Journal of King Saud University – Computer and Information Sciences (2014) 26, 441–449

King Saud University

Journal of King Saud University – Computer and Information Sciences

www.ksu.edu.sa
www.sciencedirect.com

An Arabic CCG approach for determining constituent types from Arabic Treebank

Ahmed I. El-taher [a,*], Hitahm M. Abo Bakr [a], Ibrahim Zidan [a], Khaled Shaalan [b]

[a] Derpartment of Computer and System Engineering, Faculty of Engineering, Zagazig University, Zagazig, Asharkia, Egypt
[b] The British University, Dubai, United Arab Emirates

Available online 28 September 2014

KEYWORDS

Arabic;
CCGbank;
Treebank

Abstract Converting a treebank into a CCGbank opens the respective language to the sophisticated tools developed for Combinatory Categorial Grammar (CCG) and enriches cross-linguistic development. The conversion is primarily a three-step process: determining constituents' types, binarization, and category conversion. Usually, this process involves a preprocessing step to the Treebank of choice for correcting brackets and normalizing tags for any changes that were introduced during the manual annotation, as well as extracting morpho-syntactic information that is necessary for determining constituents' types. In this article, we describe the required preprocessing step on the Arabic Treebank, as well as how to determine Arabic constituents' types. We conducted an experiment on parts 1 and 2 of the Penn Arabic Treebank (PATB) aimed at converting the PATB into an Arabic CCGbank. The performance of our algorithm when applied to ATB1v2.0 & ATB2v2.0 was 99% identification of head nodes and 100% coverage over the Treebank data.

1. Introduction

Recently, there has been an immense increase in natural language processing of Arabic, especially since the terrorist attacks on the US on 11 September 2001 (known as 9/11).

Studies on Arabic NLP have been challenged by peculiarities of the language's properties, including highly ambiguous Arabic words, highly complex morpho-syntactic characteristics, the absence of rigorous standards of written text, and the current state-of-the-art in Arabic NLP resources and tools (Shaalan, 2014). Moreover, large collections of tagged documents, corpora and Treebanks, are excellent sources that are needed when developing and testing the performance of an Arabic NLP tool or system. For these linguistic resources to be useful, they should include unbiased distribution and representative numbers of linguistic expressions that do not suffer from sparseness. Unfortunately, the available Arabic linguistic resources for conducting reliable Arabic NLP research often are expensive to create or license. The reason for this is that they require significant human annotation and verification. Few of these corpora have been made freely and publicly available for research purposes, whereas others, such as Treebanks, are available but under license agreements.

A Treebank is a linguistic resource that is composed of large collections of manually annotated and verified syntactic

* Corresponding author at: Houd-Nagih, Hehia, Asharkia governerate, Egypt. Tel.: +20 1115739309.
E-mail address: aieltaher@yahoo.com (A.I. El-taher).
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.jksuci.2014.06.005

analyses of sentences that are carefully and accurately annotated. These annotations are highly useful for the development of a variety of applications, such as tokenization, diacritization, part-of-speech (POS) tagging, morphological disambiguation, base phrase chunking, named entity recognition, and semantic role labeling (Othman et al., 2004).

A highly expressive formalism such as Combinatory Categorial Grammar (CCG) can capture many grammatical phenomena, such as long-range dependencies, where simpler formalisms cannot, as demonstrated by Hassan (2009) and Steedman (1996, 2000). Furthermore, a wide variety of high-quality NLP tools exist for CCG, and an Arabic CCGbank would make this technology available to Arabic for the first time. The Arabic CCGbank will be a transformation of the Penn Arabic Treebank into a corpus of CCG derivations. Hence, a CCGbank of Arabic would be a very beneficial linguistic resource that lends itself to the inherent characteristics of Arabic.

In regard to Arabic, there are two important treebanking efforts: the Penn Arabic Treebank (PATB) (Maamouri et al., 2004) and the Prague Arabic Dependency Treebank (PADT) (Smrž et al., 2008). Both of these efforts employ complex and very rich linguistic representations that require significant human training. The amount of details specified in the representations is impressive. The PATB not only provides tokenization, complex POS tags, and syntactic structure but also provides empty categories, diacritizations, lemma choices and various semantic tags. This information allows for important research in Arabic NLP applications. Consequently, we decided to use PATB[1] in our approach for building an Arabic CCGbank.

In this article, we describe our attempt at creating the Arabic CCGbank using the rules devised by Hockenmaier and Steedman (2005, 2007) for creating the English CCGbank. This research is focused on determining Arabic constituents' types and the necessary preprocessing step. This step is found useful for handling the Arabic Treebank of choice to correct brackets and normalize tags for any changes that were introduced during the manual annotation, during the creation of PATB, as well as for extracting useful information for determining constituents' types.

Section 2 describes related work done to create CCGbanks for languages other than English. Section 3 gives a brief overview of CCG. Section 4 presents the preprocessing of the Arabic Treebank. Section 5 introduces the process of determining Arabic constituents' types. Section 6 describes the experiments applied on PATB and discusses the obtained results. Section 7 provides the conclusions and directions for future work.

2. Related work

After the successful development of the English CCGbank[2], various efforts were made to convert treebanks of other languages into CCGbanks. One example is the conversion of the German Tiger[3] corpus into a German CCGbank (Hockenmaier, 2006). Another example is the conversion of the Turkish dependency Treebank into a Turkish CCGbank (Çakıcı, 2005).

