

Technical Report

Afribooms: Technical Report

1.0.0

D.R. van Niekerk

DOCUMENT INFORMATION

Filename	TechReport.AFRB.FinalReport.1.0.0.DvN.20140903.pdf
Type	Technical Report
Title	Afribooms: Technical Report
Version	1.0.0
Source Author(s)	D.R. van Niekerk
Commentator(s)	
Status	
Date	2014-09-03
Pages	6 (excluding attached document in Appendix A)
Security Classification	

Distribution	
Key Words	AFRIKAANS; DEPENDENCY GRAMMAR; SENTENCE PARSING
Project	Afribooms

DOCUMENT HISTORY

Version	Date	Status	Author/Commentator
Version of document when opened	Date of document as specified by previous author	Status of document when opened	Name of previous author(s) or commentator(s)
1.0.0	2014-09-03	Final	D.R. van Niekerk

CONTENTS

1.	Background.....	1
2.	Afrikaans dependency Treebank.....	1
2.1.	Development	1
2.2.	Results	2
3.	Afrikaans dependency parser.....	3
3.1.	Development	3
3.2.	Results	4
	References	5
	Appendix A: Annotation Protocol.....	6

1. Background

In this project, KU Leuven (KUL) and the North-West University (NWU) collaborated to develop an easy to use web-based linguistic search engine for Afrikaans for use by researchers and students in the Humanities and Social Sciences. The project was factored into three *work packages*, of which the North-West University was responsible for implementation and delivery of the first two:

1. A **dependency parser** for Afrikaans based on a Dutch parser and an Afrikaans-Dutch convertor (A2DC/D2AC);
2. A manually corrected parsed corpus (dependency treebank) for Afrikaans based on the freely available NCHLT corpus; and
3. GrETEL for Afrikaans.

We report on these work packages below, describing the development, testing and resulting deliverables, starting with work package 2: the **treebank** and concluding with work package 1: the **parser**. The reason for the ordering in this report is the fact that the development of the latter eventually depended on the former.

2. Afrikaans dependency Treebank

As proposed, the Afrikaans NCHLT¹ Annotated Text Corpora [1] of government domain text was used as a basis for the development of this resource. This corpus was selected because it had already been annotated with part-of-speech (POS) tags and word lemmas in a previous project. The goal for the current work package could thus be defined as annotating the corpus for word dependencies within sentence context given the pre-existing POS tags and lemmas. This is described below (and in more detail in the Annotation Protocol attached to this report as Appendix A).

In the following subsections we focus reporting on the development process and results obtained for the proposed technical specifications tests on the resulting corpus respectively.

2.1. Development

To bootstrap the annotation of the corpus a preliminary parser was constructed using the rule-based A2DC² Afrikaans to Dutch converter [2] and the Frog³ Dutch dependency parser [3]. This resulted in a first-pass parse. Word dependency relations and tags were combined with the pre-existing POS tags, mapped to the tagsets adopted here (described below) and converted to the annotation tool file format for manual correction by the annotators. The annotation tool used during manual correction was the BRAT⁴ web-based rapid annotation tool [4] of which the standard dependency annotation configuration was customised for this task.

Given the nature of the text corpus, a number of sentence fragments are present that may not be annotated sensibly for dependency structure. The first action thus performed was filtering of the corpus so that only relatively complete sentences are annotated in this work. This

¹ South African National Centre for Human Language Technologies

² <http://sourceforge.net/projects/d2ac-a2dc/>

³ <http://ilk.uvt.nl/frog/>

⁴ <http://brat.nlplab.org/>

involved a simple check for the presence of a verb and sensible end-of-sentence punctuation. *Table 1* summarises the corpus properties of the original text and after applying this initial filtering. The latter (bolded fields) is the source for the annotated corpus developed here.

