HCP with PSMA: A Robust Spoken Language Parser

Monirul Hasan, Venkatesh Manian, Christel Kemke
Department of Computer Science, University of Manitoba, Winnipeg, Canada
{kmhasan, venkat, ckemke}@cs.umanitoba.ca

Abstract

"Spoken language" is a field of natural language processing, which deals with transcribed speech utterances. The processing of spoken language is much more complex and complicated than processing standard, grammatically correct natural language, and requires special treatment of typical speech phenomena called "disfluencies", like corrections, interjections and repetitions of words or phrases. We present a parsing technique that utilizes a Hybrid Connectionist Parser (HCP) extended with a Phrase Structure Matching Algorithm (PSMA) for the syntactic processing (parsing) of spoken language. The HCP is a hybrid of a connectionist and a symbolic approach, which builds a neural network dynamically based on a given context-free grammar and an input sentence. It has an advantage over traditional parsers due to the use of graded activation and activation passing in the network. These features were exploited in techniques to detect and repair disfluencies, in combination with the Phrase Structure Matching Algorithm, which operates on the graphical structure of the network. This approach to spoken language parsing is purely syntactical and – in contrast to other work – does not require a modified grammar for spoken language, or information derived from the speech input, nor any pre-marking of disfluencies. We implemented the combined HCP and PSMA parser and tested it on a standard corpus for spoken language, the HCRC Map Task Corpus. The experiments showed very good results, especially for a purely syntactic method, which detects as well as corrects disfluencies.

1 Introduction

Over the last 50 years, researchers from different fields have worked on the problem to describe natural languages so that they can be processed by formal means and thus computers. Dale et al. [2000] classify these approaches into three groups: *symbolic methods* that represent the language with formal grammars, *empirical methods* that rely on statistical data rather than explicit handcrafted rules, and *connectionist approaches* uses neural networks for processing natural languages. Rather than relying on one particular approach researchers are also trying techniques that involve different approaches. For example, Christiansen and Chater [1985] state that the symbolic approach and the connectionist approach complement each other rather than merely provide alternative representations.

The *Hybrid Connectionist Parser* (HCP) we choose for our work utilizes both a symbolic component and a connectionist component. In the following section, we discuss previous works on spoken language processing. Then, we present the Hybrid Connectionist Parser, which is the basis for the spoken language parser. Following, we describe the combination of HCP and the Phrase Structure Matching Algorithm for spoken language parsing. The paper concludes with a summary of our experimental evaluation of this method using spoken language sentences from the HCRC MapTask corpus [Anderson et al., 1992].

2 Spoken Language Processing

In the following two sections, we give a brief introduction to phenomena typical of spoken language, and then cite relevant work in the area of spoken language processing.

2.1 Spoken Language

Spoken language is marked by the presence of pauses, interjections and repetition of words or phrases. These "features" or "speech repairs" are categorized into three classes:

Fresh starts. The speaker restates his/her previous utterance. For example: "I need to send … let's see … how many boxes can one engine take?"

Modification repairs. The speaker does not restate what he/she utters but replaces or repeats a word or a phrase. For example: "You can carry them both … tow both on the same engine."

Abridged repairs. The speaker stops or pauses while uttering a phrase and then continues what they started to say. For example: "We need to um… manage to get the bananas."

Natural Language parsers, in general, are not capable of handling these erroneous inputs. We need to special capabilities in the parsers to deal with these disfluencies.

Approaches to detecting and repairing disfluencies in spoken language can be categorized according to the kind of information they need and the method using this information. Some researchers use syntactic information based on grammar rules, some use prosodic signals, which indicate an interruption in fluent speech, and others use pattern matching to detect repetitions of words or phrases, or combinations of these methods. We present in the following sections major work done on detecting and/or repairing disfluencies in spoken language.

2.2 Earlier Work on Spoken Language Processing

Hindle [1983] used edit signals (a phonetic signal) to indicate the presence of a repair in speech. He formulated four rules namely *surface copy editor*, *category copy editor*, *stack copy editor* and *handling fresh restarts*. His method had remarkable results in experiments showing only 3% failure. However, assuming the presence of edit signal limited the applicability of such approach.

