Towards the Development of a Hybrid Parser for Natural Languages

Sardar F. Jaf[1] and Allan Ramsay[2]

1 **School of Computer Science, The University of Manchester**
2.46 Kilburn Building, Oxford Road, Manchester, M13 9PL, United Kingdom
sardar.jaf@manchester.ac.uk

2 **School of Computer Science, The University of Manchester**
2.21 Kilburn Building, Oxford Road, Manchester, M13 9PL, United Kingdom
allan.ramsay@manchester.ac.uk

Abstract

In order to understand natural languages, we have to be able to determine the relations between words, in other words we have to be able to 'parse' the input text. This is a difficult task, especially for Arabic, which has a number of properties that make it particularly difficult to handle.

There are two approaches to parsing natural languages: grammar-driven and data-driven. Each of these approaches poses its own set of problems, which we discuss in this paper. The goal of our work is to produce a hybrid parser, which retains the advantages of the data-driven approach but is guided by grammar rules in order to produce more accurate output. This work consists of two stages: the first stage is to develop a baseline data-driven parser, which is guided by a machine learning algorithm for establishing dependency relations between words. The second stage is to integrate grammar rules into the baseline parser. In this paper, we describe the first stage of our work, which is now implemented, and a number of experiments that have been conducted on this parser. We also discuss the result of these experiments and highlight the different factors that are affecting parsing speed and the correctness of the parser results.

1 Introduction

Processing human languages to determine the structural relations between words is called *parsing* in Computational Linguistics (CL) [1, p.63]. Parsing is one of the core components of many Natural Language Processing applications [2], such as: Machine Translation Systems, Tutoring and Speech Recognition Systems, Information Retrieval Systems, and Question and Answering Systems.

Producing a comprehensive parser is a challenging task due to language ambiguities [13], which is caused by such factors as multiple interpretations of words, flexibility of word order, and missing items. Hence, adequate parsing systems are often unavailable for natural languages, especially for languages with complex structures such as Arabic [18, p.82].

It is desirable that parsers have three main features - efficiency, robustness and accuracy. The efficiency of parsers is concerned with consuming as little time as possible, the robustness is for enhancing the system's ability to cope with agrammatical inputs, and the accuracy is required to ensure that the results produced are accurate. However, it is not possible to achieve all three features at once [7]. Some parsers have traded off accuracy for efficiency, while others sacrificed efficiency for robustness [18]. The goal of our work is to optimise speed and accuracy while maintaining a reasonable level of robustness. We aim to test our parser on Arabic because Arabic presents a number of challenges which make it hard to parse, and hence it will act as a rigorous test-bed for our approach.

2013 Imperial College Computing Student Workshop (ICCSW'13).
Editors: Andrew V. Jones, Nicholas Ng; pp. 49–56

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

2 Parsing natural language approaches

There are two different kinds of approaches to parsing natural languages: grammar-driven approaches, and data-driven approaches. In grammar-driven approaches, the parser depends on grammatical rules suitably specified in accordance with some linguistic theory. while in data-driven approaches, the parser mainly depends patterns extracted from preprocessed data, such as treebank data. In this section, we describe each of these approaches.

2.1 Grammar-driven approach

Grammar-driven parsing uses a formal grammar G, which defines a formal language $L(G)$ for an alphabet A. A grammar G is used as a system for generating strings over A; The language $L(G)$ that is defined by G is the set of all strings x that can be generated by G.

The assumption in grammar-driven parsing is that $L(G)$ is an approximation of the language L. However, the formal grammars that have been developed to date fail to meet this assumption [7]. Having said that, the principles behind grammar-driven approaches should not be neglected because the linguistic theories advancement may subsequently lead to better approximations, but, in the mean time it creates some practical problems for grammar-driven parsing.

Robustness in parsing natural languages is the capacity of parsers to analyse as many input strings as possible. A parser is considered robust if it can analyse a large proportion of the sentences of the language in question. One of the major problems associated with grammar-driven parsing is robustness. This problem occurs because some input strings in a given sentence may not exist in the formal language $L(G)$ that is defined by the grammar G. Generally, there are two kinds of robustness problem: (i) *coverage problem* and (ii) *robustness proper*.