Recently, Tse and Curran (2010) devised a Chinese CCGbank from the Penn Chinese Treebank.

Bos et al. (2009) derived an Italian CCGbank[4] from the Turin University Treebank, while Sandillon-Rezer and Moot (2011) devised a French CCGbank from the Paris VII annotated treebank.

As far as Arabic is concerned, it is worth noting that the only attempt was made by Boxwell and Brew (2010) in their pilot project aiming at converting the PATB into an Arabic CCGbank. The final project achievements included determining 97.99% of the head nodes and 95.06% of the arguments and complement nodes, with 100% coverage on 52.7% of the trees in the Treebank. We will show that the performance of our Arabic CCG algorithm outperforms the performance of Boxwel and Brew's Arabic CCG algorithm.

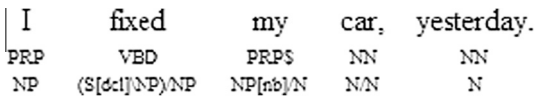


Figure 1 POS tags versus Super tags.

3. Combinatory Categorial Grammar (CCG)

Combinatory Categorial Grammar[5] (CCG) is a lexicalized grammar that directly captures the non-local dependencies involved in treebank construction, including control and raising; see Steedman (1996, 2000), Hockenmaier and Steedman (2005,2007), and Hassan (2009). A category encodes not only information about syntactic, phonological and semantic aspects of a given word but also information about categories with which it can be combined and the result of the combination. CCG has a transparent interface between the surface syntax and the underlying semantic representation.

Categories, sometimes referred to as *types*, have two forms: Primitive (atomic) and complex. Primitive types include constituents, such as N, NP, PP, and S, and they can further be distinguished by features. A complex type denotes a function type, which is a combination of primitive categories, more specifically a function from one category (primitive or function) to another, e.g., S/NP and (S/NP)/(S/NP). Functions specify the type and directionality of their arguments and the type of their results. A forward slash denotes that the argument should appear to the right, while a backslash denotes that the argument should appear on the left. For example, S\NP is an intransitive verb, e.g., "*run*", because it is looking for an NP (to the left) to form an S. In these notations, the transitive verb, e.g., "*fixed*", is denoted by (S\NP)/NP, whereas the ditransitive verb, e.g., "gave", is denoted by ((S\NP)/NP)/NP. Thus, complex categories are able to encode sub-categorization information (cf. Steedman, 2000).

CCG uses new syntactic types called *Supertags*, which can capture the extended lexical information from the grammar onto the lexicon, unlike the part-of-speech tags (POS tags) used in Treebanks. For comparison, Fig. 1 shows the sentence "*I fixed my car, yesterday*" along with its tagging with Super

[1] http://catalog.ldc.upenn.edu/LDC2003T06.
[2] http://catalog.ldc.upenn.edu/LDC2005T13.
[3] http://www.ims.unistuttgart.de/forschung/ressourcen/korpora/tiger.html.
[4] http://www.di.unito.it/~tutreeb/CCG-TUT/.
[5] http://groups.inf.ed.ac.uk/ccg/.

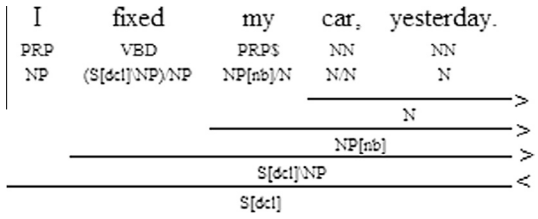


Figure 2 A simple CCG derivation.

tags and POS tags. Super types indicate types of these lexical items, where the transitive verb "fixed" is given the complex category *(S[dcl]\NP)/NP*, which could be described as "the category that, when followed by a noun-phrase *NP*, results in a verb-phrase *(S[dcl]\NP)*, which, when introduced to an *NP* to its left, results in a declarative sentence *S[dcl]*".

A combinatory operator describes the rules applied to combine categories with their arguments to produce the resultant derivations, as follows:

Forward Application: $X/YY \Rightarrow X(>)$
Backward Application $YX \setminus Y \Rightarrow X(<)$

The Forward Application (FA) operator performs the forward combination, where, if a constituent with category X/Y is immediately followed by a constituent with category Y, the operator can be used to combine them to construct a new constituent with category X.

Super tags, when combined with CCG's combinatory operators, compose CCG derivations (proof). A CCG derivation for the sentence in Fig. 1 is shown in Fig. 2. Note the direct correspondence to an upside-down constituency tree. As a simple example, the forward application "$>$" rule combines the CCG-tags of the words "car" and "yesterday", which are "N/N" & "N", respectively, resulting in the CCG-tag "N". For the backward application rule "$<$", it combines the phrase "S[dcl]\NP" with the word "I", resulting in the CCG-tag "S[dcl]".

CCG has several powerful properties that PATB does not support, including:

- CCG imposes that lexical heads of each constituent are identified.
- Complements & adjuncts are clearly distinguished; PATB uses functional tags (e.g., SBJ, CLR, ...) to support this property, but many nodes are unmarked.
- CCG requires binary branching, while PATB uses a multi-branching structure where any non-terminal node could have any number of children at the same level.