Dowding et al. [1993] developed a natural language system called GEMINI. They used both syntactic and semantic information in the bottom-up parser to generate structures. Then they use an utterance level parser to derive a semantically correct structure. When the utterance level parser fails, a correction phase is used to detect and repair. Their training set contained 178 repairs and the system was able to detect 89 repairs, and 81 of those repairs were corrected.

Lavie [1996] developed a new parsing algorithm GLR* for spoken language. This parser skipped words to find maximum set of words that form a complete structure. The system did not perform any speech repair detection and correction.

Wermter and Weber [1997] mentioned a flat screening method to handle utterances in their system SCREEN. The system was hybrid in that it used a flat representation for syntactic and semantic analysis and ANNs to deal with ambiguous structures and repairs in the utterances. The system dealt with pauses interjections and repetitions of a word or phrase.

McKelvie [1998] used symbolic rules to implement repairs. He developed a set of grammar rules for fluent speech and later extended it to handle disfluent patterns. However, McKelvie's work does not explicitly show the number of repairs detected and the number of false alarms caused.

Core [1999] designed a parser to work with metarules such as non-interference metarule, editing term metarule and repair metarule. These metarules were used to specify different patterns of speech repair. His system was able to detect 237 out of 495 speech repairs using the word and the part of speech tag information. The correct reparandum start was detected for 187 out of 237 using only word matching and 183 out of 237 using word and part of speech similarities.

Bear et. al [1992] worked on different sources of information to detect and correct repairs in speech. They used pattern matching technique, the syntactic and semantic information of a parser and acoustic information. They conclude that any single source of information is not sufficient to deal with repairs.

Nakatani and Hirschberg [33] determined the repairs by using acoustic and prosodic cues. They tested 202 utterances containing 223 repairs. Of them 186 repairs were detected with a recall of 83.4%. 177 of the repairs are detected with the help of word fragments and pause duration. In another experiment, the word fragments were not considered resulting in 174 repairs with a 78.1% recall.

Shriberg et al. [1997] showed that prosodic information can provide better results to detect repairs in speech than other methods that use lexical information. Shriberg et al. used features like the duration of pause, the duration of vowels, the fundamental frequency, and the signal to noise ratio to detect repairs. These measurements were present in the speech data that were used for training. Finally, they used a decision tree to find the interruption point that follows a filled pause, repetitions, repairs and false starts from other points..

Heeman and Allen [1999] viewed the problem of recognizing part of speech tags, discourse markers, detecting disfluent parts in speech as intertwined. They solved the problem of detecting and correcting repairs, finding utterances ends and discourse markers at the speech recognition phase. With this system, Heeman and Allen solved 66% of speech repairs and they had a precision of 74%.

In contrast to the research described above, we propose a parsing method based on purely syntactic information, the Hybrid Connectionist Parser (HCP) with an added component, the Phrase Structure Matching Algorithm (PSMA). This combined method detects disfluencies (typically corrections and repetitions), based on a parser problem to continue a once started parse tree, and repairs those disfluencies by overlapping partially matching structures of the initial parse-tree and the newly generated sub-tree. The method does not require any pre-marking of the interruption point, or additional phonetic or prosodic information. In the next sections, we describe the basic principles of the HCP, and the augmentation of the HCP with the PSMA, and illustrate how this method is used to parse spoken language.

3 The Hybrid Connectionist Parser

Given a Context-Free Grammar (CFG), the Hybrid Connectionist Parser (HCP) constructs the underlying Artificial Neural Network (ANN) dynamically. The parser works online, incrementally reading words from the input sentence and merging partially completed networks, starting with mini-networks created from the rules of a given CFG. This idea of constructing the ANN dynamically is relatively uncommon but more suitable for processing dynamic structures of unlimited size. Previously, [Elman 1990; Sharkey 1992; Wermter and Weber 1997; Jain and Waibel 1990] used Recurrent Neural Networks (RNN) to imitate the recursive generative capability of natural languages. In this section, we outline the main characteristics of the HCP. A more detailed description can be found in [Kemke 1996, 2001a, 2001b, 2002].