The coverage problem normally occurs when an input string x is not part of the formal language $L(G)$ even though it is grammatically a legitimate sentence of L. Hence, at least in theory, it should exist in $L(G)$, because $L(G)$ is an approximation of L, but the sentence may not exist in $L(G)$ because it is not covered by G. On the other hand, robustness proper occurs when an input string x is understandable by the speaker of L but it is not part of the language L and so it should not exist in $L(G)$ either. Robustness proper actually occurs if a word is misspelled or if material is omitted or agreement constraints are violated.

The robustness problem can be solved by relaxing grammar constraints in parsers [4]. But, relaxing grammar constraints could result in many analyses becoming available for a given input text, hence it leads to the problem of *disambiguation*, which is a major problem for parsing natural language sentences, because applications that depend on the output of parsing systems typically require a small number of analyses (preferably just one analysis) for a given input text x. Having many analyses for a given x means that parsers will have to consume more time and resources exploring these analyses which create a problem with efficiency and also may result in selecting an incorrect analyses, which may affect accuracy.

The problem of robustness and disambiguating aggravates the problem of *accuracy*. Robustness attempts to produce analyses for input strings x that may not exist in $L(G)$ that is defined by G, while disambiguation insists on removing extra analyses that are assigned to x by G. These moves by any parsers may reduce the chance that an x that is part of a text is given the correct analysis by parsers. Hence, a joint optimisation between robustness, disambiguating and accuracy is necessary, or at least they are prioritised and the trade off between them is carefully chosen based on the nature of the parsing system.

Joint optimisation between robustness, disambiguating and accuracy triggers the problem

of *efficiency*. The problems with efficiency in grammar-driven approach depend mainly on the expressivity and the complexity of the formal language that is used for parser [7]. The most commonly used algorithms are at least N^3 in the length of the input text, which is daunting in situations where sentences may contain tens or even hundreds of words.

2.2 Data-driven parsing

In data-driven approaches, an inductive mechanism is used for mapping from input strings to output analyses. This mechanism, that is applied to a text sample $T_t = (x_1,...,x_n)$ from the language L to be analysed, makes the abstract problem of data-driven approach that is used to approximate text parsing a problem of inductive inference.

According to [7], data-driven parsers consist of three main components: (i) permissible analyses for sentences in the language L as defined by a formal model M. (ii) a sample of text $T_t = (x_1,...,x_n)$ from L with or without the correct analyses $A_t = (y_1,...,y_n)$, and (iii) actual analyses for the sentences $T = (x_1,...,x_n)$ in L is defined by an inductive inference scheme I, which is relative to model M and T_t and possibly A_t. Based on these components, a model M could represent a formal grammar G for restricting string representations to strings of the language L.

Training data, that is used in this approach, is a sample of text T_t. This could be raw data or an annotated treebank of the language L, where treebanks may or may not be annotated with representations satisfying the constraints of M. If the sample data is a treebank then a form of *supervised machine learning* is used for inductive learning because, according to the treebank annotation, the correct analyses of an input string x_i is in the sequence of analyses $A_t = (y_1,...,y_n)$. While *unsupervised machine learning* is used if the sample data is raw text because no sequence of analyses will exist in T.

Similarly to grammar-driven approaches, data-driven approaches are also based on the approximation that the formal language L is an approximation of the language L, but, this approximation is different in data-driven parsing because it is based on inductive inference from a finite sample $T_t = (x_1,...,x_n)$ to the infinite language L

The problem of robustness also exists in data-driven approach, robustness here depends on the formal model M properties as well as the inference scheme I which are used for processing new sentences. According to [7], in most existing data-driven parsers any input strings x are assigned at least one analysis, which means that data-driven parsers are highly robust. However, the extreme robustness of data-driven parsers means that they will assign analyses that are probably not in the language L.