4. Arabic treebank preprocessing

The preprocessing step stated by Hockenmaier and Steedman (2005) was essentially to correct tagging and bracketing errors found in the Penn English treebank, which are largely avoided in the Arabic treebank because PATB is much newer (PATB was first released in 2003, while English Treebank-3 was released in 1999).

As a member of the Semitic languages, Arabic is based on a root-and-template morphology with abundant bound morphemes. These morphemes include possessives, pronouns, and discourse connectives (Zitouni, 2014). PATB introduced problems related to Arabic word segmentation. Segmenting bound morphemes reduces lexical sparsity and simplifies syntactic analysis. Word segmentation is a necessary step for Natural Language Processing (NLP) of morphologically rich languages, such as Arabic. It helps improve the quality of Arabic NLP applications, such as machine translation where some English words correspond to only a morpheme (substring) in Arabic words (Abdel Monem et al., 2008). For example, consider the English sentence "The child is playing with the car" which consists of seven words. Its Arabic translation "الطفل يلعب بالسيارة" consists of three words that correspond to the following segments: "The-child", "is-playing" and "with-the-car". Words of the English sentence correlate with the Arabic morphemes (segments). PATB annotators add morphological analysis tags in addition to the part-of-speech (POS) tag that incorporates new tags that are not dealt with in the standard Penn Treebank tagging. These tags need to be normalized to conform to the CCG analysis.

In the following subsections, we describe our efforts to preprocess PATB using techniques derived from Kulick et al. (2006), namely Improved Handling of Punctuation, and those specified by Maamouri et al. (2008) such that the conversion process could be commenced.

4.1. Tree analysis

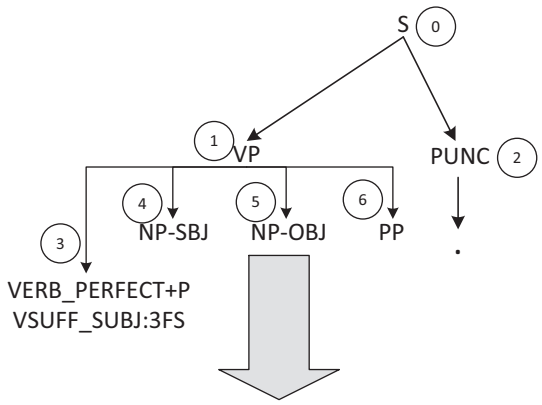
Traditionally, Penn Treebanks' files are presented as text files with a tree-per-line format with parentheses delimiting tree segments. However, PATB does not strictly follow the tree-per-line rule, which could lead to spurious analysis if the line was taken as a whole. Therefore, we devised a preprocessing step that analyzes each line based on the brackets' balance and then returns subtrees. In this step, we handle the case where a parenthesis is missing instead of discarding the entire tree. This step has a significant impact on the quality of the analysis process. For example, Part 1 of PATB consists of 4519 lines (presumably with 4519 trees). However, when we check and correct the missing brackets, it results in 5845 individual trees.

Once individual trees are identified, an analysis of each tree is performed to extract from each node the following information:

- Node's features: tag, word (for the terminal node), trace (e.g., *T*), co-reference (e.g., −1) and gap-reference (e.g., =1).
- Node's relation: its parent node and, for non-terminal nodes, its children nodes.

During this step, we identified nodes' tags by applying the heuristics presented by Sang and Buchholz (2000)[6] to Part 1 of PATB, but we found cases where some nodes are untagged, i.e., they are given the temporary tag "NOTAG". For example, consider the Noun Phrase "حزيران (يونيو)" (June), which appears in PATB as follows: "(NP (NOUN_PROP HuzayorAn) (_) (NOUN_PROP yuwniyuw))", where the node "(_)" is changed to "(NOTAG _)".

[6] http://ilk.uvt.nl/team/sabine/.



Tags	
0	S
1	VP
2	PUNC
3	VERB...
4	NP-SBJ
5	NP-OBJ
6	PP

Parents	
0	undefined
1	S
2	S
3	VP
4	VP
5	VP
6	VP

Figure 3 Partial tree analysis and its implementation.

We implemented our software in Perl, and we used *hash objects* to store different nodes' features. We used hashes because they are efficient in storing and retrieving data in the form of "key-value" pairs, where each key is unique in the hash and its corresponding value can be accessed directly when its key is known (i.e., without searching).

During the analysis step, our implementation gives each node an identifying number (starting from zero at the root node) as a key. Fig. 3 shows the results of analyzing the upper three levels of the sentence[7]: "خطت الولايات المتحدة وبريطانيا خطوة جديدة فى حربهما....,"xaT + atAl + wilAy + At + uAl + mut~aHid + ap + uwa--briyTAniy AxaTow + ap + Fjadiyd + ap + FfiyHarob + i- -himA...." (The United States and Britain took a new step in their war ...).

This figure shows the analysis tree and key assignment for each node. In our implementation, this tree is represented by two tables: Tags and Parents. The entry of the Tags table is the assigned key along with its node. The entry of the Parents table is the key along with its parent node. For example, the key "1" in the Tags table points to "VP", which has the parent "S" in the Parents table. The full tree analysis is shown in Fig. 4.