3.1 Representation

The mini-networks in HCP represent the grammatical rules of the given CFG. A mini-network is a two-layer network with one root-node, representing the syntactic categories on the left hand side (LHS) of the grammar rule and one or more child node(s), representing the right hand side (RHS) of the rule. The link weights between each of the child nodes and the root node are $1/n$, if n is the number of child nodes. An example mini-network for the grammar rule $A \rightarrow A_1 A_2 ... A_n$ is shown in Figure 1.
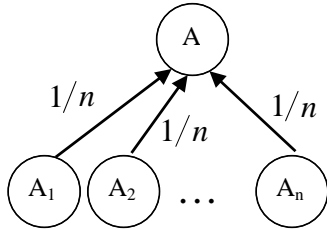

Figure 1: Mini-network representing a grammar rule

In the general case, the units in the network are simple threshold units with the threshold value set to 1.0. Thus, each root node gets somewhat activated, when one or more of its child nodes are fully activated, but gets fully activated (and fires) only when all its children are fully activated. Thus, the HCP simulates a deterministic, incremental parser. In order to make the representation and use of these mini-networks more efficient, we introduced complex mini-networks, which can represent rules with the same LHS but different RHS. This is typical for natural language grammars, which often have slightly different RHS for the same category on the LHS. For this purpose, we use hypothesis nodes. When there are multiple RHS for the same LHS, each of the RHS is represented through a hypothesis node, which is connected to the root node representing the LHS. The link weight for each of these new connections is set to 1.0. All other nodes not belonging to a hypothesis node are connected to it through negative link weights to facilitate mutual inhibition. Thus, the root node for the LHS gets activated, when at least one of the children is fully activated. Figure 2 illustrates this idea for the sample grammar rule:

$VP \rightarrow V\ (H1)\ |\ V\ NP\ (H2)\ |\ V\ PP\ (H3)\ |\ V\ NP\ PP\ (H4)$
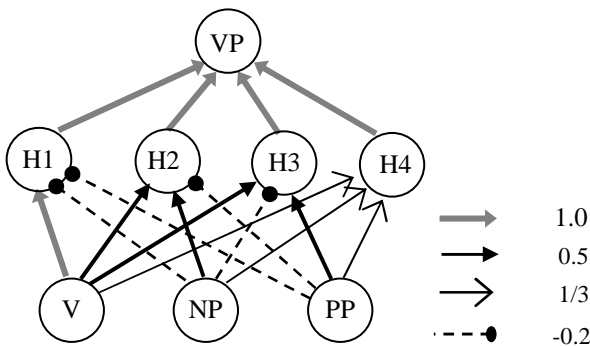

Figure 2: Complex mini-network with hypothesis nodes

3.2 The Algorithm

The HCP starts with a set of mini-networks generated from the context-free grammar. The input is read word by word, and the parser proceeds incrementally. After processing each word from the input sentence, it searches the existing mini-networks to see if this word can be bound to a child node of one of those mini-networks. While searching for such networks, it has to be ensured that the order of the terms in the grammar rules is maintained. In other words, if we are to bind to a node A_k of a network representing the rule $A \rightarrow A_1 A_2 ... A_n$, we need to ensure that the nodes $A_i < A_k$ have already been fully activated and bound. This can be achieved through introducing proper inhibition mechanisms, similar to the ones described above for complex mini-networks. When all the child nodes in a mini-network are activated, the root node of that network becomes fully activated. If the root-node of a network is fully activated, the search for a network, to which this root-node can be bound, starts again. If some input has already been processed, and thus incomplete networks have been generated, the fully activated network can be bound to such an existing, partially activated network, in a process called "merging". The root-node of the later network has to match with the left-most unbound node of the earlier, partially activated network. Alternatively, a new mini-network can be created in parallel, if the root-node is the left-most node on the RHS of the mini-network. After processing all input words, the parse was successful if at least one fully activated network exists, whose root-node is labeled with the start symbol of the grammar. This network corresponds to a complete parse tree for the input sentence. In case of structurally (syntactically) ambiguous sentences, the parser will derive several complete, fully activated networks.

