University of Amsterdam

Master Thesis

Task-Based Semantic Evaluation of Parse Tree Discontinuity

Author:
Stijn de Gooijer

Supervisor:
Dr. W.H. Zuidema

A thesis submitted in fulfilment of the requirements
for the degree of Master of Science in Artificial Intelligence

September 2016

UNIVERSITY OF AMSTERDAM

Abstract

Task-Based Semantic Evaluation of Parse Tree Discontinuity

by Stijn de Gooijer

The concept of discontinuity has historically been addressed in parsing frameworks in a number of different ways. Some parsing frameworks produce discontinuous parse trees, while others rely on feature passing in continuous parse trees in order to account for discontinuity. In this research, we evaluate the influence of discontinuity by comparing continuous and discontinuous parse trees. We explore two methods of task-based evaluation, one based on dynamic semantics and one based on tree-based neural networks, of which the latter seems most capable of evaluating parse tree discontinuity. We end up running two semantic classification tasks: sentiment analysis and question classification. A tree-based long short-term memory model is trained based on discontinuous parse trees, as well as based on continuous parse trees. Classification accuracy is then compared to a baseline of left-branching parse trees and to each other. We find that in the sentiment analysis task, the discontinuous model has the advantage on the part of the dataset containing discontinuities. In the question classification task, we find that both the continuous and the discontinuous model have their strengths: the continuous model performs better on the fine classification task, while the discontinuous model performs better on the coarse classification task. We conclude that, while it is not always necessary to model discontinuity, handling discontinuity with the use of discontinuous constituents does improve semantic task performance in a number of cases.

Acknowledgements

I would like to thank Jelle Zuidema, for his insights and supervision. I would also like to thank Andreas van Cranenburgh and Phong Le for providing me with software and support for running my experiments. Finally, I would like to thank all members of the Cognition, Language, and Computation research group for listening to my ideas and giving feedback.

Contents

Chapter 1

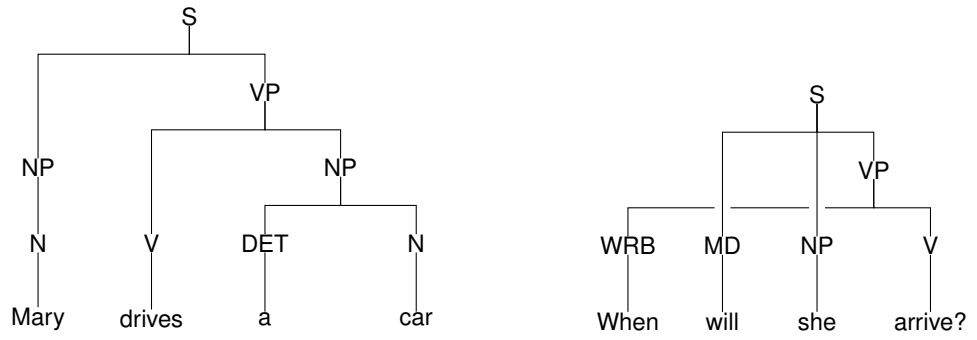
Introduction

Natural language processing algorithms play an important role in making information universally accessible. If we visit a website that is not available in our native language, our browser offers us a convenient button that says "Translate this website to your own language". Speech recognition algorithms allow us to input text by just using our voice, and speech synthesis algorithms allow text to be read back to us over our computer's speakers.

In order for a computer to be able to carry out these tasks, it needs to have some sort of understanding of the language it is processing. Finding the right translation of a sentence requires careful selection of words and phrases to preserve its original meaning. Correctly recognizing a sentence produced by a human voice requires knowledge of context to disambiguate between phrases that sound identical but with different textual representation. Synthesizing speech requires discourse information to identify the correct words and syllables to stress. Many tasks related to natural language require advanced understanding of the language to be performed optimally.

There are many techniques to help natural language processing (NLP) algorithms to achieve this advanced understanding of language. One technique that is widely used among NLP tasks is *parsing*. Parsing is the process of identifying the syntactic structure of a sentence, usually in the form of a constituency-based parse tree or dependency-based parse tree (see Figure 1.1).

These parse trees can take on different forms based on the grammar formalism used for parsing, which in turn may affect how well NLP algorithms perform certain tasks. In this thesis, we will research different types of parse trees through their performance in specific NLP tasks.



(a) Continuous (b) Discontinuous

Figure 1.1: Examples of constituency-based parse trees.

1.1 Thesis Contributions

This thesis will focus on comparing two types of parse trees: continous parse trees and discontinuous parse trees. Continuous parse trees are parse trees which contain only continuous constituents, usually parsed using a context-free grammar formalism. Figure 1.1a shows an example of such a parse tree. Discontinuous parse trees allow for discontinuous constituents, an example of which can be seen in Figure 1.1b. As can be seen from the figure, these discontinuous constituents lead to crossing branches in the parse tree.

There are two key differences between the two parse tree types: expressiveness and parsing complexity. Discontinuous parse trees have more expressive power, as discontinuous constituents allow the modeling of certain linguistic phenomena that are impossible to model using continuous constituents. However, this comes at the cost of higher parsing complexity, as the parsing algorithms have to account for possible discontinuities.

Historically, there has been relatively little attention for discontinuous parse trees, compared to continuous parse trees. Several parsing frameworks have been developed for discontinuous parsing ([1], [2], [3]), and parsing results have been compared to parse results from continuous parsing frameworks. However, this is usually done based on bracketing precision and recall, which does not tell us much, since the parse tree structures are of a different nature.

In this thesis, we evaluate the influence of discontinuity, by using continuous and discontinuous parse trees as part of a semantic task and measuring task performance. For this evaluation, we combine existing techniques that have recently been developed. The contribution of this thesis is thus the application of previous research in a novel way, in order to provide new insight in an area in which relately little research has been done.

The evaluation method outlined in this thesis makes it possible to evaluate discontinuity a in meaningful way: difference in task performance between parse tree types will indicate how much discontinuity contributes to helping the NLP algorithm better 'understand' the text. And this, in the end, is the real goal of parsing text.

1.2 Thesis Outline

This thesis is structured as follows:

- In Chapter 2, we provide background on the concept of discontinuity. We discuss linguistic phenomena linked to discontinuity, and discuss associated grammar formalisms and parsing algorithms. We also introduce dynamic semantics and tree-based neural networks as possible approaches for evaluating the influence of discontinuity through NLP tasks.
- In Chapter 3, we explore how dynamic semantics could be utilized for evaluating grammar formalisms. We discuss Discourse Representation Theory and the role of parse trees in constructing semantic representations of sentences. We also explore possible NLP tasks based on dynamic semantics for evaluating discontinuity.
- In Chapter 4, we explore tree-based neural networks, focusing on the Long Short-Term Memory model. We discuss word embeddings, recurrent and recursive neural networks, and the role of parse trees in these models. We also propose possible NLP tasks for evaluating discontinuity based on neural network models.
- In Chapter 5, we describe in detail two NLP tasks: sentiment analysis and question classification. We report experimental setup and results, and discuss the results in-depth.
- In Chapter 6, we arrive at a conclusion for this research. We summarize the contributions of the research, and list possible future work on the subject.

Chapter 2

Discontinuity in Parse Trees

In this chapter, we provide background on the central topic of this thesis: parse tree discontinuity. We first explain the general concept, and the linguistic context that gives rise to the need for discontinuity. We do so by discussing natural language examples.

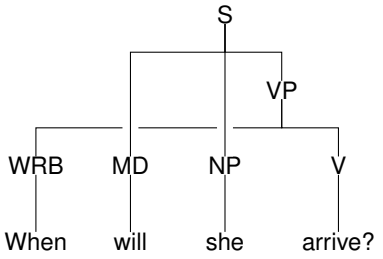
Secondly, we discuss how these discontinuous parse trees may be produced. We discuss indirect methods such as traces and feature passing, as well as direct parsing methods involving mildly context-sensitive grammars.

Finally, we take a closer look at why continuous parse trees are so heavily favoured in state-of-the-art NLP algorithms. We explore different ways to compare continuous and discontinuous syntax structures. Most importantly, we discuss the use of formal semantics and neural networks for task-based evaluation.