Furthermore, the problem of disambiguation can be even more severe in data-driven parsers because the improved robustness is the result of extreme constraints relaxation. but, this is compensated by the fact that the inductive inference scheme I provides a mechanism for disambiguation, by associating a score with each analysis intended to reflect some optimality criterion, or, by implicitly maximising this criterion in a deterministic selection.

Regarding the problem of *efficiency*, it is argued that data-driven approaches is superior to grammar-driven approaches [7], but it is often at the expense of less accurate output [8]

3 Arabic

Ambiguity is a central problem in natural language parsing [11, 13]. Arabic contains many complexities and subtleties [10], which lead to even greater potential for ambiguities than is present with other languages. In the following sections we briefly highlight some of the main sources of ambiguities in Arabic.

3.1 Missing diacritics

Arabic diacritics are short strokes placed above or below consonants. There are three sets of diacritics: (i) Short vowels are symbols placed either above or below letters, such as /a/ sound as اَ, /u/ sound as اُ, or /i/ sound as اِ. (ii) Double case endings are also vowels and they suggest indefiniteness and are manifested in the form of case marking or in conjunction with case marking. these are placed on the final letter of a word, such as a /aN/ sound as in اً, /uN/ sound as in اٌ or /iN/ sound as in اٍ. (iii) Syllabification marks are placed above Arabic letters denoting the doubleing of the consonant, they are usually combined with short vowels. There are two types of syllabifications: (i) is called shadda written as a gemination marks as رَّ and (ii) is called sukun, which is a small circle as اْ, and it marks the boundaries between syllables, end of verbs, or it indicates that the word does not contain vowels.

Arabic texts without diacritics are ambiguous. Many words with different diacritic patterns appear identical in a diacriticless settings but they may have different syntactic roles [6]. e.g. the word علم *alam* can have many roles when diacritised, such as: noun as in عِلْمٌ *'ilmuN* "knowledge", transitive verb as in عُلِّمَ *'u-llima* "is taught" or intransitive verb as in عُلِمَ *'ulima* "is known". Written Modern Standard Arabic (MSA) generally omits diacritics, which leads to widespread lexical ambiguity of the kind shown [17].

3.2 Free word order

Arabic has a high degree of syntactic flexibility [10]. The canonical order of an Arabic sentence is VSO. But, a range of other word orders such as VOS, SVO and OVS are also possible [3], which is a source of ambiguities in Arabic [14, p.179]. It is not easy to distinguish between the nominative and accusative cases when word orders are changed, i.e. it is hard to identify the subject and object of a sentence, for example, in the sentence اَحمَد يَحتَرم عَلي *a.hmad ya.hatarm 'aly* "Ahmed respects Aly" it is clear that Ahmed is the subject in the sentence and Aly is the object. However, reordering the words in the same sentence as يَحتَرم أَحمَد عَلي *ya.htarm 'a.hmad 'aly* "respects Ahmed Aly" means that the subject could be either Ahmed or Aly. This results in structural ambiguity.

3.3 Arabic clitics

Clitics are morphemes [1] that possess the syntactic characteristics of a word, but, they are morphologically bound to other words [5]. Arabic clitics could be attached to the start or end of words, this often alters their formation, for example they could alter word types from noun to verbs, or even changes the verb type from transitive to intransitive [17]. For example, conjunctions in Arabic can often appear as clitics and modify Arabic verbs. For instance, the sentence وليَّهُم عَليٌ في المسآلة *wali-yyahum'alyuN fy Al mas'Ala* "Ali is the leader in their situation" where وَلِيَّهُمْ *walyahum* "their leader", which is a noun, is ambiguous because the letters و /w/ and ل /l/ could be clitics attached to the word لِيَّهُم *li-yyahum* "take charge" and can modify these words into verbs, as in the sentence وليَّهُم عَليٌ في المسآلة *wa li -yyahum 'alyuN fy Al mas'AlT* "and Ali to take charge of the situation", where the word is a verb.

[1] A morpheme is a small grammatical unit of a language.