Table 1: Corpus partitions and properties

Corpus/subsections	Number of words	Number of sentence
Original train	55386	2610
Original test	5834	329
Filtered train	43895	1663
Filtered test	5381	271

The POS annotation in the NCHLT corpus was based on a fine-grained *tagset* developed specifically for Afrikaans. As some of this information is superfluous for the task of determining the sentence dependency structure and the additional granularity increases the difficulty (and thus the reliability) of the manual annotation task, the POS tagset was simplified to only distinguish a largely universal set of POS tags [5] (more information can be found in Appendix A). For the dependency relation annotation, a subset of the Stanford tagset [6, 7] was adopted and the conventions described in these papers applied. One principal in particular that might differ from other conventions such as those followed by Frog is that content words should be related to each other directly where possible instead of, for example, via conjunctions such as *and*. The full set of allowed dependency tags is documented in Appendix A, however, in practice annotators were instructed to prioritise finding the correct dependency structure of sentences and encouraged to use more generic tags if unsure of the exact linguistic relation. This was done to ensure a high level of consistency during annotation from which future work can focus on systematically improving the finer accuracy involving the use of specific tags (see the subsection below where the inter-annotator agreement (IAA) is quantified). Lastly, all sentences were validated for adherence to the protocol defined, including projectivity of dependency relations.

Informal evaluations showed that only about 20% to 30% of connections from the preliminary (A2DC/Frog) parser were kept during manual annotation. Thus, only around 10% of the corpus was annotated from the output of the preliminary parser before training a new neural network-based parser [8]. Two more iterations updating the parser were performed during annotation to incrementally improve the starting point for manual correction (and therefore reduce annotation time). The parser is discussed in the following section.

2.2. Results

The corpus was annotated by one primary annotator and a subset (containing 943 words) was annotated by a second annotator experienced in computational linguistics (and especially Afrikaans linguistics). For this subset we calculated the IAA in terms of both the *labelled attachment score* (LAS) and *unlabelled attachment score* (UAS) averaged over words [9]. These results are presented in *Table 2*.

Table 2: Inter-annotator agreement

UAS	0.889
LAS	0.825

We interpreted this result as indicating that the task defined in the annotation protocol (Appendix A) was sufficient to result in consistent annotation. The LAS of 82.5% is higher than what was anticipated during the proposal (65% was required) and we thus consider the

annotation effort a success. Further analysis of annotator differences primarily pointed out that the primary annotator often used more generic tags where more specific ones were appropriate. In the final corpus of the full set of 31 tags defined in the annotation protocol, only 20 were used (and only 18 in significant numbers). *Table 3* shows the frequency of occurrence of each tag, it is clear that the generic tag “mod” is frequently used, and could be partitioned into more specific tags such as “prep” and “neg”. Specific cases have been noted in the README file accompanying the delivered corpus and should be amenable to systematic or even automatic or rule-based adjustment as required in future work.

Table 3: Occurrence of tags in the corpus

Tag	Frequency	Tag	Frequency	Tag	Frequency	Tag	Frequency
comp	3	poss	830	cc	1886	obj	3763
mark	5	dep	1009	conj	2359	punct	4497
abbrev	63	arg	1130	amod	2447	det	5101
dobj	111	prt	1375	aux	2534	pobj	6106
num	462	root	1870	subj	2605	mod	11120

The final corpus is released in two parts (as for previous instances of the text corpus divided into: *train* and *test* sets) with accompanying lemmas and POS tags in XML⁵ format. The reason for integrating POS tags and lemmas in this release are two-fold: firstly, a small number of POS tags were corrected during annotation and secondly, both these features are used to develop the parser described below and are thus tied to that deliverable. Other technical details are documented in the README file accompanying the corpus.

3. Afrikaans dependency parser

In the following subsections we briefly motivate and describe the development of the parser and present results on the annotated corpus developed above.

3.1. Development

Our initial proposal envisaged the development of a parser based on A2DC and Frog. Such a system would in principle be able to re-use POS taggers, lemmatisers and the dependency parsing component itself developed for Dutch. While this approach indeed added value in initially bootstrapping the annotation process described above, the accuracy obtainable from this approach was limited in practice. One clear reason for this is simply the cumulative effect of the propagation of errors resulting from a sequence of imperfect components starting with the word-for-word Afrikaans-to-Dutch translation and subsequently POS tagging, lemmatisation and finally dependency parsing. Another factor influencing the results obtained is some of the differences in convention and mapping between the Stanford protocol and the conventions followed by Frog. The LAS and UAS obtained by this initial parser on the manually annotated test set are presented in the following subsection along with the results for the final parser.