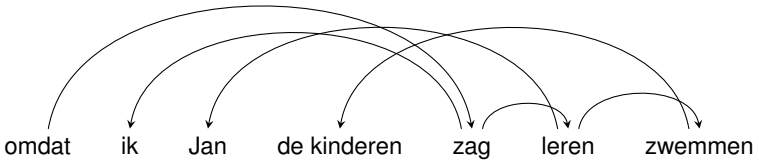
2.1 Linguistic Motivation

We may define discontinuity as follows: discontinuity is the phenomenon that occurs when two words or phrases that are directly related are separated by at least one other word or phrase. This will result in a tree structure with crossing branches. An example of a discontinuous constituency tree is shown in figure 2.1a. In this case, the phrases "when" and "arrive" form a verb phrase together, however, the phrases occur on opposite sides of the sentence. As you can see, this leads to crossing branches. Discontinuity also presents itself in dependency trees, as shown in Figure 2.1b, although much less frequently.

Discontinuities occur in most natural languages, if not all. However, there are different types of discontinuity, and not every language contains every type of discontinuity. The



(a) Wh-movement in a constituency-based parse tree



(b) Dutch cross-serial dependency in a dependency-based parse tree

Figure 2.1: Examples of discontinuous parse trees.

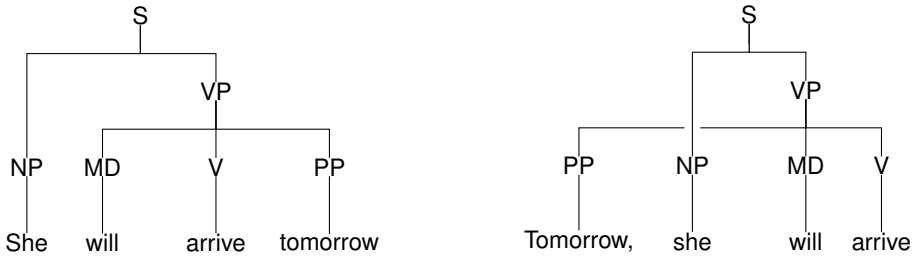
most prominent types of discontinuity are wh-movement, topicalization, extraposition, and scrambling.

The first type, *wh-movement*, can be seen in the example given previously (Figure 2.1a). Wh-movement appears most commonly in questions. The name wh-movement comes from the discontinuity that often coincides with interrogative words, which in English often start with *wh-* (*what*, *who*, *which*, etc.). This type of discontinuity often occurs in direct and indirect questions and in relative clauses. Example 2.1 below shows the sentence as a statement, while 2.2 shows the sentence in question form, which causes wh-movement to occur.

She will arrive **tomorrow**. (2.1)

When will she arrive? (2.2)

The second type, *topicalization*, occurs when a word or phrase is fronted in order to establish it as the topic of the sentence. An example of topicalization is shown below. In general, phrases that are subject to wh-movement are also subject to topicalization. Figure 2.2 shows the parse tree of the sentence before and after topicalization takes place.



(a) Before topicalization (b) After topicalization

Figure 2.2: Parse trees showing discontinuity as a result of topicalization.

She will arrive **tomorrow**. (2.3)

Tomorrow, she will arrive. (2.4)

Extraposition occurs when an expression appears further to the right of where it would normally appear. A common occurrence is the it-extraposition, in which case the word "it" appears as a placeholder for a phrase that appears later in the sentence.

That it rained occurred to me. (2.5)

It occurred to me **that it rained**. (2.6)

Scrambling occurs in languages that allow freedom in word order. Word order in English is quite strict, but in other languages like German, scrambling discontinuities are common. An example is given below, in which the sentence object may appear either before or after the past participle.

Ich habe versucht **dich** zu verstehen. (2.7)

Ich habe **dich** versucht zu verstehen. (2.8)

Note that discontinuity is not limited to the four types described above. Depending on the language, there may be many more cases in which discontinuity appears. Typically, about 15 to 25% of sentences ([4]) contain a discontinuity, with the rest being continuous. Freedom in word order is generally the determining factor in the prevalence of discontinuity, with English falling on the low end of the spectrum.

2.2 Parsing Discontinuities

Now that we have established some linguistic context, let us consider how to arrive at a discontinuous parse tree when presented with a sentence that calls for a discontinuous constituent.

Context-free grammars (CFGs) are a popular grammar formalism for parsing, as there is an efficient parsing algorithm available which parses in time complexity $\mathcal{O}(n^3)$ ([5]). However, CFGs cannot be used to generated discontinuous parse trees directly, since production rules do not allow for 'gaps' between right-hand side constituents. Traditionally, this problem has been solved by applying either syntactic movement or feature passing.

The theory of *syntactic movement*, as often applied in transformational grammars, assumes that a sentence is generated in its canonical form, and then movement takes place in order to account for discontinuities as mentioned in the previous section. The contituents moved in this way leave behind a trace of some sort in their original location, with certain linguistic functions of their own. An example is shown below. These traces are present in some state of the art treebanks, such as the Penn Treebank ([6]). *Feature passing* does not assume any movement, but rather assumes that information is passed through the rest of the tree to connect the 'moved' constituent and its original parent.

You mean **what**? (2.9)

What do you mean t? (2.10)

There are, however, a number of discontinuity types that cannot be generated by context-free grammars, even with the aforementioned mentioned techniques. This was first confirmed by [7], proving that cross-serial dependencies in Swiss-German cannot be generated by a context-free grammar. An example of cross-serial dependency in Dutch, which was also analyzed in [8], is shown in Figure 2.3.

In order to account for these types of discontinuities, we need to move to grammars that are not context-free. The class of grammar formalisms that can deal with these kinds of discontinuities, while still being able to produce parse trees in polynomial time, is called *mildly context-sensitive*. Examples of mildly context-sensitive grammars include Tree Adjoining Grammars (TAG) ([9], [10]) and Linear Context-Free Rewriting Systems (LCFRS) ([11], [12], [1]).

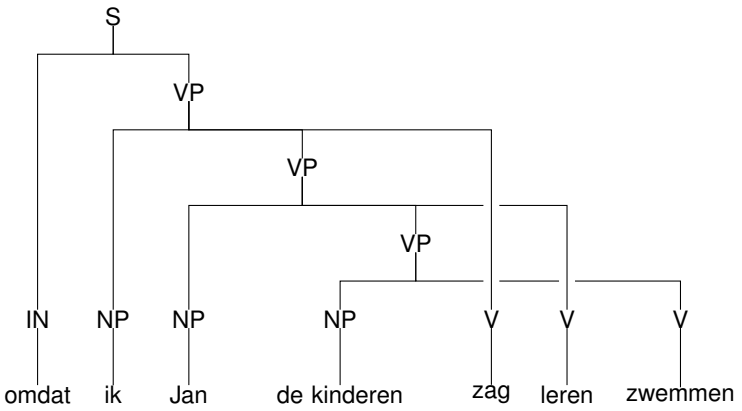


Figure 2.3: Dutch cross-serial dependency in a constituency-based parse tree.

The way this mild context-sensitivity is achieved differs per formalism. TAG allows replacing not only leaves of trees with new trees, but also replacing internal nodes. LCFRS allows production rules to yield not just strings, but tuples of non-adjacent strings. Most importantly, these formalisms allow the parsing of discontinuous constituents *directly*. The result is that these formalisms can generate all types of discontinuity, including cross-serial dependencies.

And the advantage of directly parsing discontinuities goes beyond the ability to generate certain obscure linguistic phenomena. Grammar formalisms like LCFRS may also lead to a much more concise set of production rules, when compared to a CFG. In CFGs, depending on the way discontinuities are handled, the number of production rules may 'explode': for every discontinuity, a continuous production rule needs to account for all possible phrase structures that could occur in the gap. In LCFRS, a single production rule will suffice to handle a discontinuity.

The mildly context-sensitive grammar formalisms come with a serious drawback though, in the form of parsing complexity. While parsing with context-free grammars may be done in time complexity $\mathcal{O}(n^3)$, parsing algorithms for TAG have time complexity $\mathcal{O}(n^6)$ and for LCFRS $\mathcal{O}(n^{3k})$ (where n is the length of the input sentence, and k is the fan-out)([13]).

2.3 Evaluating the Influence of Discontinuity

As we have seen in the previous section, there are many ways to deal with discontinuity when parsing. Context-free grammars deal with discontinuity using movement or feature passing, while mildly context-sensitive grammars are able to deal with discontinuity in a more principled way by parsing discontinuities directly.

In the general field of linguistics, context-free grammars are still used much more commonly than mildly context-sensitive grammars. One of the reasons for this is the previously mentioned computational complexity of parsing, which greatly favors context-free grammars. Another important consideration is the availability of treebanks: while theory on discontinuity has been around for a long time, many treebanks do not contain discontinuous trees, nor the traces to reconstruct discontinuities. This severely limits the availability of discontinuous trees, and consequently the ability to train high quality discontinuous parsers. Discontinuous treebanks do exist, for example the TIGER[1] ([14]) and NEGRA[2] ([15]) treebanks for German. However, generally a lot more research seems to be done for the English language, in which discontinuity is less prevalent, thus reducing the urgency to properly deal with discontinuity.