3.3 Special Features

The HCP has some advantages over conventional parsers for CFGs, due to the online, incremental, dynamic construction of an underlying neural network. The algorithm exhibits parallelism in the way nodes are bound in each stage – thus the HCP can be ported to a parallel architecture. It is also possible to represent CFGs in a condensed form for processing by the HCP. One example are the complex mini-networks described above, and optional terms on the RHS of a grammar rule, which could be simply added to a network using a link weight of 0, and thus can be accepted by the parser but are not required.

Some characteristics of this parsing method make it particularly useful for spoken language processing. If an input word or root-node of a fully activated network cannot be bound to an existing network, nor create a new mini-network, the parser cannot continue to derive a complete parse tree. This characteristic is used to detect interruption points and disfluencies in spoken language. The use of graded activation and activation passing enables the parser to maintain information about partially accepted, as well as expected structures, and this can be used to process incomplete and superfluous structures, reflecting repetitions and corrections in spoken language.

4 Augmenting the HCP with the PSMA

We augmented the HCP with a *Phrase Structure Matching Algorithm* (PSMA), extending the concept of the "Graph Matching Algorithm" by Kemke [2001b]. We developed the PSMA to correct disfluent structures present in an utterance by detecting structural similarities between the reparandum and its alteration. The PSMA is triggered, when an incomplete structure is detected during parsing with the HCP. This leads to an inconsistent state of the parsing process, which cannot be continued, since a newly generated sub-tree cannot be merged into the initial incomplete parse tree, and can also not otherwise be expanded. The PSMA takes the partial parses generated by the HCP as input and corrects the disfluency by trying to find a structural match between the new sub-tree and the initial parse tree. The following example illustrates the idea.

The sentence *"I have a ... I have got a picket fence"* is an example of a fresh start. The speaker meant to say *"I have got a picket fence"*, but inadvertently started with *"I have a"* and then starts the sentence over with *"I have got a picket fence."*

The syntactic parsing of the initial, false start:

S [NP[PRP[*I*]] VP[HV[*have*]] NP[DT[*a*] NOM[ ]]]]

reveals that the parser is trying to recognize as next completed structure a Noun Phrase NP starting with *"a"* and expects as next category a Nominal NOM (Figure 3).

In this parsing situation, a nominal is thus expected as root node of the next completed sub-tree. The HCP continues to parse and derives a new, fully activated sub-tree, which is in this example a complete sentence structure. The algorithm detects that a nominal is expected but it is not present in the expected position in the parsed input sentence. Instead, the root node is labeled with the sentence symbol S. Thus, it is assumed that a disfluency has occurred and this point in the parse tree is interpreted as the interruption point. The PSMA takes the complete sentence structure returned by the HCP:

S [NP [PRP [*I*]] VP [HV[have] VP[VBP[got]

NP [DT[*a*] NOM [*picket fence*]]]]

and tries to match the root node S of this structure with previous incomplete structure shown in Figure 3.
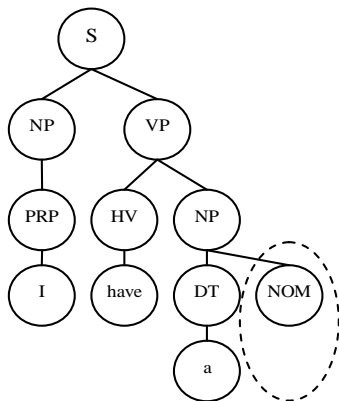

Figure 3: Partial parse of "I have a"

The PSMA searches backward through the initial, incomplete parse tree, which contains the unbound, not activated node (NOM in Figure 3). When it can confirm a significant structural similarity, it overlaps the matching structures and thus creates a new, corrected parse tree. At this point, the previous structures are discarded and the new, combined structure is retained.