The implementation of the parser consists of a number of components: A word tokeniser, POS tagger, lemmatiser and the dependency parsing component. The tokeniser has a simple implementation that tokenises words on white-space and punctuation, the lemmatiser was developed on a separate corpus, while the POS tagger component is taken from the work done in [1]. With the POS tagger having been developed on the same corpus (naturally

⁵ Specifically FoLiA XML as used by the Frog parser and documented here: <http://ilk.uvt.nl/fofia>

trained only on the *train* set), results should be considered representing a “best-case” scenario for the composite parser. For the dependency parsing component we adopted the approach described in [8], and specifically the implementation (called IDP) freely available⁶ under the GNU General Public License (GPL) version 3. For the final version of the parser and all versions used to generate results for this report, we used the default parameters suggested with the software, and with the frequency cut-off free parameters both set to 5.

3.2. Results

Two sets of results are of interest: firstly, for the dependency parsing *component* (i.e. given known lemmas and POS tags) which is analogous to the inter-annotator agreement presented above, and secondly, for the *composite* parser where unannotated input is parsed by a process of tokenisation, POS tagging, lemmatisation and dependency parsing. In *Table 4* results for these cases are presented, training (where applicable) was done on the previously described *train* set (we used 10%, randomly selected, as *development* set during training) and test results on the *test* set. The first line shows the score for the parser component given known POS labels and lemmas and may be compared to the IAA in *Table 2*. The bold lines represent results for parsing from unannotated input sentences (real-world case) and may be compared directly

Table 4: Results for the parsers developed.

	UAS	LAS
IDP parser <i>component</i>	0.922	0.901
Composite parser	0.888	0.814
A2DC+Frog parser	0.432	0.218

As expected, the parser component scores better with known tags and lemmas, with only a slight drop in accuracy for the composite parser, owing to the high accuracy achieved by the POS tagger (the POS tagger was trained on the same corpus).

The composite parser implementation is released as deliverable for this project, trained on the entire corpus (i.e. *train* and *test* combined), with 10% randomly selected as development set, however individual components need to be downloaded separately. These will all be available from the South African Language Resource Agency (RMA)⁷.

⁶ Version 0.0.4 available at: <http://cui.unige.ch/~titov/idp/>

⁷ <http://rma.nwu.ac.za/>

References

- [1] R. Eiselen and M. J. Puttkammer, “Developing Text Resources for Ten South African Languages,” in *Proceedings of the 9th edition of the Language Resources and Evaluation Conference (LREC)*, Reykjavik, Iceland, 2014.
- [2] G. B. Van Huyssteen and S. Pilon, “Rule-based conversion of closely-related languages: a Dutch-to-Afrikaans convertor,” in *Proceedings of the 20th Annual Symposium of the Pattern Recognition Association of South Africa*, Stellenbosch, South Africa, 2009, pp. 23–28.
- [3] A. Van den Bosch, B. Busser, S. Canisius, and W. Daelemans, “An efficient memory-based morphosyntactic tagger and parser for Dutch,” in *Computational linguistics in the Netherlands: Selected papers from the Seventeenth CLIN Meeting*, 2007, pp. 99–114.
- [4] P. Stenetorp, S. Pyysalo, G. Topić, T. Ohta, S. Ananiadou, and J. Tsujii, “BRAT: a web-based tool for NLP-assisted text annotation,” in *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, 2012, pp. 102–107.
- [5] S. Petrov, D. Das, and R. McDonald, “A universal part-of-speech tagset,” in *Proceedings of the 8th Conference on Language Resources and Evaluation*, Istanbul, Turkey, 2012.
- [6] M.-C. De Marneffe and C. D. Manning, “The Stanford typed dependencies representation,” in *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, 2008, pp. 1–8.
- [7] M.-C. De Marneffe, B. MacCartney, and C. D. Manning, “Generating typed dependency parses from phrase structure parses,” in *Proceedings of LREC*, 2006, vol. 6, pp. 449–454.
- [8] I. Titov and J. Henderson, “Fast and Robust Multilingual Dependency Parsing with a Generative Latent Variable Model,” in *EMNLP-CoNLL, 2007*, pp. 947–951.
- [9] J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret, “The CoNLL 2007 shared task on dependency parsing,” in *Proceedings of the CoNLL shared task session of EMNLP-CoNLL*, 2007, pp. 915–932.

Appendix A: Annotation Protocol

Original document follows on the next page.