3.4 Noun multi-functionality

It is difficult to define Arabic nouns in comparison to its verbs because they encompass a wide range of categories. One of the reasons that Arabic nouns create ambiguities is that some nouns are derived from verbs, and they can function as verbs sometimes [14]. e.g., البَحث *Alba.h_t* "search" can function as a noun as in اِستَمَرَّ التِلميذُ في البَحثِ لِلجَامِعَة *istama-rra Altilmy_du fy Alba.h_ti liljAmi'aT* "the student continued in his research for the university", and as a verb as in اِستَمَرَّ التِلميذُ في البَحثِ عَن الجَامِعَة *istama-rra Altilmy_du fy Alba.h_ti 'an AljAmi'aT* "the student continued searching for the university"

3.5 Arabic pro-drop

In pro-drop, the subject of a sentence could be omitted if the verb's agreement features are rich enough to recover its content [15]. Arabic verbs recover missing subjects by conjugating themselves to indicate the gender, number and person of the omitted pronoun subject [14]. Arabic pronouns may be omitted if the verb can recover them, as in, اكلت الدجاجة *Akalat Al dajAjT* "ate the chicken". The verb اكلت *Akalat* "ate" indicates that the missing subject is a singular, feminine and third person pronoun. In Arabic, verbs can be transitive and intransitive when a pronoun is dropped. It is not clear from the above sentence that the *NP* الدجاجة *Al dajAjT* "the chicken" following the verb اكلت *Akalat* "ate" is the subject. The sentence would mean *the chicken was eaten* and the verb اكلت *Akalat* "ate" is intransitive if the *NP* is the subject. But, the sentence would mean *she ate the chicken* and the verb اكلت *Akalat* "ate" is transitive if the *NP* is the object of the verb and the subject is an omitted pronoun (as "she"). Hence, due to pro-drops, parsers generate different structural analysis.

4 Work to date

We have implemented a dynamic programming version of shift-reduce parsing algorithm [1, p.368]. The dynamic programming algorithm stores partial parse analyses that are generated by the shift-reduce parsing algorithm. The shift-reduce parser has two data structures, queue and stack which contain items with the following features: (i) Part of Speech (POS), (ii) word form, (iii) word span within the sentence and (iv) words actual position in the sentence. Three operations are performed on queues and stacks: (i) *shift*, (ii) *left-reduce*, and (iii) *right-reduce*, where each of these operations results in a new queue and a new stack which are stored in a database as partial parse analyses. Shift operation moves the first token from the queue to the top of the stack producing a new queue and a new stack. *left-reduce* and *right-reduce* operations perform two main operations: First, a parent-daughter (dependency) relation between the first item on the queue and one of the items on the stack is determined. Second, the daughter (dependent) from the dependency relation is removed and the parent's start and end position is modified to create a span covering the position of the daughter within the parent's start and end position. Left-reduce operation makes a token from the stack a dependency parent of the token at the beginning of the queue. Right-reduce operation makes the token at the beginning of the queue a dependency parent of a token on the stack. These three operations continue until the queue is empty and a stack with one token is found in which the token's start and end position covers the whole sentence length in question, in which case the parse is considered successful. Otherwise, is unsuccessful.

We integrated an *oracle* in the parser for determining parser's action. We ask the oracle whether we should do a shift, left-reduce or right-reduce operation. If it suggests more than

one operations then we add them onto an agenda. The parser will then work through the agenda to explore the suggested actions specified in the agenda. The oracle determines the parser type, the parser becomes a grammar-driven parser if the oracle is a formal grammar, and it becomes a data-driven parser if the oracle is a set of inferred rules, which are extracted from a treebank. At this stage, the oracle is a set of inferred rules, which are extracted from a dependency treebank using a decision tree algorithm. Data-points for the decision tree algorithm are state:action pairs.