The choice for context-free grammars appears unfortunate, since we already know that CFGs cannot deal with all linguistic phenomena. In the future, we would like to be able to perfectly parse every language. And for this, we will need to move beyond context-free grammars. In an effort to support such a transition, more research is required on the subject of discontinuity. In this thesis, we contribute by testing different types of trees in a task-based environment.

While direct parsing tasks may seem like a more suitable way to compare syntax structures, the different nature of continuous and discontinuous parse trees makes it impossible to directly and fairly compare the two. We can, however, define a task that utilizes parse trees, and measure task performance. For example, a text classification task like sentiment analysis may utilize parse trees in the classification process. Performing the task once with continuous trees, and then once with discontinuous trees, will objectively show possible differences between the parse tree types.

We explore two approaches to task-based evaluation: one based on dynamic semantics, and one based on tree-based neural networks. *Dynamic semantics* allows us to construct a logic representation of a sentence, with the help of tree structures. In a classification task, we can determine if the logic representation of the sentence is constructed correctly through various tree structures. This approach is explored in Chapter 3.

Using *tree-based neural networks*, we may perform basically any text classification task. Word vectors are propagated up through the tree, resulting in a single vector representation of the sentence, which may then be mapped to a class. The structure of the tree affects the order in which word vectors are combined, which leads to a different sentence

[1]The TIGER Treebank is available at the following link: `http://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/tiger.en.html`

[2]The NEGRA corpus is available at the following link: `http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/negra-corpus.html`

vector, with possibly a different class as a result. If we take the task of sentiment analysis, for example, the vector may be mapped to a sentiment class. The tree-based neural network approach is explored further in Chapter 4.

Chapter 3

Dynamic Semantics

In this chapter, we explore how the influence of discontinuity may be evaluated through dynamic semantics. First, we explain Discourse Representation Theory (DRT), which is the semantic representation we will be working with. We discuss the benefits of this semantic representation and how such a representation is constructed for natural language sentences.

Second, we explore how we may define a task based on tree structures and DRT, that will help evaluate the influence of discontinuity. In doing so, we will discuss available datasets, possible algorithms, and related theory.

3.1 Discourse Representation Theory

Discourse Representation Theory is a theory of dynamic semantics. Dynamic semantics was developed as an adaptation of formal semantics, aiming to account for context dependence of meaning. The DRT framework was first proposed by [16], and has since been joined by other dynamic semantic theories like dynamic Montague grammars ([17]) and dynamic predicate logic ([18]). An important change from formal semantics is the shift towards information as the central concept, rather than truth values.

The main advantage of DRT is how well it handles inter- and intrasential anaphoric relations between indefinite NPs and personal pronouns. Consider the natural language example below, followed by its formal semantic representation.

A student participated. He won. (3.1)

$$\exists x[student(x) \wedge participate(x) \wedge win(x)] \tag{3.2}$$

While the formal semantic representation is correct, the process of generating this representation requires treating the full stop between sentences as a conjuction, which is not correct in all cases. Thus, formal semantics fails to account for these sentences in a principled way. DRT handles this by representing semantics in two parts: the universe, which consists of a set of discourse referents, and the conditions, which is a set of conditions. While a DRT representation of a sentence can thus be represented as a tuple of sets (3.3), a "box notation" is often used for readability (3.4).

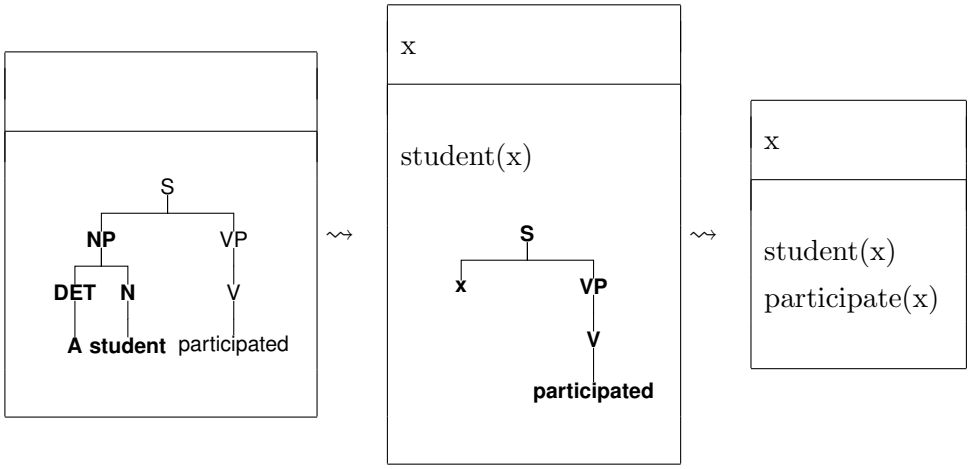
$$\langle \{x, y\}, \{student(x), participate(x), win(y), x = y\} \rangle \tag{3.3}$$

x, y
student(x) participate(x) win(y) x = y

(3.4)

There are two main approaches for constructing such a Discourse Representation Structure (DRS): top-down and bottom-up construction. Both construction methods are based on parse trees. DRS construction has been defined for many grammar formalisms, including CFG ([16]) and categorial grammars ([19]), but to our knowledge has not yet been defined for grammar formalisms with discontinuous constituents like LCFRS.

The top-down approach is based on a number of construction rules, which match certain parts of a parse tree and converts those into discourse referents and/or conditions. An example is shown below. Subsequent sentences are constructed within the context of the previous DRS, allowing personal pronouns to bind to discourse referents defined in the previous sentence.



(3.5)

The bottom-up approach is a more modern DRS construction algorithm, and is detailed in [20]. The approach breaks DRS construction into two phases: constructing a preliminary sentence representation, and verifying and merging presuppositions. The preliminary representation is constructed again based on a parse tree, but this time starting by assigning semantic representation to the leaves of the tree and combining these while going up the tree. This bottom-up approach is generally preferred over the top-down approach, since it handles presuppositions more successfully.

3.2 Evaluating Discontinuity through DRS Construction

As detailed in the previous section, parse trees are a native part of the DRS construction process, both in the top-down and bottom-up approaches. Evaluation of discontinuity could take place based on the accuracy of different parse tree types in creating a correct DRS representation. After all, the success of DRS construction is heavily dependent on the quality of the underlying parse tree ([21]). We describe two possible tasks, one based on the top-down approach, and one based on the bottom-up approach.

For the top-down approach, a classification task could be defined, comparing constructed DRS representations with a golden standard as defined in a DRS-analyzed dataset like the Groningen Meaning Bank [22][1]. Measuring classification accuracy would then provide insight in the ability of different grammar formalisms to correctly produce a semantic analysis of a sentence.

One roadblock for this approach is the fact that DRS construction is currently undefined for grammar formalisms with discontinuous constituents. If we would like to perform

[1]The Groningen Meaning Bank is freely available for download at the following link: `http://gmb.let.rug.nl/`

the experiment as mentioned above, we would need to create a set of DRS construction rules for a grammar formalism like LCFRS. Additionally, since construction rules are dependent on parse tree labels, in addition to its structure, the experiment does not purely evaluate discontinuity. Rather, it evaluates grammar formalisms in general.

It is possible, however, to define an algorithmm for DRS construction that does not rely on parse tree node labels, but relies only on parse tree structure. In the bottom-up construction approach, semantic representation of child nodes are combined into a new semantic representation in the parent node. This is done based on node label information. However, it is possible to discard label information and simply learn to combine child nodes representations based on example occurrences in a dataset. This would allow for DRS construction, independent from node labels.

Unfortunately, preliminary results suggested that the DRS construction rules obtained using this statistical approach generalize poorly. The number of construction rules grows almost linearly as the dataset grows, and DRSs for unseen sentences could often not be constructed due to missing rules. Additionally, a dataset would be required that contains both continuous and discontinuous parse trees for its sentences (or at least contains traces in order to reconstruct discontinuities), in addition to DRT semantic analysis. Learning construction rules on a dataset containing only continuous parse trees, such as the Groningen Meaning Bank mentioned earlier, would bias task success towards continuous parse trees.

