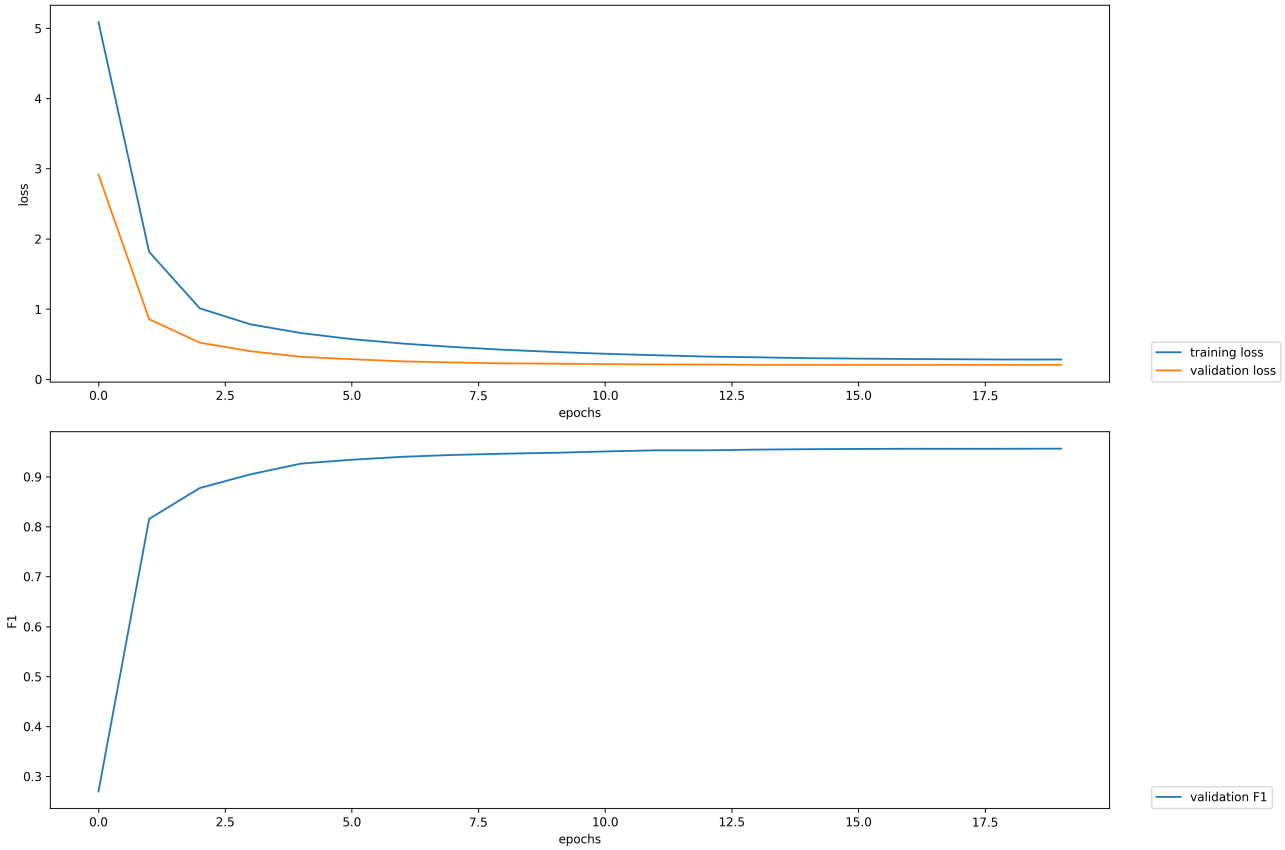

Figure 14: Surface + BERT

C MODEL PARAMETERS

Model	Parameters	Value
Surface + Glove + No CRF	Optimiser	AdamW
	Epochs	20
	Learning Rate	6e − 06
	Word Dropout	0.05
Surface + Glove + CRF	Optimiser	AdamW
	Epochs	20
	Learning Rate	6e − 06
	Word Dropout	0.05
Surface + BERT + No CRF	Optimiser	AdamW
	Epochs	20
	Learning Rate	6e − 06
	Word Dropout	0.05
Surface + BERT + CRF	Optimiser	AdamW
	Epochs	
	Learning Rate	6e − 06
	Word Dropout	0.05
Abstract + Glove + No CRF	Optimiser	AdamW
	Epochs	100
	Learning Rate	6e − 06
	Word Dropout	0.05
Abstract + BERT + No CRF	Optimiser	AdamW
	Epochs	80
	Learning Rate	6e − 06
	Word Dropout	0.05

Table 5: Final Model Parameters