5 Preliminary results

We have conducted some preliminary tests on the the data-driven parser using a combination of techniques. The parser, by construction, is efficient and robust, because we provide it with shift, left-reduce or right-reduce action at each parse step, which deterministically lead to some analysis. However, the efficiency and robustness of the parser comes at the expense of its accuracy. We identified a number of factors that may potentially affect the accuracy, the main factors are: (i) the size of the queues and stacks used for parser training (where the Penn Arabic Treebank is used for training and testing the parser), (ii) the length of sentences, and (iii) the size of training data. As shown in Table 1, having one item on the queue affected the accuracy greatly, while variations in the stack size have less effects on the accuracy. The parser accuracy improved significantly when we trained it with queues containing more than one item. We also tested the parser on various sentence lengths. We divided the sentences in the test data into different sets of data size ranging from eight words to one hundred or more words. The highest accuracy is obtained with sentences that have less than twenty five words, as shown in Table 2. Using smaller training size than testing size also affects the accuracy. Larger training data produce better accuracy as can be seen from Table 3. The current state-of-the-art data-driven parser achieve %85 accuracy by combining a weighted combination of two state-of-the-art parsers (MaltParser and MSTParser) [12]. Our accuracy is below state-of-the-art because, at this preliminary stage, we included a limited number of features on that queue and stack which are used for parser training.

Table 1 Parser training with various number of items on queue and stack.

Queue items	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4
Stack items	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Results (%)	25	25	24	10	57	57	57	57	60	60	60	60	60	61	61	61

Table 2 Parsing with various sentence length.

No. words in Sentences	8 to 25	26 to 50	51 to 75	76 to 100	over 100
Results (%)	66	59	59	58	57

Table 3 Parsing 500 sentences with training data ranging from 125 to 5000 sentences.

Training data	125	250	500	1000	2000	3000	4000	5000
Results (%)	55	57	58	59	61	60	60	60

5.1 Related work

There is an increasing interest for combining various parsing algorithms. Some work involved combining state-of-art dependency data-driven parsers, such as MaltParser [7] and MSTParser [18], while some other works focused on combing data-driven and grammar-driven parsers. The latter is the type of work that is more relevant to the work we present in this paper.

[16] combined a grammar-driven parser (XLE system), which is based on Lexical Functional Grammar (LFG), with a data-driven parser (MaltParser). In their approach, they supply a data-driven parser with outputs from a grammar-driven parser. The grammar-driven parser outputs phrase structured trees containing grammatical features. They convert the output of their XLE platform to dependency trees in order to have two parallel versions of the treebank: (i) a gold standard treebank, (ii) and a dependency treebank by converting the XLE system output which contains additional grammatical features. They extend the gold standard treebank with additional information from the corresponding LFG analyses. MaltPaser is then trained on the enhanced gold standard treebank. Their results showed a small improvement in accuracy when applied to English and German.

A similar work in this area is conducted by [9]. They constrain a Head-driven Phrase Structure Grammar (HPSG) parser with outputs from a data-driven parser. HPSG parsers use a small number of schemas for explaining general construction rules, and a large number of lexical entries for expressing word-specific syntactic and semantic constraints. HPSG parse trees are converted to Context Free Grammar style (CFG-style) trees and a dependency treebank is then extracted from the CFG-style trees. the dependency treebank is used for training a dependency data-driven parser, such as MaltPaser and MSTParser. Outputs from data-driven parsers are used to constrain the HPSG parser. During HPSG parsing process, the lexical head of each partial parse tree is stored and in each schema application the head child is determined. Having such information about the head child and the lexical head, the dependency produced by the schema application is identified and whether the schema application violates the dependencies in the dependency treebank is checked. The HPSG parser is forced to produce parse trees that are consistent with the dependency trees. This approach is tested on English and some improvements in accuracy was achieved.

5.2 Next stage

The next stage is to implement a hybrid parser by integrating grammatical rules into this parser. The aim is to constrain our data-drive parser with features from grammar-driven approaches. In order to produce a hybrid parser, we make the oracle a weighted combination of grammar W and decision tree D. We fix D to be 1 (or 1 times whatever probability we can extract from it). If $W = 0$ then the parser is a data-driven parser. If $W = N{+}1$, where N is the length of sentence, then the parser becomes a grammar-driven parser. Intermediate values produce combinations. Low value for W will be fairly fast but prone to producing non-conforming trees, high value for W will be slow but trees will tend to be legal.