All in all, it appears that evaluating the influence of discontinuity formally using dynamic semantics is a challenging proposition. Discontinuous parse trees are currently not integrated well with leading semantic formalisms like DRT, and as a result, it is difficult to obtain unbiased results when evaluating discontinuity. Part of the problem is the fact that corpora with DRT annotation do not contain discontinuous parse trees, making objective evaluation difficult. As we will see in Chapter 4, tree-based neural networks offer a much easier way to evaluate discontinuity in parse trees, and will be used in our experiments as detailed in Chapter 5.

Chapter 4

Tree-based Neural Networks

Neural networks have seen a lot of success in natural language processing recently. One of the keys to this success is how easily parse tree information may be incorporated into deep networks. Another important factor is the representation of words as meaningful vectors, called word embeddings.

In this chapter, we first explain the benefit of word embeddings. Subsequently, we discuss a number of neural network architectures and their relationship with different parse tree types. Lastly, we explore possible NLP tasks for evaluating the influence of discontinuity.

4.1 Word Embeddings

Representing words as vectors is central to the success of neural network models in NLP tasks. Many NLP techniques, like the dynamic semantics approach discussed in the previous chapter, treat words as atomic units, ignoring any similarity between words. Representing words as vectors opens up the possibility to incorporate contextual information into words, and offers a natural distance measure in the vector cosine distance.

As an example of how these word embeddings represent meaning, let us consider the example given in [23]: the words 'king' and 'queen' are similar, in the sense that they both describe a ruler. The difference is that 'king' is a male ruler, while 'queen' is female. If we look at the vectors, we can see that if we compute $vector(king) - vector(man) + vector(woman)$, we find a vector that is very similar to $vector(queen)$. This indicates that the word vectors support the meaning as we have just described.

Many methods exist for generating these so-called 'word embeddings'. Some methods utilize neural network models, others are based on statistical analysis. Every method,

however, is based on word co-occurrences in a corpus as the primary source for learning these word representations.

Among the current state of the art models is the Global Vectors (GloVe) model [24] [1]. This method utilizes word co-occurrence statistics directly by applying matrix operations to the original word co-occurrence matrix. Another popular model is the word2vec model [23] [2], which utilizes CBOW and Skip-gram log-linear models.

It is also possible to optimize word embeddings for specific tasks. [25] proposes a method for improving word embeddings with regards to sentiment information, using a mix of unsupervised and supervised techniques. However, for the purpose of our research, we will stick to the more modern, unsupervised methods mentioned earlier.

4.2 Recurrent and Recursive Neural Networks

Once we have word vector representations, we may employ recurrent neural networks to generate a vector representation for an entire sentence. This is useful for performing classification tasks, like, for example, the previously mentioned sentiment analysis task. Recurrent neural networks may also be generalized to recursive neural networks, which makes it possible to integrate parse tree information into the learning process.

Recurrent neural networks are defined as having at least one directed ring in its structure. Formally, the task of reducing a sentence to a single vector for classification purposes using a recurrent neural network may be defined as follows:

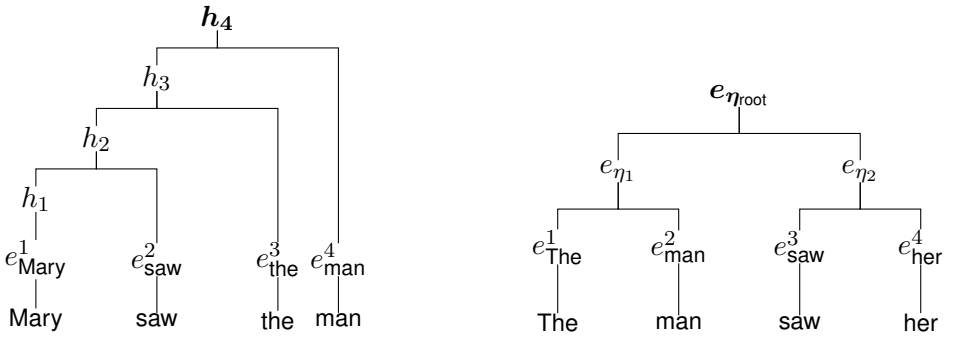
We have a sentence S defined as a sequence of words: $S = \{w_1, w_2, ..., w_{N_S}\}$, where N_S is the number of words in the sequence. Each word is defined by a K-dimensional word embedding $e_w = \{e_w^1, e_w^2, ..., e_w^K\}$.

The recurrent neural network combines at each timestep t the previously built hidden vector h_{t-1} with the current word embedding w_t, resulting in h_t:

$$h_t = f(W \cdot h_{t-1} + V \cdot e_t) \tag{4.1}$$

where W and V denote compositional matrices, and function f represents the neural network application. The desired output, a K-dimensional vector representing the whole

[1] GloVe code available at this link: `http://nlp.stanford.edu/projects/glove/`
[2] word2vec code available at this link: `https://code.google.com/p/word2vec/`



(a) Recurrent neural network

(b) Recursive neural network

Figure 4.1: Examples of recurrent and recursive networks. The bolded 'root' nodes contain the K-dimensional vector representation of the whole sentence.

sentence, is thus given by h_{N_S}. A visual representation of the recurrent neural network is seen in Figure 4.1a.

Recurrent neural networks are easily generalized to recursive neural networks, replacing the incremental left-to-right nature of the recurrent model by a bottom-up approach that combines child nodes at each node of a parse tree, until the root of the tree is reached. This means the timesteps of the recurrent model are no longer necessary, and instead progress is saved per node η. Recursive neural networks may then be defined as:

$$e_\eta = f(W \cdot e_{\eta_{\text{left}}} + V \cdot e_{\eta_{\text{right}}}) \tag{4.2}$$

Figure 4.1b shows a visual representation of the recursive neural network. As was mentioned, recursive networks are a generalization of recurrent neural networks. The figure clearly shows why this is the case: recurrent neural networks are equal to recursive neural networks when a left-branching parse tree is used.

Important to note is that these recursive neural networks seemlessly support discontinuous parse trees: at each node, the left and right child is taken as input, regardless of whether the phrases spanned by these children are next to each other in the sentence or not. Even trees in which nodes have multiple parents, as generated by range concatenation grammars ([26]), are supported, as a single node may be used as input multiple times without problems.

When performing classification tasks, a *softmax* layer may be added as the final layer of the neural network, to compute the probability of assigning a class c to an input x; in this case the sentence vector.

4.3 The Long Short-Term Memory model

The Long Short-Term Memory (LSTM) model was originally proposed by [27] as an extension to the recurrent neural network models discussed in the previous section. The goal of these models is to improve the handling of long-term dependencies: when using recurrent neural network models, a large distance between dependents means the influence of the original dependent is almost completely gone when encountering the second dependent. This is called the 'vanishing gradient problem', and is discussed at length in [28].

LSTM models address this problem by introducing memory state, and a number of gates: an input, memory, and output gate, respectively denoted as i_t, f_t, and o_t. These gates determine exactly how the new input e_t influences the current memory state c_t, and what the vector representation for the current timestep, h_t, will be. The vector representation h_t will then be given by:

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ l_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{bmatrix} W \cdot \begin{bmatrix} h_{t-1} \\ e_t \end{bmatrix} \tag{4.3}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot l_t \tag{4.4}$$

$$h_t^S = o_t \cdot c_t \tag{4.5}$$

W in this case is a $4K \times 2K$ dimensional real-valued weight matrix, assuming K-dimensional word embeddings. Figure 4.2 shows a visual representation of the LSTM model[3].