Afribooms: Annotation guide

Author: Daniel van Niekerk (daniel.vanniekerk@nwu.ac.za)

Version: 1.0.0 (July 2014)

Table of Contents

Introduction.....	1
Dependency parsing.....	2
The annotation task.....	3
NCHLT Afrikaans POS tags.....	3
Dependency relations.....	4
Examples.....	5
Annotation environment.....	6
References.....	7

Introduction

This guide details the annotation process to be followed to verify and correct dependency relations in the NCHLT Afrikaans corpus for the Afribooms project. A first-pass parse is obtained using the rule-based Afrikaans to Dutch convertor *A2DC*¹ [1] and the Dutch *Frog*² dependency parser [2].

The annotation process will make use of the *BRAT*³ web-based rapid annotation tool [3], and this document also serves as a quick-start guide for annotaters using this software.

1 <http://sourceforge.net/projects/d2ac-a2dc/>
2 <http://ilk.uvt.nl/frog/>
3 <http://brat.nlplab.org/>

Dependency parsing

“Dependency grammar (DG) is a class of modern syntactic theories that are all based on the *dependency relation* and that can be traced back primarily to the work of Lucien Tesnière. The dependency relation views the (finite) *verb* as the structural center of all clause structure. *All other syntactic units (e.g. words) are either directly or indirectly dependent on the verb*. DGs are distinct from phrase structure grammars (constituency grammars), since DGs *lack phrasal nodes* - although they acknowledge phrases. *Structure is determined by the relation between a word (a head) and its dependents*. Dependency structures are flatter than constituency structures in part because they lack a finite verb phrase constituent, and they are thus well suited for the analysis of languages with free word order, such as Czech and Turkish.” [4]

We summarise this into a more precise set of properties (for a concrete example see Figure 1):

1. Given a sentence, the set of *dependency relations* between words form a *dependency graph* with *words as nodes* (vertices) and *lines/connections for each relation* (edges).
2. There are no phrasal nodes, only a node for each word and a special “*ROOT*” node.
3. Relations are represented using *directed connections* (with *arrows*) always from a *head* word towards its *dependent*. This results in a graph where all arrows *point away* from the *central verb* (the only exception is the connection from the *ROOT* node).
4. A word *may have multiple dependents*, but *not multiple heads*. Thus each word may have *only one arrow head*.
5. Each sentence must result in a *single dependency graph*. That is, all words must be *reachable from the ROOT node*.
6. The dependency graph for a sentence should be *projective*. That means that *connection lines* are *not allowed to cross each other*.
7. Each *dependency relation* can be labelled to establish the *type of dependency*.

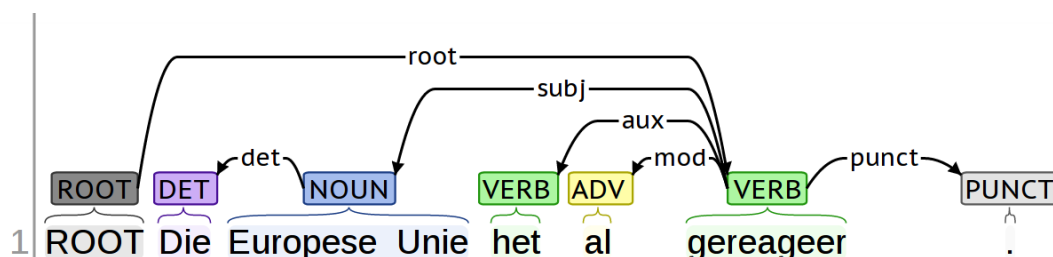


Figure 1: Example of a *dependency graph* for a simple sentence.

The annotation task

Sentences from the NCHLT Afrikaans corpus have been manually annotated for part-of-speech (POS) during a previous project. The goal of the current task is to add dependency relations to this corpus to further improve this linguistic resource and support the development of parsing technology for Afrikaans.

The starting point for manual correction of the dependency annotation is the corpus, with known POS tags and dependencies determined automatically by a Dutch parser (via an Afrikaans to Dutch convertor). Annotators will thus be presented with a representation of each sentence, containing:

1. The original word tokens.
2. Known POS tags for each word.
3. A machine-generated dependency graph with dependency labels.

The task is to verify and edit the dependency relations: **connections and labels** assuming the POS tags are correct. If minor errors in the POS tags are detected they may be corrected, however this is not the main task.

The following subsections will briefly document the POS tags and dependency relations.