4.2. Tag conversion and correction

Arabic annotators have augmented morphological analysis information into PATB's tags, but these tags were different from the tag-set commonly used in annotating the Penn Treebank. As a consequence, PATB tags should be normalized. However, it is problematic to map hundreds of tags (reaching its maximum of 668 tags in version 3.1 of PATB) to the 48 in the standard tag-set of Penn Treebank. Fortunately, PATB documents include mapping guidelines.

We used the methods described by Ann Bies[8], Bikel (2002, 2004)[9] and Habash et al. (2009a) to construct lookup tables that cover all parts of PATB and that implement unique mappings. There are two tables in particular: one for version 3.0 and earlier versions and another table for version 3.1 and later versions.

The unmapped tags are mainly punctuations indicated by the tag "PUNC" in PATB, which is too general as it covers all punctuations. The standard Penn Treebank tag-set has 12 different tags for punctuation in addition to the *SYM* tag that indicates symbolic tags (" + ", " = ", "&", ..., etc.).

Therefore, we followed the standard punctuation tagging practice and used the genuine word (token) at each node when mapping punctuations.

Table 1 shows the mapping of punctuations to each of the 12 tags. For example, the question mark, "?", maps to the tag ".".

We found cases where the preposition "عن" "Ean" (from) is erroneously tagged as "PUNC", which we corrected to the intended preposition tag "IN".

Nevertheless, the following tags need special handling:

- The *NEG_PART* + *PVSUFF_SUBJ:3MS* tag pattern is mapped to the *VBP* (imperfect verb) tag if its parent is the *VP* tag. Otherwise, it is mapped to the *RP* tag (Adverb).
- The *NO_FUNC* tag is mapped to the *CC* (Coordination Conjunction) tag if the current word is "و" (w) when it functions as a clitic attached to the following word. Otherwise, we check whether the tag functions as a punctuation tag to perform the punctuation mapping; if not, it is mapped to the *NNP* (Proper noun) tag.
- Both *NON_ALPHABETIC* and *NON_ARABIC* tags are checked as to whether the tag functions as a punctuation tag to perform the punctuation mapping. If not, we check whether they are numbers to map to the *CD* (Cardinal Number) tag. Otherwise, they map to the *Foreign Word (FW)* tag.
- The temporary *NOTAG* tag discussed earlier in Section 4.1 is mapped to one of the 12 punctuation tags.

The application of tag normalization on the tree shown in Fig. 4 is illustrated in Fig. 5. For example, the verb "خطت" "xaT + at" (took) is tagged as "VERB_PERFECT + PVSUFF_SUBJ:3FS" (perfect verb with third person singular feminine subject suffix) in PATB. This tag is converted to "VBD" (Perfect verb). This figure also shows the punctuation mapping for ".".

4.3. Segmenting determiners

In PATB, determiners (tagged as DET or DEM) attached to words are not cliticized (takes spate token) from their respective words because they do not affect the structure of the analyzed sentence. However, this is not applicable to CCGbank

[7] The sentence is extracted from the file: "UMAAH_UM.ARB_20020120-a.0006.tree" in Part 2 of PATB.

[8] Bies's mapping: http://www.ldc.upenn.edu/Catalog/docs/LDC2003T06/arabic-POS tags-collapse-to-PennPOS tags.txt.

[9] Bikel's mapping: http://www.ldc.upenn.edu/Catalog/docs/LDC2005T02/taglist-conversion-to-PennPOS.lisp.

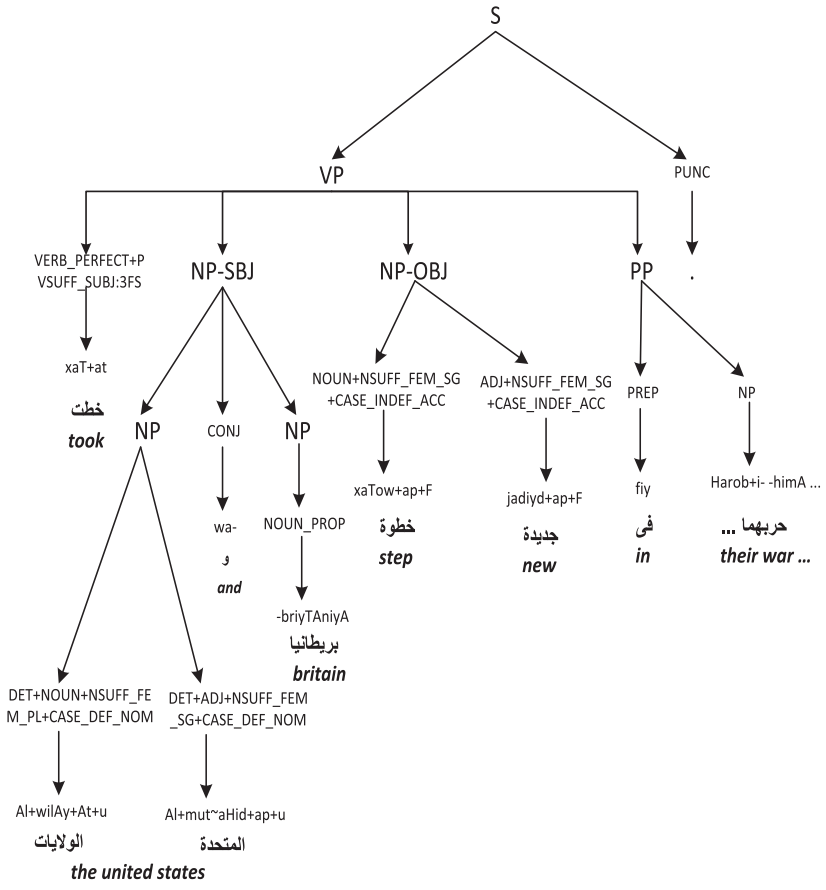


Figure 4 Full tree analysis extracted from PATB.