To find a match between the new sub-structure and the initial, incomplete one, the PSMA starts at the right bottom of the initial parse tree and step-by-step moves towards its parent nodes, in case of failure of a match with the label of the root node of the new structure. In the example, the PSMA starts the comparison with the node "a" and its category DT in the initial parse tree. The root node of the new structure is S which does not match with DT. The PSMA moves to the next higher level NP, which also fails to match. The third attempt, to match with VP, is also unsuccessful. Finally, the PSMA moves to the highest node S and finds a match of the initial structure with the new, complete structure. The two structures are then merged, through an overlapping process, in which the newer generated structure precedes over the initial, earlier created structure. This reflects the assumption that speakers correct themselves in later parts of the utterance, and thus the later parts override earlier parts of the input. The result is a new, complete parse tree (Figure 4). The comparison of the two structures through top down traversal confirms a structural match to a high degree. This makes the PSMA a robust and reliable method to correct the found disfluency.
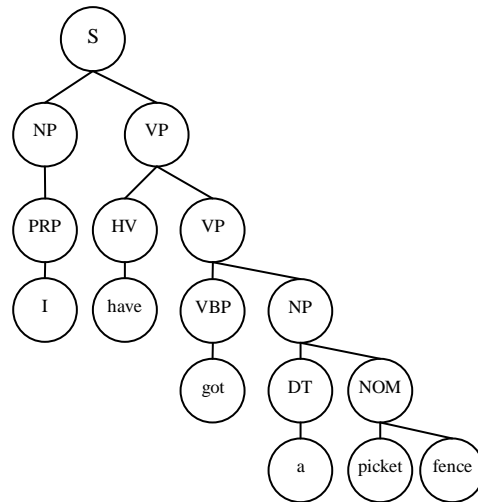

Figure 4: Full parse of "I have got a picket fence"

In the example, we see that the PSMA has detected the interruption point based on the fact that the parse could not be continued as expected. The search and match procedure found a substantial overlap and proper replacement for the incorrect structure. The PSMA does not expect, however, that a phrase is repeated exactly to correct a disfluency. A successful match starting of a parent node in the initial parse tree and the root node of the new sub-tree would be sufficient to allow the PSMA to merge the two structures.

4 Evaluation

We tested the method with the HCRC Map Task Corpus by Anderson et al. [1992]. This corpus contains transcribed dialogues regarding path finding in a terrain, between an instruction giver with a route marked map and an instruction follower with an unmarked map. Since it is based on spontaneous spoken language, the Map Task Corpus contains disfluencies typical for natural conversation. An advantage of the Map Task Corpus is the restriction to a particular domain of conversation, which made the grammar and lexicon development for our tests with the HCP and PSMA relatively fast and easy.

4.1 Evaluation Metrics

The merit of the speech repair tool is calculated using the measures of *precision* and *recall* – these two metrics have been used earlier [Core, 1999; Heeman and Allen, 1999; Nakatani and Hirschberg, 1993] and are a standard measurement for disfluency detection.

Precision

The precision is calculated by $Tp/(Tp+Fp)$, where Tp (*True positive*) is the total number of repairs that are detected as repairs and Fp (*False positive*) is the total number of false repairs that are detected as repairs.

Recall

The recall is calculated by $Tp/(Tp+Fn)$, where Fn (*False negative*) is the total number of repairs that are not detected as repairs. In our tests based on the HCRH Map Task Corpus, all examples contained at least one disfluency. We thus did not calculate the precision, as there were no false positives in the test set.

4.2 Experiments

We implemented the PSMA integrated with the HCP in two different ways: using a post-processing method, in which the PSMA starts to work only after the HCP has created all possible structures, and an incremental method, in which the PSMA is activated as soon as parsing cannot be continued and a disfluency is assumed.

In our first set of experiments we let HCP to finish parsing of the full utterance. The PSMA then takes over trying to fix the disfluencies marked by interruption points. We conducted the experiment in three stages. For the first stage, we had 55 test sentences with a single repair instance. The parser was able to correct 38 of them. For the second stage we expanded the grammar and ran tests on 71 sentences and the parser corrected 28 of them. In the third stage, we increased the number of repair instances. Of the 98 sentences we tested, the parser corrected 42 of them.