NCHLT Afrikaans POS tags

For this task, a simplified set of POS tags will be used [5]:

POS tag	Description	Beskrywing
NOUN	Nouns, including proper nouns and others.	Selfstandige naamwoorde, insluitend eiename, versamelname, ens.
PRON	Pronouns.	Voornaamwoorde.
VERB	Verbs, including auxiliary verbs.	Werkwoorde, insluitend hulpwerkwoorde.
ADJ	Adjectives.	Byvoeglike naamwoorde.
ADP	Prepositions.	Voorsetsels.
ADV	Adverbs.	Bywoorde.
CONJ	Conjunctions.	Voegwoorde.
DET	Determiners, but used here <i>only for definite and indefinite articles</i> (e.g. ' <i>n</i> and <i>die</i>).	Bepaalde en onbepaalde lidwoorde (bv. ' <i>n</i> en <i>die</i>).
NUM	Numerals.	Telwoorde.
PRT	Particles.	Partikels.
PUNCT	Punctuation.	Leestekens.
X	A catch-all class for other categories such as abbreviations, interjections and foreign words.	'n Klas vir oorblywende woorde, insluitend unieke afkortings, tussenwerpsels en leenwoorde.

These tags will aid in labelling dependencies, and in certain cases, to apply *strict constraints* between the POS tag and dependency tag and allowable relations. See the *notes on usage* in the following section.

Dependency relations

Unlike POS tags, dependency relation tags are hierarchical. This allows one to specify dependencies between words even when the nature of the dependency is not accurately captured in one of the more specific categories. For example, the most generic relation is *dependent* and this may be specialised to *aux*, *arg* or *mod*, and these are further specialised as shown in the table below.

The dependency types used for this annotation task are a simplified version of the set in [6] and are obtained by mapping the output of [2], the hierarchy is indicated here, however during annotation only the last (bolded) field is displayed to simplify the tree representation:

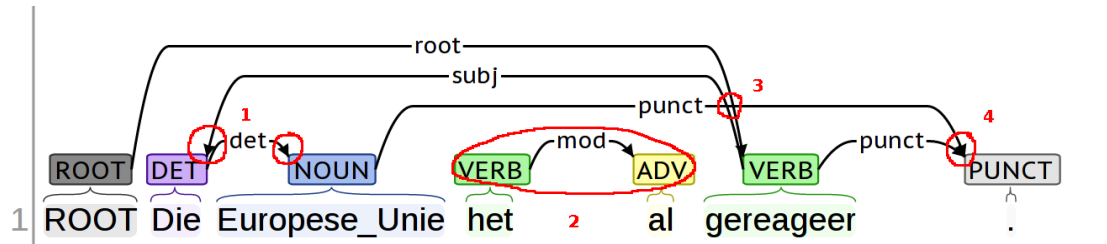
Dep. tag	Description	Dep. tag	Description
dep	dependent	dep: mod	modifier
dep: punct	punctuation	dep:mod: advcl	adverbial clause modifier
dep: root	root	dep:mod: tmod	temporal modifier
dep: aux	auxiliary (verb)	dep:mod: amod	adjectival modifier
dep: conj	conjunct	dep:mod: num	numeric modifier
dep: cc	coordination (e.g. to conjunctions)	dep:mod: number	element of compound number
dep: arg	argument	dep:mod: appos	appositional modifier
dep:arg: subj	subject	dep:mod: abbrev	abbreviation modifier
dep:arg: comp	complement	dep:mod: adv	adverbial modifier
dep:arg:comp: obj	object	dep:mod:adv: neg	negation modifier
dep:arg:comp:obj: dobj	direct object	dep:mod: poss	possession modifier
dep:arg:comp:obj: iobj	indirect object	dep:mod: prt	phrasal verb particle
dep:arg:comp:obj: pobj	object of preposition	dep:mod: det	determiner
dep:arg:comp: compl	complementiser	dep:mod: prep	prepositional modifier
dep:arg:comp: mark	marker (introducing adverbial clause)		
dep:arg:comp: rel	relative (introducing relative clause)		
dep:arg:comp: acomp	adjectival complement		

Notes on usage [7]:

1. *Underspecified relations* (e.g. *dep:arg* and *dep:mod*) should be used to deal with the complexities of real text where necessary.
2. Where possible, relations should be between *content words*, not indirectly mediated via function words.
3. *dep:root* is connected to the main *verb* in a sentence or clause.
4. *dep:punct* should only be *connected to conjuncts* (e.g. for a comma as one would connect a conjunction *dep:cc*) *or to the main verb* (e.g. for end of sentence or clause punctuation).