Table 1 Punctuation mapping to the Penn Treebank tag set.

Tag	Punctuations
SYM	& @ = -PLUS- *
-LRB-	-LRB- -LCB- (Left Paren)
-RRB-	-RRB- -RCB- (Right Paren)
#	# (the pound sign)
$	$ (the dollar sign)
.	.? ! (Sent final punct)
,	(comma)
:	;: _ … (mid sent punc)
″	“ ‘ (left quote)
″	” ’ (right quote)
NN	%
CD	Numbers (e.g., 910, 192, 2)

because determiners are considered functions from Noun Phrases to Nouns. Hence, to capture this relation, we should cliticize determiners. This also has the advantage of reducing data sparseness. For the purpose of accuracy in segmenting determiners from their attached words, we relied on the diacritized Treebank version of PATB, as word syllables are explicitly delimited by the “ + ” symbol. To properly handle words tagged “NNP” or “NNPS”, we decided not to cliticize their attached determiners provided the word under consideration is any of the following:

- It is part of a name; its adjacent words are tagged “NNP” or “NNPS”,
- It is the only child, or
- Its adjacent words are punctuation.

Splitting the determiner from the attached word (the leaf/terminal node in the parse tree) is replaced by a new sub-tree. Its children are the determiner (with the tag DT) and the segment that remains after splitting the determiner (the sibling node with the tag of the original node). The tag of the root of the new sub-tree is determined by the tag of the original word, as shown in Table 2. For instance, if the original word tag is **JJ** (adjective), then the tag of the sub-tree root is **ADJP** (adjective phrase).

Fig. 6 illustrates the effect of separating determiners on the structure shown in Fig. 5.

4.4. Removing vowels

PATB uses “a, i, o & u” vowel letters and the “∼” to denote diacritization of Arabic. Our objective is to handle Modern Standard Arabic (MSA). The orthography of conventional written MSA does not require the inclusion of short vowels; see (Abo Bakr et al., 2008) (Shaalan et al., 2009). Hence, we decided to discard the vowel letters. The only exception made is in the case of non-Arabic (foreign) words, which are annotated with either FW or Latin tags.

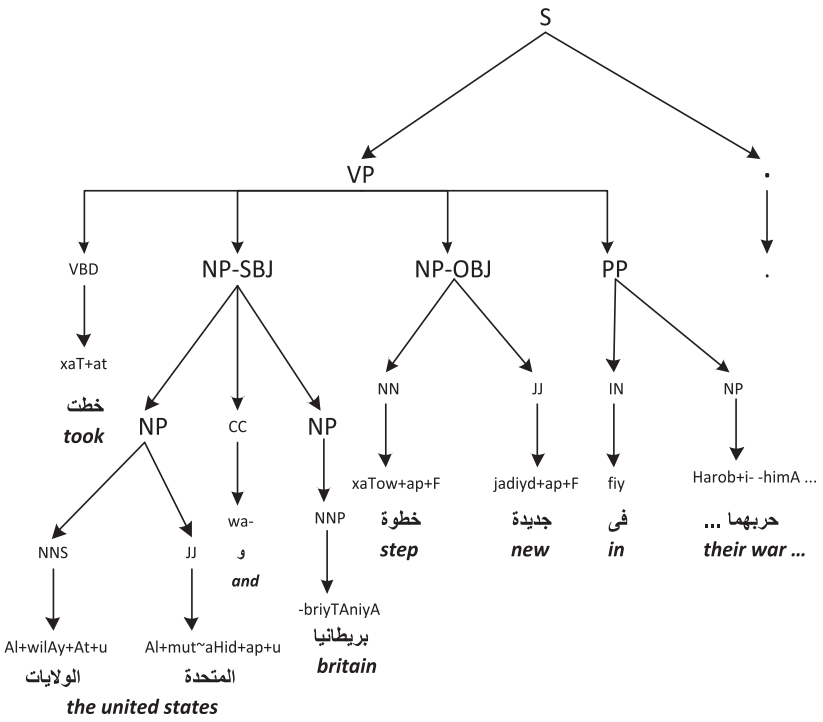


Figure 5 Tag conversion.

Table 2 Assignment of the parent node's tag.

Original word tag	Parent's tag (sub-tree root)
JJ, JJR	ADJP
RB	ADVP
UH	INTJ
NN, NNS, NNP, NNPS	NP
IN	PP
RP	PRT
CD	QP

5. Determining constituents' types

This step identifies the type of each node within a segment of the tree. The constituents' types handled in this stage are head, complement, and adjunct nodes for each nonterminal node.