In the first set of experiments, it turned out that the post-processing method had problems to correct utterances with more than two repair instances. We subsequently tested an incremental version of the combined HCP and PSMA, which brought about better results.

In this second set of experiments, the PSMA started as soon as a disfluency was detected. After receiving a new complete sub-tree from the HCP, the PSMA started immediately to search and match with the initial tree, in order to perform an overlapping of structures to correct the disfluency in the utterance. In the incremental processing tests, the PSMA detected 89 out of 121 repair instances with a recall rate of 73.5%. Of these 89 repairs, the PSMA was successfully able to correct 80 of them.

5 Future Work

As our results have shown, the incremental processing method has a promising prospect. We would like to extend our work with this method in the following directions.

Parallelizing the HCP. The HCP is implicitly parallel in nature. When each node is activated, we have the choice of binding it to (several) other nodes, or to create a new mini-network for it. These options are mutually exclusive, i.e. they cannot be present in the same parse tree, and could thus be executed in parallel. We are considering a parallel implementation of the HCP to speed up the computation. Although speeding up the parser does not improve the recall rate, it makes the parser more applicable for practical use.

Removing useless mini-networks. In our implementation, the HCP can merge and create new mini-networks as needed. None of these mini-networks were removed during run time, even though they might not have been needed anymore after a certain point in time. After processing sufficient input incrementally, we can detect, when a partial parse tree / mini-network may no longer be useful. Removing the mini-network can reduce excessive memory requirements and at the same time speed up the whole computation.

Using a refined and extended grammar. The choice of grammar can be a determining factor in robust parsing. Fine tuning the grammar rules can improve the repair detection capability of the parser. Verb subcategorization [Jurafsky and Martin 2000], for example, can be employed to refine the grammar rules. We would like to use a more elaborate grammar notation, and do further tests with a larger set of grammar rules and an expanded lexicon, to see how the spoken language parser improves and/or scales up.

6 Conclusion

We have extended a Hybrid Connectionist Parser (HCP) with a Phrase Structure Matching Algorithm (PSMA) in order to perform parsing of spoken language. This approach combines the use of symbolic and connectionist methods and features, in particular, the use of symbolic features like syntactic categories as nodes, and neural network features like graded activation and threshold dependent processing in the HCP. The neural network features are suitable to deal with incomplete and inconsistent structures, and thus it deemed useful to employ the parser for spoken language processing. Typical phenomena of spoken language are disfluencies, in particular corrections and modifications of utterances during spontaneous speech. The HCP detects such disfluencies, and the PSMA repairs them based on a struc-

tural match and overlapping of comparable structures. The experimental results we obtained with a detection and correction degree of approximately 70% are good, for a purely syntactic method, which does not require any prosodic, phonetic or other information.

The method, however, seems to depend on the tuning of the grammar, which has to accurately reflect the linguistic input structures, and thus scalability and performance should be improved through a more elaborate grammatical notation and potentially by adding further information derived from the speech signal, like hesitations, pauses or fillers. Overall, based on the first experimental results, the approach has a promising prospect.

Acknowledgments

References

[Aho et al., 1986] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques and Tools.* Pearson Education, 1986.

[Anderson et al., 1992] A. Anderson, M. Bader, E. Bard, E. Boyle, G.M. Doherty, S. Garrod, S. Isard, J. Kowtko, J. McAllister, J. Miller, C. Sotillo, H.S. Thompson and R. Weinert. The HCRC Map Task Corpus. *Language and Speech*. 34: 351–366, 1992.

[Bear et al., 1992] J. Bear, J. Dowding, and E. Shriberg. Integrating Multiple Knowledge Sources for Detection and Correction of Repairs in Human-Computer Dialog. *Proc. 30th Annual Meeting of the Association for Computational Liguistics*, 56-63, 1992.

[Core, 1999] M. G. Core. Dialog Parsing: From Speech Repairs to Speech Acts. PhD thesis, University of Rochester, New York, 1999.