The idea of LSTM may also be applied to recursive models, in the same way we generalized recurrent models to recursive models. We can define the vector representation e_η at node η as:

$$e_\eta = z_\eta \cdot (e_{left} + e_{right}) + (1 - z_\eta) e'_\eta \tag{4.6}$$

$$e'_\eta = f(W \cdot r_\eta \cdot e_{left} + V \cdot r_\eta \cdot e_{right}) \tag{4.7}$$

[3]This visualization was taken from a blog post by Christopher Olah on LSTM models: `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`

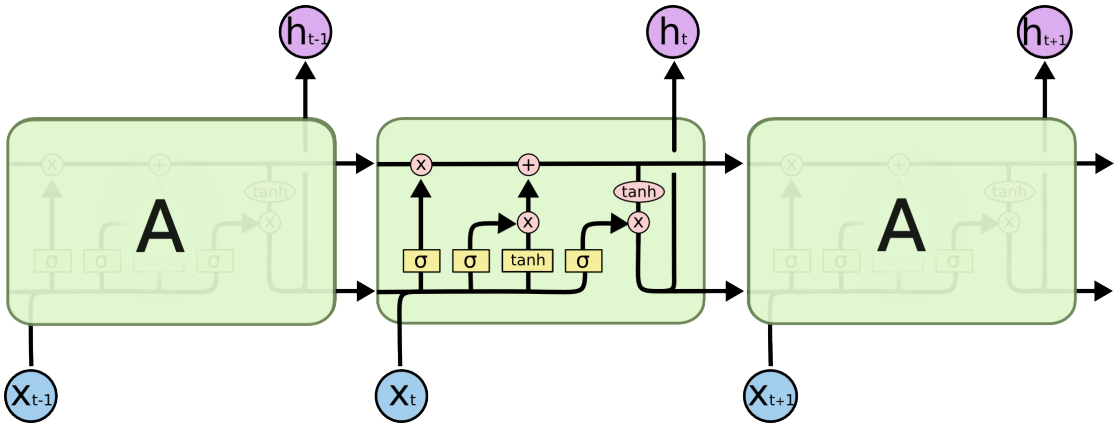


Figure 4.2: A visualization of the LSTM model structure.

where r_η and z_η are the control gates associated with η, given by:

$$z_\eta = \sigma(W_z \cdot e_{left} + V_z \cdot e_{right}) \tag{4.8}$$

$$r_\eta = \sigma(W_r \cdot e_{left} + V_r \cdot e_{right}) \tag{4.9}$$

There are a number of variations available of this tree-based version of the LSTM model, also called Tree-LSTM, which achieve state-of-the-art results on many natural language classification tasks ([29], [30], [31]). It is, however, also argued that the recurrent version of the model can achieve similar results in certain cases ([32]).

Different variants of the LSTM model have been proposed, most of which are based on slight variations on the gating principle. Most notably, Gated Recurrent Unit models ([33]) and Depth-gated LSTM models ([34]) have produced consistently good results. Bi-directional LSTM models have also been proposed, both for recurrent ([35]) and recursive ([36]) networks. Which variant of the LSTM model is best is debatable; [37] compares a number of LSTM variants and interestingly finds no significant improvement over the basic LSTM model. For this reason, we stick to the basic LSTM model, as explained in this section, for our research and experiments.

4.4 Evaluating Discontinuity through Tree-LSTM

As detailed in the previous sections, text classification tasks may be performed based on neural network analysis, followed by softmax classification. [32] have performed multiple text classification tasks this way, including sentiment analysis, sentence-target

matching, and semantic relation classification, successfully comparing different neural network models. Since our aim is to compare parse tree types, we may set up a similar experiment, but varying the parse trees while keeping the neural network model constant.

As mentioned, the recursive models support different kinds of parse trees well. Additionally, the Tree-LSTM model supports long-distance dependencies, making it ideal for evaluating discontinuity. The Tree-LSTM model is thus a clear choice for our experiments.

The challenge remains to select the right classification tasks to maximally highlight the influence of discontinuity. We select two tasks: sentiment analysis and question classification. In sentiment analysis, long distance dependencies may have a large influence on the classification results. In question classification, discontinuity is very prevalent in the data, mostly in the form of wh-movement. Details of these tasks will be discussed in Chapter 5.

All in all, it is clear that tree-based neural networks are well-suited for evaluating the influence of discontinuity. Therefore, we choose to focus on this approach, rather than the dynamic semantic approach detailed in Chapter 3. In the next chapter, we detail the experiments in which we perform and evaluate the previously mentioned classification tasks, based on tree-based neural networks.

Chapter 5

Experiments

As discussed in Chapter 4, tree-based neural network models are especially well-suited for comparing different parse tree types. In this chapter, we describe two NLP tasks: sentiment analysis and question classification. For both these tasks we utilize a long short-term memory model. We first describe the specifics of this model, and how we obtain the parse trees for the datasets. Then we discuss the specifics of each task and the results.

5.1 Experimental Setup

While the datasets for the sentiment analysis task and the question classification task differ, the way we obtain parse trees is the same for both tasks. The neural network model, and the process of parameter tuning, is also identical. In this section, we will explain these parts of the experiment.

5.1.1 Parse Tree Types

The goal of this experiment is to compare continuous and discontinuous parse trees. These parse trees were obtained using the disco-dop parser[1] made by Andreas van Cranenburgh ([2]). The grammars used for generating the continuous and discontinuous trees were extracted from the Wall Street Journal sections 2-21. This part of the treebank contains traces in the annotation, which makes it possible to convert the originally continuous parse trees to discontinuous parse trees.

[1]The disco-dop parser is freely available at `https://github.com/andreasvc/disco-dop`

The original (continuous) parse trees that were part of the dataset were discarded in favor of these newly generated parse trees. This was done to ensure that the trees generated for both types were based on the same treebank and generated by the same parsing algorithm. While this does mean that the text is parsed using out-of-domain grammar, the resulting continuous parse trees were compared to the original parse trees and did not perform significantly worse (in some cases, task performance was actually better).

As a baseline for these two parse tree types, we use left-branching parse trees. This type of parse tree effectively mimics the recurrent neural network model, incrementally processing each word of the sentence one-by-one.

5.1.2 Model and Parameter Tuning

The model we use is the long short-term memory model. The main reason for using this model is that it is the first model that solves the vanishing gradient problem, as discussed in section 4.3. The reason this is important, is that our parse trees do not have classification at intermediate nodes; class information is only available at the sentence level, and thus this information needs to be propagated through the entire tree.

We do not make use of newer variants of the LSTM model. While these networks seem very promising, for the case of this research it seemed appropriate to use a type of network that is more established in the literature. Hence the choice for the original LSTM model.

For word embeddings, we use 300-dimensional GloVe word vectors ([24]). From earlier research ([29], [32]), we see that for these tasks a dimensionality of between 50 and 300 is often used, with higher dimensionality offering better task performance.

The LSTM model parameters that we tune are mini-batch size, number of dimensions, and learning rate. The parameters are tuned separately for each parse tree type. For each parameter set, we train fifty models and evaluate them on the development set, noting the mean score. The parameter set with the highest mean score is the one we select to generate scores for the test set.

5.1.3 Test set scoring

The results of the experiments are the classification accuracy on the test set for each of the parse tree types. The classification accuracy is given by $100 \times \frac{\#correct}{\#total}$. Since there

is some randomness involved in training a neural network, the training process is run multiple times, and the test set score is averaged over multiple runs.

In total, 20 test set scores are generated for each model. For each test set score, the network is trained 15 times and evaluated on the development set. The network model with the highest accuracy on the development set is used to generate a score for the test set. The results reported are the mean and standard deviation of these scores.

5.2 Sentiment Analysis

The first task we discuss is sentiment analysis. Sentiment analysis is the task of identifying the sentiment of a piece of text. Often the text is classified binarily: positive or negative. As an example, *"Today was a great day."* would be classified as having a positive sentiment, while *"The weather today is horrible."* would be classified as negative. Alternatively, sentiment may be categorized more finely, including multiple positive and negative categories (e.g. 'very positive') and optionally a neutral category.

There are a number of reasons for selecting this specific task for comparing syntax structures. First of all, it is a semantic task. Thus the performance of the task will tell us about how well each syntax structure type captures meaning. Second, task performance is easy to measure. Since it is a classification task, performance can be measured using a simple accuracy measure. This means that different models will be able to be evaluated and compared objectively. Third, sentiment analysis is a task in which discontinuity can play an important role in determining the outcome.

Other sentiment analysis approaches include a subjective lexicon, and n-gram modelling techniques.

5.2.1 Dataset

The dataset we use for our experiment is the Stanford Sentiment Treebank[2] ([38]). This dataset includes text from movie reviews, with five sentiment categories (very positive, positive, neutral, negative, very negative).

The dataset includes a total of 11,855 sentences, which is split as suggested into a training set of 8544 sentences, a development set of 1101 sentences, and a test set of 2210 sentences.

[2]The Stanford Sentiment Treebank is freely available for download at the following link: `http://nlp.stanford.edu/sentiment/index.html`

The dataset can also be used for binary sentiment analysis by removing the neutral examples, resulting in 6920 sentences for training, 872 for development, and 1821 for testing.

An important statistic to note is that for this dataset, 9.76% of sentences contain at least one discontinuous constituent. This is a quite low amount, which is related to the domain: many sentences are just noun phrases, for example *"A deep and meaningful film."*, and these sentences are adequately described by continuous constituents. This means that the difference between continuous and discontinuous may not be as pronounced in this task as we would like.

5.2.2 Results

Table 5.1 shows the results of the sentiment analysis task. Two versions of the task have been carried out: a binary classification task, classifying sentences as either *positive* or *negative*, and a harder version of the task with five sentiment categories (*very positive*, *positive*, *neutral*, *negative*, *very negative*). The binary task is denoted in the table under 'Coarse', and the task requiring more fine-grained classification is denoted under 'Fine'.