5.1. Head node identification

For each non-terminal node, the head node is identified using heuristics derived from rules described in Hockenmaier and Steedman (2005). We successfully achieved 99% for identifying head nodes of the non-terminal nodes of PATB. The remaining nodes failed to comply with the devised heuristics, mainly because PATB annotators applied co-indexing on non-terminal nodes in contrast to the convention of applying the co-indexing on terminals.

This co-indexing scheme resulted in constituents having a trace of only one child or an unconventional constituent structure of two or more children, which could not be handled by normal heuristics. Therefore, we dealt with this case by deriving new heuristics where the head node is determined using its location within the constituent; see also Magerman (1994) and Collins (1999).

5.2. Identification of complement and adjunct Nodes

After determining head nodes, the remaining nodes are either complements or adjuncts. Complement nodes, when combined with the head node, comprise a complete CCG analysis, while adjunct nodes do not affect the analysis. One of the following heuristics would determine the types:

- Check explicitly whether the node functions as a complement tag (e.g., SBJ or OBJ) or an adjunct tag (e.g., ADV or LOC),
- Check for exceptions (e.g., NP-TPC is a complement if it was co-indexed), and determine type,
- For any other nodes, use the heuristics of their constituent phrase to determine their complements, or
- Consider the remaining nodes as adjuncts.

The heuristics in Hockenmaier and Steedman (2005) failed to address verbs having two objects because PATB denotes one of the objects with the adverbial tag "BNF", resulting in an adjunct analysis. However, using heuristics from Bies et al. (1995), we successfully gave the object a complement analysis.

Fig. 7 illustrates the results of applying our heuristics on the tree shown in Fig. 6 to determine the constituents' types. In this figure, the verb خطت "xT + t" (took) that is tagged with **VBD** is the head node ***(h)*** of the verb phrase (VP), while the preposition phrase (PP) is an adjunct ***(a)***.

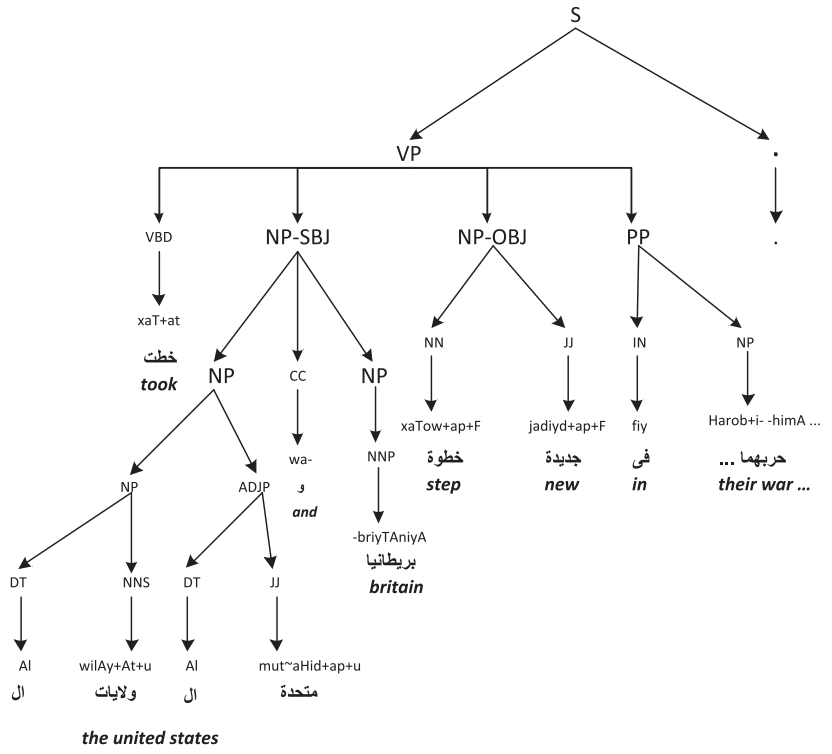


Figure 6 Separating determiners.