6 Conclusion

Problems associated with using grammar-driven approaches and data-driven approaches are discussed in this paper. The main structural complexities of Arabic are identified and described in section 3. The first stage of our approach to hybrid parsing is explained and the next stage for full hybrid parsing implementation is established. Preliminary results for the parser is included in section 5. The parser is tested on Arabic because it is a complex

language, compare to some other languages, hence it provides a rigorous test-bed. Finally, two various related works in hybrid parsing approaches for natural language processing is identified and briefly described.

References

1 Alfred, V., Aho and Jeffery, D., Ullman. *The Theory of Parsing, Translation, and Compiling*, volume 1. Prentic-Hall, 1972.

2 Ali, Farghaly and Khaled, Shaalan. Arabic Natural Language Processing: Challenges and Solutions. *ACM Computing Surveys*, 8(4):1–22, 2009.

3 Allan, Ramsay and Hanady, Mansour. Local Constraints on Arabic Word Order. In *Proceedings of the 5th international conference on Advances in Natural Language Processing*, FinTAL'06, pages 447–457, Berlin, Heidelberg, 2006. Springer-Verlag.

4 Christa, Samuelsson and Mats, Wirèn. *Parsing techniques.* Marcel Dekker, 2000.

5 David, Crystal. *A First Dictionary of Linguistics and Phonetics.* Deutsch, London, 1980.

6 Imed, Zitouni and Jaffery, S., Sorensen and Ruhi, Sarikaya. Maximum Entropy Based Restoration of Arabic Diacritics. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 577–584, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

7 Joakim, Nivre. Inductive Dependency Parsing. Springer, 2006.

8 Ronald M. Kaplan, Stefan Riezler, Tracy H. King, John T. Maxwell III, Alexander Vasserman, and Crouch Richard. Speed and accuracy in shallow and deep stochastic parsing. In *Proceedings of Human Langauge Technology and the Conference of the North American Chapter of teh Association for Computational Linguistics (HLT-NAACL*, pages 97–104, 2004.

9 Kenji, Sagae and Yusuke, Miyao. Hpsg parsing with shallow dependency constraints. In *In Proc. ACL 2007*, 2007.

10 Kevin, Daimi. Identifying Syntactic Ambiguities in Single-parse Arabic Sentence. 35(3):333–349, 2001.

11 Marlyse, Baptista. On the Nature of Pro-drop in Capeverdean Creole. 5:3–17, 1995.

12 Maytham, Alabbas and Allan, Ramsay. Evaluation of combining data-driven dependency parsers for arabic. In *Proceedings of the 5th Language & Technology Conference Human Language Technologies as a Challenge for Computer Science and Linguistics*, pages 546–550, 2011.

13 Michael, Collins. Head-Driven Statistical Models for Natural Language Parsing. *Comput. Linguist.*, 29(4):589–637, 2003.

14 Mohammed, A., Attia. *Handling Arabic Morphological and Syntactic Ambiguities within the LFG Framework with a View to Machine Translation.* PhD Thesis, School of Languages, Linguistics and Cultures, Manchester University, 2008.

15 Noam, Chomsky. *Lectures on Goverment and Binding.* Dordrecht: Foris, 1981.

16 Øvrelid, Lilja and Kuhn, Jonas and Spreyer, Kathrin. Improving data-driven dependency parsing using large-scale lfg grammars. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, ACLShort '09, pages 37–40, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

17 Rani, Nelken and Stuart, M., Shieber. Arabic Diacritization Using Weighted Finite-State Transducers. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, Semitic '05, pages 79–86, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

18 Ryan, MacDonald. *Discriminative Learning and Spanning Tree Algorithms for Dependency Parsing.* PhD Thesis, Computer and Information Science, the University of Pennsylvania, 2006.