[Dale et al., 2000] R. Dale, H. Moisl, H. Somers, and H. L. Somers (Eds.). *Handbook of Natural Language Processing.* Marcel Dekker, New York, 2000.

[Dowding et al., 1993] J. Dowding, J. M. Gawron, D. Appelt, J. Bear, J. Cherny, R. Moore and D. Moran. Gemini: A Natural Language System for Spoken Language Understanding. *Proc. 31st Annual Meeting of the Association for Computational Linguistics*, 54-61, Columbus, Ohio, 1993.

[Elman, 1990] J. L. Elman. Finding Structure in Time. *Cognitive Science*, 14: 179–211, 1990.

[Heeman and Allen, 1999] P. Heeman and J. Allen. Speech Repairs, Intonational Phrases and Discourse Markers, Modelling Speakers Utterance in Spoken Dialogue. *Computational Linguistics*, 25:527-571, December 1999.

[Hindle, 1983] D. Hindle. Deterministic Parsing of Syntactic Non-fluencies. *Proc. 21st Annual Meeting of the Association of Computational Linguistics*, 123–128, 1983.

[Jain and Waibel, 1990] A. N. Jain and A. H. Waibel. Incremental Parsing by Modular Recurrent Connectionist Networks. In D. S. Touretzky (ed.): *Advances in Neural Information Processing Systems 2*, Morgan Kaufmann, San Mateo, CA, 1990.

[Jurafsky and Martin, 2000] D. Jurafsky and J. H. Martin. *Speech and Language Processing*. Prentice Hall, 2000.

[Kemke, 1996] C. Kemke. A Hybrid Approach to Natural Language Parsing. von der Malsburg, von Seelen, Vorbrueggen, Sendhoff (Eds.): *Artificial Neural Networks, Proc. ICANN'96*, 875–880, Bochum, Germany, 1996.

[Kemke, 2001a] C. Kemke, *Connectionist Parsing with Dynamic Neural Networks – or: Can Neural Networks make Chomsky Happy?*, Technical Report MCCS-01-327, CRL, NM State University, 2001.

[Kemke, 2001b] C. Kemke. Generative Connenctionist Parsing with Dynamic Neural Networks. *Proc. 2nd Workshop on Natural Language Processing and Neural Networks*, 31-37, Tokyo, Japan, 2001.

[Kemke, 2002] C. Kemke. A Constructive Approach to Parsing with Neural Networks – The Hybrid Connectionist Parsing Method. *Advances in Artificial Intelligence*, LNAI-2338, 310-318, Springer, 2002.

[Lavie, 1996] A. Lavie. GLR*: A Robust Grammar-Focused Parser for Spontaneously Spoken Language. Ph.D. Thesis, Carnegie Mellon University, 1996.

[Manian, 2005] V. Manian. *Integrating a Connectionist parser and a Graph Matching Algorithm for Handling Disfluenciesnin Spoken Language Processing*. M.Sc. Thesis, University of Manitoba, 2005.

[McKelvie, 1998] D. McKelvie. *The Syntax of Disfluency in Spontaneous Spoken Language.* Human Communications Research Centre, HCRC/RP-95, Edinburgh, May 1998.

[Nakatani and Hirschberg, 1993] C. Nakatani and J. Hirschberg. A Speech-First Model for Repair Detection and Correction. *Proc. 31st Annual Meeting of the Association for Computational Linguistics*, 46–53, 1993.

[Sharkey, 1992] N. Sharkey. *Connectionist Natural Language Processing*. Intellect, Oxford, England, 1992.

[Shriberg, 1997] E. Shriberg, R. Bates, and A. Stolcke. A Prosody-only Decision-Tree Model for Disfluency Detection. *Proc. 5th European Conference on Speech Communication and Technology*, 2383-2386, Rhodes, Greece, 1997.

[Wermter and Weber, 1997] S. Wermter and V. Weber. SCREEN: Learning a Flat Syntactic and Semantic Spoken Language Analysis Using Artificial Neural Networks. *Journal of Artificial Intelligence Research*, 6:35–85, 1997.