As we can see from the table, the left-branching baseline is significantly outperformed by all other models. This confirms that parse tree type does influence the task performance for this task. When comparing the continuous and discontinuous model, it seems that the continuous model performs better, however, the difference is not significant ($p = 0.169$ for the fine task, $p = 0.245$ for coarse). The mixed model is a combination of the continuous and discontinuous models, and will be explained later in this section.

Model	Fine	Coarse
Left-branching	45.5 (0.8)	85.3 (0.6)
Continuous	47.1 (1.0)	86.6 (0.5)
Discontinuous	46.6 (1.2)	86.3 (1.0)
Mixed	46.9 (1.2)	86.4 (0.9)

Table 5.1: Results for the sentiment analysis task. The scores are the percentage of correctly classified sentences, followed by the standard deviation.

In order to better investigate the influence of discontinuity, we repeat the experiment, but reduce the development and test sets to sentences containing at least one discontinuous constituent. The results of this experiment can be seen in Table 5.2.

Again, we find that the left-branching baseline is clearly outperformed by the continuous and discontinuous models. This is the case both for the fine classification task ($p = 0.0141$ and $p = 0.0604$ respectively) and the coarse classification task ($p = 0.00248$ and $p = 4.81 * 10^{-5}$ respectively). The difference between the continuous and discontinuous

models in the fine task is not significant ($p = 0.635$). In the coarse task, however, the discontinuous model significantly outperforms the continuous model ($p = 0.0485$).

Model	Fine	Coarse
Left-branching	39.9 (3.9)	76.2 (2.5)
Continuous	42.8 (2.8)	78.6 (1.8)
Discontinuous	42.3 (3.7)	**79.8 (1.8)**

Table 5.2: Sentiment analysis results of sentences with discontinuous constituents. The bold font means that the score is significantly greater than all the other scores for that experiment.

This experiment seems to indicate that discontinuous parse trees may indeed lead to better task performance, at least for the coarse classification task, yet the previous experiment finds no significant difference between the continuous and discontinuous models. One explanation is that the discontinuous model actually performs better, but the experiment didn't lead to a significant result. This is plausible, due to the relatively low number of sentences in the data containing a discontinuous constituent. Additionally, since the Tree-LSTM model native handles long-distance dependencies well, we expect the advantage of discontinuous parse trees to be somewhat diminished.

Another explanation is that the grammar used to generate the parse trees for the continuous model produces higher quality trees for the sentences that do not contain discontinuities, since it does not need to account for possible discontinuities. These higher quality trees would then lead to a better task performance, due to better classification of the continuous sentences outweighing a lower score on sentences with discontinuities.

In order to further investigate this second explanation, we introduce a new model which combines the continuous and discontinuous model. This 'mixed' model utilizes the discontinuous syntax trees produced by the discontinuous grammar, yet for the sentences containing no discontinuities uses the trees produced by the continuous grammar. This model would get the benefit of the both the higher quality continuous trees and correct analysis of sentences with discontinuities.

The result of this mixed model is reported in Table 5.1. It scores somewhere in between the continuous and the discontinuous models, however, there is no significant difference between the mixed model and the continuous or discontinuous model.

5.2.3 Discussion

The fact that the experiment leads to only a small number of significant results makes it difficult to draw clear conclusions. The conclusions we can draw are the following:

- Parse tree structure influences task performance. This becomes clear from the left-branching baseline being structurally outperformed by the other models.
- Discontinuous parse trees may lead to better task performance. This becomes clear from the performance of the discontinuous model on the sentences containing discontinuities.

These conclusions seem to confirm our hypothesis that using discontinuous parse trees may increase NLP task performance. However, the fact that only one out of four experiments leads to a significant result calls this into question. Why do we not find significant results in the other experiments?

The main reason for the lack of significant results seem to be the low number of sentences containing discontinuous constituents. The mixed model explored the possibility of the continuous model having higher quality continuous trees, however, this did not lead to a significant result either. All we can say about this is that all the models lie quite closely together, and that the differences did not become clear through the twenty test set scores that were calculated per model.

We can, however, look at individual sentences and see what each model does differently. For every sentence, we registered the predicted sentiment class during each of the twenty runs, leading to an accuracy score per sentence. By analyzing the accuracy of the different models on individual sentences, we can pinpoint the sentences that the models struggle with, or handle well. Table 5.3 shows the sentences with the greatest difference in accuracy between the continuous and the discontinuous model.

From Table 5.3, we do not immediately find clear differences between the two sets of sentences. Sentence length is similar, and there seems to be no syntax construction that is exclusive to either set. What is remarkable, however, is that a simple sentence like "Excellent acting and direction ." can lead to such a great difference in accuracy between models. In order to investigate this further, we look at the parse trees of this sentence, for both models.

Figures 5.1a and 5.1b show the binarized parse trees as used in the experiment, while figures 5.1c and 5.1d show the original parse trees generated by the parser. The sentence contains no discontinuities, but seems to be analyzed better by the discontinuous model. Especially after binarization, the tree of the continuous model becomes a right-branching tree, while the tree of the discontinuous model retains a more intuitive noun phrase structure. It should be noted that the choice for right-branching binarization may not

	Continuous winning
1	What Jackson has accomplished here is amazing on a technical level .
2	It 's disappointing when filmmakers throw a few big-name actors and cameos at a hokey script .
3	A long-winded and stagy session of romantic contrivances that never really gels like the shrewd feminist fairy tale it could have been .
4	If there 's nothing fresh about Wannabes , which was written by Mr. DeMeo , who produced and directed the film with Charles A. Adessi, much of the time the movie feels authentic .
5	Still pretentious and filled with subtext , but entertaining enough at ' face value ' to recommend to anyone looking for something different .

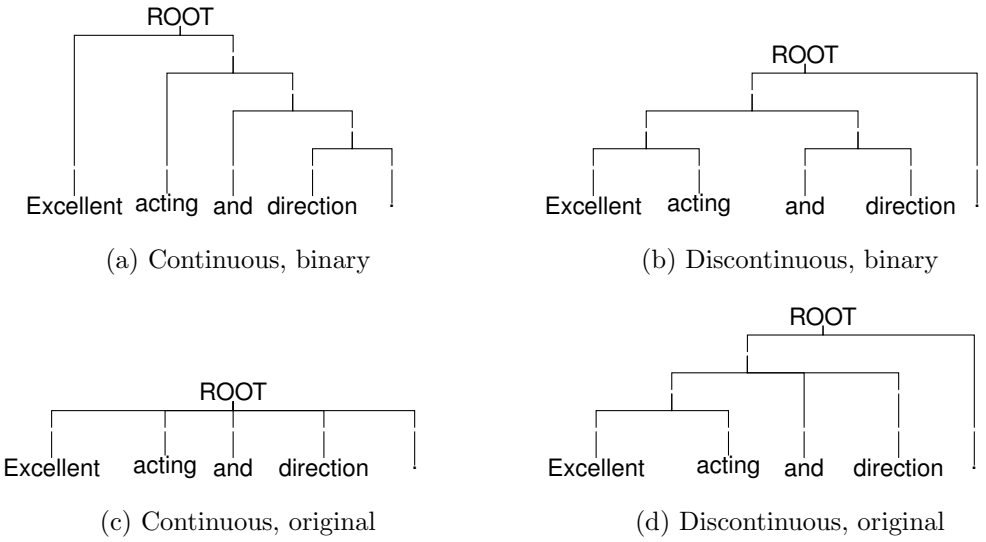
	Discontinuous winning
1	Excellent acting and direction .
2	Where last time jokes flowed out of Cho 's life story , which provided an engrossing dramatic through line , here the comedian hides behind obviously constructed routines .
3	By the end , I was looking for something hard with which to bludgeon myself unconscious .
4	Ostensibly celebrates middle-aged girl power , even as it presents friendship between women as pathetic , dysfunctional and destructive .
5	I did n't laugh at the ongoing efforts of Cube , and his skinny buddy Mike Epps , to make like Laurel and Hardy 'n the hood .

Table 5.3: The sentences for which the continuous model achieves higher accuracy are listed first, followed by the sentences where the discontinuous model outperforms the continuous model.

have been optimal, as the baseline model for the sentiment analysis task is the left-branching tree. However, in this case, the right-branching tree does not seem inferior to the left-branching tree.