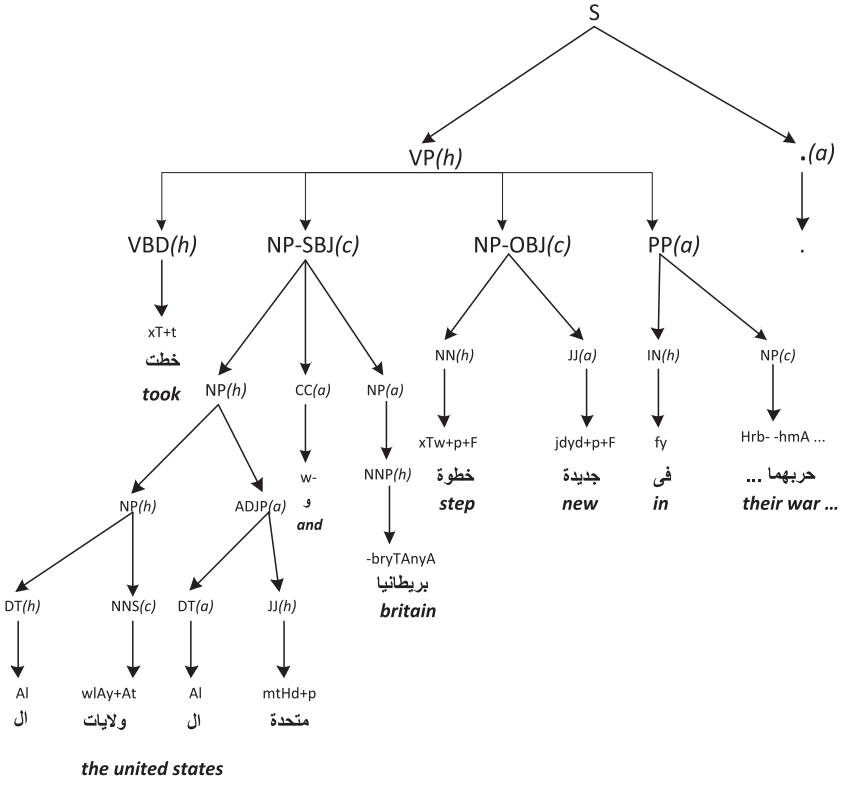


Figure 7 Determining types (*h*: head, *c*: complements and *a*: adjuncts).

Table 3 Results.

	PATB1v2.0	PATB2v2.0
Number of lines	4519	2591
Number of trees	5845	4302
Non-terminal nodes	198,849	214,470
Undefined heads	1668	1930
Heads found (%)	99.16	99.1
Complements & Adjuncts	All complements and adjuncts were determined, provided their head nodes were identified	

6. Experiments and results

We tested our algorithm on Parts 1 & 2 of version 2.0 (denoted ATB1v2.0 & ATB2v2.0), where they had the following characteristics:

- ATB1v2.0 (LDC2003T06). It includes 734 stories from the Agency France Press (AFP) newswire, representing 140,265 words.
- ATB2v2.0 (LDC2004T02). It includes 501 stories from the Ummah Arabic News Text, representing 144,199 words.

Each Treebank's vowel part was merged to facilitate processing. The results are presented in Table 3.

For the unidentified head nodes, we adapted rules from Magerman (1994) and Collins (1999) such that we could capture those heads and, consequently, their complements & adjuncts.

7. Conclusions and future work

This paper reports an ongoing research project that aims to develop a new Arabic CCGbank that would introduce Arabic NLP for the first time to the sophisticated tools developed for CCG.

We decided to use PATB, which has become a de facto standard linguistic resource widely used in Arabic NLP tasks. Characteristics and peculiarities of Arabic usually necessitate a preprocessing step. This is required for normalizing PATB and making it suitable and accurate for the conversion to CCGbank. Furthermore, during the preprocessing step, we successfully enriched the lexicon of PATB through cliticizing determiners, which introduced new words that were not available in the lexicon such that it fully captured the effect of determiners on the lexicon. We developed a complete stage for determining constituents' types, which is considered a building block for producing the Arabic CCGbank; the remaining steps of creating a CCGbank rely heavily on the performance of determining the constituents' types stage to the extent that the binarization step solely depends on this stage.

The performance of our algorithm when applied to ATB1v2.0 and ATB2v2.0 was 99% identification of head nodes and 100% coverage over the treebank data.

We are working on completing the remaining stages of creating an Arabic CCGbank, namely binarization and category conversion. Ultimately, we will make our CCGbank tool freely available for the Arabic NLP research community.

After fully developing the Arabic CCGbank, we will use it to train an English-to-Arabic translation system. We plan to benefit from the Arabic NLP tools devised in Habash et al. (2009b), Diab (2009), Clark and Curran (2007) and Curran et al. (2007) to process the Arabic side of the translation system's training data. Additionally, we plan to use tools devised in Koehn et al. (2007) to train the translation system using techniques from Hassan (2009), Koehn and Hoang (2007) and Birch et al. (2007).

References

Abdel Monem, Azza, Shaalan, Khaled, Rafea, Ahmed, Baraka, Huda, 2008. Generating Arabic text in multilingual speech-to-speech machine translation framework. Mach. Transl. 20 (4), 205–258, Springer, Netherlands.

Abo Bakr, Hitham, Shaalan, Khaled, Ziedan, Ibrahim, 2008. A hybrid approach for converting written Egyptian colloquial dialect into diacritized Arabic. In: Proceedings of INFOS2008, the special track on Natural Language Processing, 27–29 March, Cairo, Egypt.

Bies, Ann, Ferguson, Mark, Katz, Karen, MacIntyre, Robert, 1995. Bracketing Guidelines for Treebank II StylePenn Treebank Project. Technical Report, LDC.

Bikel, Daniel M., 2002. Design of a multi-lingual, parallel-processing statistical parsing engine. In: Proceedings of HLT2002, San Diego, CA.

Bikel, Daniel M., 2004. Intricacies of Collins' parsing model. Comput. Ling. 30 (4), 479–511.

Birch, Alexandra, Osborne, Miles, Koehn, Philipp, 2007. CCG Super tags in Factored Statistical Machine Translation. In: Proceedings of ACL.

Bos, Johan, Bosco, Cristina, Mazzei, Alessandro, 2009. Converting a dependency Treebank to a categorial grammar Treebank for Italian. In: Proceedings of TLT 8, Milano, Italy.

Boxwell, Stephen A., Brew, Chris, 2010. A pilot Arabic CCGbank. In: Proceedings of LREC-10, Valleta, Malta.

Çakıcı, Ruken, 2005. Automatic induction of a CCG grammar for Turkish. In: Proceedings of ACL Student Research Workshop. pp. 73–78.