Examples

Below is an (exaggerated) example illustrating annotation that violates some of the principles and conventions set out in this document:



1. Here the **subject relation** is not directed at the content word (NOUN) and consequently the direction of the **determiner relation** is incorrect.
2. The words here are **not connected to the rest of the graph**, the verb is not dependent on any other word tokens.
3. Connection **lines should not cross each other**, and connecting the punctuation to this noun is not according to convention.
4. A word token may not be dependent on more than one other token. A final check should always be to ensure that each word token has exactly one arrow head (this procedure would also have identified problem 2).

Compare Figure 1 for a valid dependency graph for this sentence. A few more examples of valid dependency graphs may be viewed directly in the BRAT setup under the *afr_dep_example* collection⁴. See the following section regarding the annotation environment.

⁴ http://ctext-data1.puk.ac.za:443/~ntldvn/brat_ref/#/afr_dep_example/dvn

Annotation environment

Annotation will be done using the BRAT online annotation tool in the webbrowser.

The following will be required:

- The Chrome webbrowser.
- An Internet connection.

Each annotator will be given a different *URL*, *username* and *password* with a subset of the corpus to annotate (annotators will not work on the same data simultaneously). The following needs to be done to start annotating:

1. Load the provided *URL* in the Chrome browser.
2. Press TAB or select “Collection” from the drop-down menu (on the left).
3. Select the “NCHLT” folder.
4. Select the document you wish to annotate.
5. Log in using the *username* and *password*.

A document with current annotation (as obtained from the automated parsing procedure described earlier) should now be loaded and editable. Annotators may now start editing by *creating*, *deleting* or *editing* dependency relations (POS tags may similarly be edited but this should rarely be necessary and this is not the main task at hand).

- **To create a relation:** click and drag the mouse from one POS tag to another, select the dependency tag and click “OK”.
- **To delete a relation:** double click on a relation, click on “DELETE” and then “OK”.
- **To edit a relation:** double click on a relation, select a new dependency tag and click “OK”.

Once comfortable with editing, the process may be sped up by *disabling the need to confirm each action*, to do this access the “Options” menu from the drop-down menu (on the right) and switch the “Annotation mode” to “Normal” from “Careful”.

BRAT is pre-configured to *only allow relevant dependency tags given the POS* (one will thus not see the full list of tags for each relation currently being edited).

BRAT automatically saves annotations as they are edited, so it is *important to be online during annotation*. When refreshed, the “Collection” dialogue shows how long ago the last edit was made.

These instructions should suffice for the annotation task at hand, however, more information on the BRAT interface may be found here: <http://brat.nlplab.org/manual.html>

References

- [1] G. B. Van Huyssteen and S. Pilon, “Rule-based conversion of closely-related languages: a Dutch-to-Afrikaans convertor,” in *Proceedings of the 20th Annual Symposium of the Pattern Recognition Association of South Africa*, Stellenbosch, South Africa, 2009, pp. 23–28.
- [2] A. Van den Bosch, B. Busser, S. Canisius, and W. Daelemans, “An efficient memory-based morphosyntactic tagger and parser for Dutch,” in *Computational linguistics in the Netherlands: Selected papers from the Seventeenth CLIN Meeting*, 2007, pp. 99–114.
- [3] P. Stenetorp, S. Pyysalo, G. Topić, T. Ohta, S. Ananiadou, and J. Tsujii, “BRAT: a web-based tool for NLP-assisted text annotation,” in *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, 2012, pp. 102–107.
- [4] “Dependency grammar,” *Wikipedia, the free encyclopedia*. 29-Jun-2014.
- [5] S. Petrov, D. Das, and R. McDonald, “A universal part-of-speech tagset,” in *Proceedings of the 8th Conference on Language Resources and Evaluation*, Istanbul, Turkey, 2012.
- [6] M.-C. De Marneffe, B. MacCartney, and C. D. Manning, “Generating typed dependency parses from phrase structure parses,” in *Proceedings of LREC*, 2006, vol. 6, pp. 449–454.
- [7] M.-C. De Marneffe and C. D. Manning, “The Stanford typed dependencies representation,” in *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, 2008, pp. 1–8.