Figure 5.2 shows the parse trees for the sentence on which the continuous model outperforms the discontinuous model by the largest margin. Interestingly, this sentence does contain a discontinuous constituent, although this discontinuity does not seem like a determining factor in expressing the sentiment of the sentence. However, other than this one discontinuity, the trees are identical between the two models, which means in this case the discontinuous constituent is directly responsible for poor task performance. It seems that the modelling of wh-movement does not always lead to better task performance.

From the analysis of these parse trees, it appears that the continuous model does not necessarily produce higher quality parse trees for continuous sentences. Continuous sentences are in some cases analyzed even better by the discontinuous model. In addition, the modeling of discontinuity with discontinuous constituents does not seem to always lead to better task performance, reducing any advantage the model may have in this



(a) Continuous, binary

(b) Discontinuous, binary

(c) Continuous, original

(d) Discontinuous, original

Figure 5.1: Example from the sentiment analysis dataset. On this sentence, the discontinuous model performs better than the continuous model.

area. These factors contribute to the fact that the results of the models lie so closely together.

5.3 Question Classification

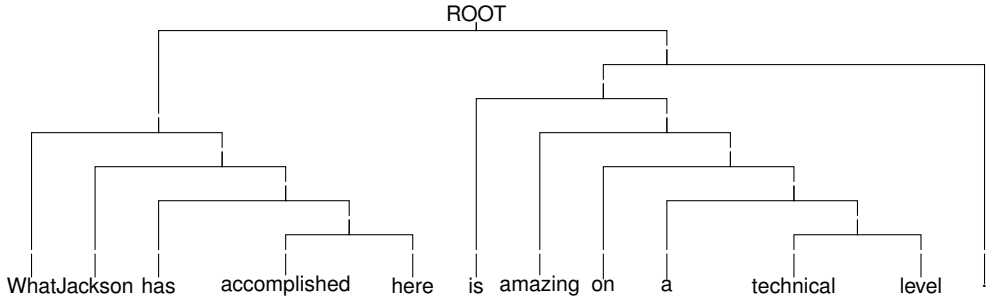
As we have seen in Chapter 2, discontinuity occurs often when a sentence is stated in the form of a question. This is modeled well with discontinuous constituents, and thus a question classification task is exceptionally well-suited to explore the influence of discontinuous constituents on NLP tasks.

Question classification is the task of categorizing a question according to the type of answer it expects. For example, the question *"Who wrote 'Hamlet'?"* expects an answer in the form of the name of an individual, while the question *"What does palindromic mean ?"* asks for a definition. Classifying questions according to such categories can be seen as a first step in automated question answering.

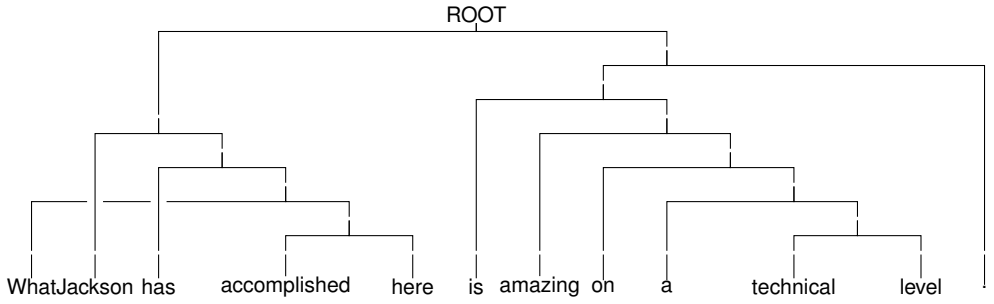
5.3.1 Dataset

For this experiment, we use the question classification dataset that was made available by Xin Li and Dan Roth ([39]) [3]. In the dataset, six main question classes are distinguished: abbreviation, entity, description, human, location, and numeric. Additionally, each class

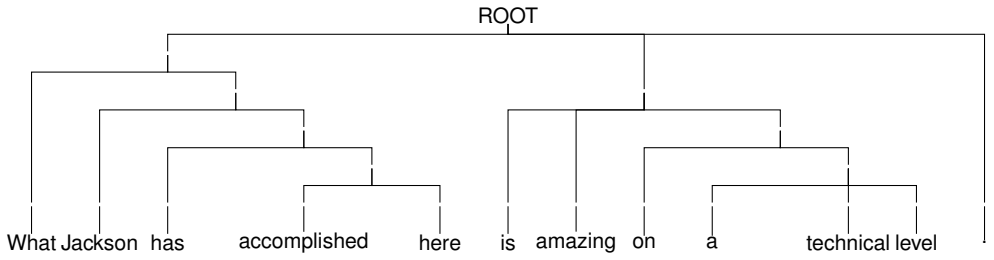
[3]The Question Classification data is freely available for download at the following link: `http://cogcomp.cs.illinois.edu/Data/QA/QC/`



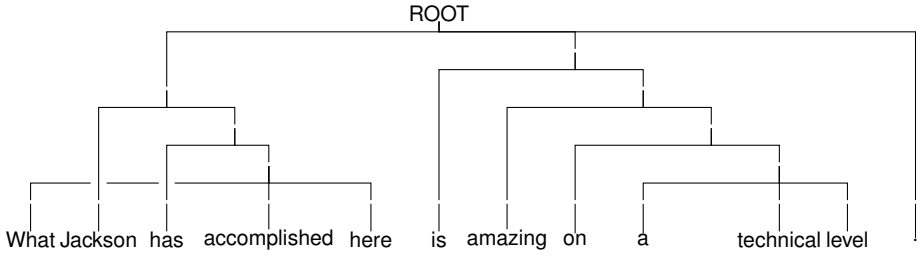
(a) Continuous, binary



(b) Discontinuous, binary



(c) Continuous, original



(d) Discontinuous, original

Figure 5.2: Example from the sentiment analysis dataset. On this sentence, the continuous model performs better than the discontinuous model.

has multiple subclasses, making for a total of 50 subclasses. This allows us to set up the experiment similarly to the sentiment analysis experiment, with a fine-grained classification task (all 50 subclasses) and a coarse version (only the six main classes).

The dataset offers a standard training and test set, but no development set. We randomly select 10% of the training set for development, leading to a training set of 4906 sentences, a development set of 546 sentences, and a test set of 500 sentences.

In this dataset, 29.6% of sentences contain at least one discontinuous constituent, which is more than three times the amount as compared to the sentiment analysis dataset. This will better highlight the influence of discontinuity in the parse trees.

5.3.2 Results

For the question classification experiment, we repeat the steps we took for the sentiment analysis experiment: we execute the task both on the original dataset, including the use of a mixed model, and on a version of the dataset that contains only the sentences that contain discontinuities. The results can be seen in Table 5.4 and Table 5.5.

From Table 5.4, we find an interesting difference between the fine and the coarse task: for the fine task, the continuous model significantly outperforms the discontinuous model ($p = 0.0289$), while for the coarse task, the discontinuous model wins by a significant margin ($p = 0.0392$). Additionally, the there seems to be no significant difference between the losing model and the left-branching baseline. The mixed model falls somewhere in between, just as it did in the sentiment analysis experiment.

Model	Fine	Coarse
Left-branching	76.6 (1.1)	87.8 (0.6)
Continuous	**77.8 (1.1)**	87.8 (1.0)
Discontinuous	76.9 (1.3)	**88.5 (1.0)**
Mixed	77.0 (1.3)	88.2 (1.3)

Table 5.4: Question classification results.

On the part of the dataset with discontinuities, the results as reported in Table 5.5 show the continuous model as having the best task performance in both the fine and coarse task, altough the difference with the other models is not significant. There is however an interesting contrast between the results of this experiment as opposed to the sentiment analysis experiment: for this experiment, task performance increases on the dataset with discontinuities, whereas sentiment analysis performance went down.

Model	Fine	Coarse
Left-branching	80.8 (2.1)	91.4 (1.6)
Continuous	82.3 (2.5)	93.2 (1.3)
Discontinuous	81.4 (2.1)	92.5 (1.3)

Table 5.5: Question classification results on sentences with discontinuous constituents.

5.3.3 Discussion

The results of the question classification task may be summarized as follows:

- Continuous models perform better on the fine classification task, while discontinuous models perform better on the coarse classification task.
- Overall task performance for all models is higher on sentences with discontinuous constituents.

The fact that the continuous model is outperforming the discontinuous model on the fine classification task is interesting, especially with the discontinuous model winning the coarse classification task. We would expect the discontinuous model to perform especially well on the fine classification task, since the option for discontinuous parse trees allows a wider range of possible analyses to capture the necessary intracies for correctly classifying 50 subclasses. However, the results show that the opposite is true, and the discontinuous model actually shines in the coarse classification task. A possible explanation for this is that the continuous model is more stable. The discontinuous model has greater expressive power, but can therefore be somewhat less accurate. The result, in this case, is a better performance on the coarse task, which doesn't require as much accuracy, and a lower performance on the fine task.