Clark, Stephen, Curran, James R., 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. Comput. Ling. 33 (4).

Collins, Michael, 1999. Head-Driven Statistical Models for Natural Language Parsing (Ph.D. thesis). Computer and Information Science, University of Pennsylvania.

Curran, James R., Clark, Stephen, Bos, Johan, 2007. Linguistically motivated large-scale NLP with C&C and boxer. In: Proceedings of ACL demo. pp. 33–36.

Diab, Mona T., 2009. Second generation AMIRA tools for Arabic processing: fast and robust tokenization, POS tagging, and base phrase chunking. In: Proceedings of 2nd International Conference on Arabic Language Resources and Tools.

Habash, Nizar, Faraj, Reem, Roth, Ryan, 2009. Syntactic annotation in the Columbia Arabic Treebank. In: Proceedings of MEDAR, Cairo, Egypt.

Habash, Nizar, Rambow, Owen, Roth, Ryan, 2009. MADA + TOKAN: a toolkit for Arabic tokenization, diacritization, morphological disambiguation, POS tagging, stemming and lemmatization. In: Proceedings of MEDAR, Cairo, Egypt.

Hassan, Hany, 2009. Lexical Syntax for Statistical Machine Translation (Ph.D. thesis). Dublin City University.

Hockenmaier, Julia, 2006. Creating a CCGbank and a wide coverage CCG lexicon for German. In: Proceedings of the ACL, vol. 44. p. 505.

Hockenmaier, Julia, Steedman, Mark, 2005. CCGbank: User's Manual. Technical Report MS-CIS-05-09. Department of Computer and Information Science, University of Pennsylvania.

Hockenmaier, Julia., Steedman, Mark., 2007. CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. Comput. Ling. 33 (3), 355–396.

Koehn, Philipp, Hoang, Hieu, 2007. Factored translation models. In: Proceedings of EMNLP, Prague, Czech Republic.

Koehn, Philipp, Hoang, Hieu, Birch, Alexandra, Callison-Burch, Chris, Federico, Marcello, Bertoldi, Nicola, Cowan, Brooke, Shen, Wade, Moran, Christine, Zens, Richard, Dyer, Chris, Bojar, Ondrej, Constantin, Alexandra, Herbst, Evan, 2007. Moses: open source toolkit for statistical machine translation. In: Proceedings of ACL, Demonstration Session, Prague, Czech Republic.

Kulick, Seth, Gabbard, Ryan, Marcus, Mitchell, 2006. Parsing the Arabic Treebank: analysis and improvements. In: Proceedings of TLT 6, Prague, Czech Republic.

Maamouri, Mohamed, Bies, Ann, Buckwalter, Tim, Mekki, Wigdan, 2004a. The Penn Arabic Treebank: building a large-scale annotated Arabic corpus. In: Proceedings of NEMLAR. pp. 102–109.

Maamouri, Mohamed, Bies, Ann, Kulick, Seth, 2008. Enhancing the Arabic treebank: a collaborative effort toward new annotation guidelines. In: Proceedings of LREC'08, Marrakech, Morocco.

Magerman, David M., 1994. Natural Language Parsing as Statistical Pattern Recognition (Ph.D. thesis). Department of Computer Science, Stanford University.

Othman, Eman, Shaalan, Khaled, Rafea, Ahmed, 2004. Towards resolving ambiguity in understanding Arabic sentence. In: Proceedings of the International Conference on Arabic Language Resources and Tools, NEMLAR, 22nd–23rd Sept., 2004, Egypt. pp. 118–122.

Sandillon-Rezer, Noémie-Fleur, Moot, Richard, 2011. Using tree transducers for grammatical inference. LACL 2011. LNAI 6736, 235–250.

Sang, Erik Tjong Kim, Buchholz, Sabine, 2000. Introduction to the CoNLL-2000 shared task: Chunking. In: Proceedings of the CoNLL, pp. 127–132.

Shaalan, Khaled, 2014. A survey of Arabic named entity recognition and classification. Comput. Ling. 40, 2, MIT Press.

Shaalan, Khaled, Abo Bakr, Hitham, Ziedan, Ibrahim, 2009. A hybrid approach for building Arabic diacritizer. In: Proceedings of EACL 2009, Workshop on Computational Approaches to Semitic Languages, Association for Computational Linguistics, Athens, Greece, 31 March, 2009. pp. 27–35.

Smrž, Otakar, Bielický, Viktor, Kouřilová, Iveta, Kráčmar, Jakub, Hajič, Jan, Zemánek, Petr, 2008. Prague Arabic dependency treebank: a word on the million words. In Proceedings of LREC 2008, Marrakech, Morocco. pp. 16–23

Steedman, Mark, 1996. Surface Structure and Interpretation. MIT Press, Cambridge, MA.

Steedman, Mark, 2000. The Syntactic Process. MIT Press, Cambridge, MA.

Tse Daniel, Curran, James R., 2010. Chinese CCGbank: extracting CCG derivations from the Penn Chinese Treebank. In: Proceedings of Coling 2010. pp. 1083–1091.

Zitouni, Imed, 2014. Natural language processing of Semitic languages. In: Theory and Applications of Natural Language Processing. Springer, Heidelberg.