An interesting thing to note is that the mixed model does not outperform the continuous or discontinuous models in any of the experiments. This indicates that our initial assumption that the discontinuous model should perform better on sentences containing discontinuities, and the continuous model should perform better on sentences without discontinuities, is wrong. This is also supported by the results of the models on sentences containing discontinuities, which shows no significant difference between models. We can conclude that the discontinuous model performs better overall on the coarse classification task, not just on sentences with discontinuities. And the same can be said of the continuous model, which performs better overall on the fine classification task.

The overall increase in performance on the set of sentences with discontinuities is almost certainly correlated with the question domain. In questions, discontinuity is often caused by wh-movement, as explained in section 2.1. Sentences which are able to be analyzed

as containing wh-movement, and thus parsed with discontinuous constituents, are most likely the easier questions to answer. Once sentences get very complex, the wh-movement may not be as clear, and as such these sentences may end up being parsed with only continuous constituents. This means that the dataset of discontinuous constituents contains, on average, questions that are less complex syntactically, and as a result easier to classify correctly.

Chapter 6

Conclusion

In this research, we have explored the concept of discontinuity, and how discontinuity presents itself in parse trees, with the goal of evaluating the influence of discontinuity in NLP tasks. We have discussed different linguistic phenomena associated with discontinuity, and discussed the way different parsing methods deal with this concept. Subsequently, we introduced both dynamic semantics and tree-based neural network models as a basis for evaluating discontinuity in NLP tasks. Finally, we carried out sentiment analysis and question classification tasks based on the Tree-LSTM neural network model, comparing the performance of continuous and discontinuous parse trees.

Both the sentiment analysis task and the question classification task show a clear influence of the parse tree type on task performance. While the different parse tree models produce quite similar results in general, the continuous and discontinuous models do produce significant results in a number of cases. In the sentiment analysis task, the discontinuous model has the advantage on the part of the dataset containing discontinuities. This is according to our expectation that discontinuity is modeled well by discontinuous constituents. In the question classification task, we find that both the continuous and the discontinuous model have their strengths: the continuous model performs better on the fine classification task, while the discontinuous model performs better on the coarse classification task. It appears that the continuous model produces somewhat more refined results, while the discontinuous model is more expressive yet unrefined.

In conclusion, the results do not unambiguously point to discontinuous parse trees as a better representation of meaning. There are some conditions under which continuous parse trees actually produce better NLP task results. We can, however, conclude that discontinuous parse trees have the advantage in a number of NLP tasks, and as such,

discontinuity has a noticable influence on NLP task performance that should not be ignored.

6.1 Contributions and Future Work

In this thesis, we have combined existing methods, including tree-based neural network models and discontinuous parsing, in a novel way in order to evaluate the influence of discontinuity in a task-based environment. This work also provides insight into the interaction between discontinuous parse trees and tree-based neural networks, which to our knowledge has not been tested before in the literature.

In addition to these contributions, we have also explored the option for evaluation based on dynamic semantics. While we did not end up testing this, it is worth exploring DRS construction for mildly context-sensitive grammars. This will allow for evaluation of grammar formalisms, rather than parse tree structure.

The experiments we have run could be expanded upon in a number of ways. Obviously, there are more possible NLP tasks that could be evaluated. A good candidate would be a semantic relation classification task as detailed in [40], which aims to learn long-distance relationships. It would also be interesting to see the effect of neural network models other than the Tree-LSTM model. Since the Tree-LSTM model natively handles long-distance dependencies well, we would expect to see a larger influence of discontinuity in simpler recusive neural network models. Finally, expanding the experiments to languages other than English will yield interesting results on the influence of discontinuity in different languages. We would expect greater influence for languages in which discontinuity is more prevalent, such as German.

Bibliography

[1] Kilian Evang and Laura Kallmeyer. Plcfrs parsing of english discontinuous constituents. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 104–116. Association for Computational Linguistics, 2011.

[2] Andreas van Cranenburgh and Rens Bod. Discontinuous parsing with an efficient and accurate dop model. *IWPT-2013*, page 7, 2013.

[3] Wolfgang Maier and Timm Lichte. Discontinuous parsing with continuous trees. *Proceedings of DiscoNLP*, pages 47–57, 2016.

[4] Joakim Nivre and Jens Nilsson. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 99–106. Association for Computational Linguistics, 2005.

[5] Daniel H Younger. Recognition and parsing of context-free languages in time n 3. *Information and control*, 10(2):189–208, 1967.

[6] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[7] Stuart M Shieber. Evidence against the context-freeness of natural language. In *The Formal complexity of natural language*, pages 320–334. Springer, 1985.

[8] Riny Huybregts. The weak inadequacy of context-free phrase structure grammars. *Van periferie naar kern*, pages 81–99, 1984.

[9] Aravind K Joshi, Leon S Levy, and Masako Takahashi. Tree adjunct grammars. *Journal of computer and system sciences*, 10(1):136–163, 1975.

[10] Aravind K Joshi and Yves Schabes. Tree-adjoining grammars. In *Handbook of formal languages*, pages 69–123. Springer, 1997.

[11] Krishnamurti Vijay-Shanker, David J Weir, and Aravind K Joshi. Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings*

of the 25th annual meeting on Association for Computational Linguistics, pages 104–111. Association for Computational Linguistics, 1987.

[12] David Jeremy Weir. Characterizing mildly context-sensitive grammar formalisms. 1988.

[13] Laura Kallmeyer. *Parsing beyond context-free grammars*. Springer, 2010.

[14] Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen-Schirra, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. Tiger: Linguistic interpretation of a german corpus. *Research on Language and Computation*, 2(4):597–620, 2004.

[15] Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. An annotation scheme for free word order languages. In *Proceedings of the fifth conference on Applied natural language processing*, pages 88–95. Association for Computational Linguistics, 1997.

[16] Hans Kamp. A theory of truth and semantic representation. *Formal semantics-the essential readings*, pages 189–222, 1981.

[17] Jeroen AG Groenendijk, Martin Johan Bastiaan Stokhof, et al. Dynamic montague grammar. 1990.

[18] Jeroen Groenendijk and Martin Stokhof. Dynamic predicate logic. *Linguistics and philosophy*, 14(1):39–100, 1991.

[19] Jonathan Calder, Ewan Klein, and Henk Zeevat. Unification categorial grammar: A concise, extendable grammar for natural language processing. In *Proceedings of the 12th conference on Computational linguistics-Volume 1*, pages 83–86. Association for Computational Linguistics, 1988.

[20] Hans Kamp and Uwe Reyle. From discourse to logic; introduction to the modeltheoretic semantics of natural language. 1993.

[21] Hans Kamp, Josef Van Genabith, and Uwe Reyle. Discourse representation theory. In *Handbook of philosophical logic*, pages 125–394. Springer, 2011.

[22] Valerio Basile, Johan Bos, Kilian Evang, and Noortje Venhuizen. Developing a large semantically annotated corpus. In *LREC*, volume 12, pages 3196–3200, 2012.

[23] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[24] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543, 2014.

[25] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150. Association for Computational Linguistics, 2011.

[26] Pierre Boullier. Range concatenation grammars. In *Proceedings of the 6th International Workshop on Parsing Technologies*, pages 53–64. International Conference on Parsing Technologies, 2000.

[27] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[28] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[29] Phong Le and Willem Zuidema. Compositional distributional semantics with long short term memory. *arXiv preprint arXiv:1503.02510*, 2015.

[30] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

[31] Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. Long short-term memory over tree structures. *arXiv preprint arXiv:1503.04881*, 2015.

[32] Jiwei Li, Dan Jurafsky, and Eudard Hovy. When are tree structures necessary for deep learning of representations? *arXiv preprint arXiv:1503.00185*, 2015.

[33] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[34] Kaisheng Yao, Trevor Cohn, Katerina Vylomova, Kevin Duh, and Chris Dyer. Depth-gated lstm. *arXiv preprint arXiv:1508.03790*, 2015.

[35] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[36] Ozan Irsoy and Claire Cardie. Bidirectional recursive neural networks for token-level labeling with structure. *arXiv preprint arXiv:1312.0493*, 2013.

[37] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *arXiv preprint arXiv:1503.04069*, 2015.

[38] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer, 2013.

[39] Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics, 2002.

[40] Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 94–99. Association for Computational Linguistics, 2009.