# Converting an HPSG-based Treebank into its Parallel Dependency-based Treebank

**Masood Ghayoomi**[†] [‡] **Jonas Kuhn**[‡]

[†]Department of Mathematics and Computer Science, Freie Universität Berlin
[‡] Institute for Natural Language Processing, University of Stuttgart

Masood.Ghayoomi@fu-berlin.de Jonas.Kuhn@ims.uni-stuttgart.de

**Abstract**

A treebank is an important language resource for supervised statistical parsers. The parser induces the grammatical properties of a language from this language resource and uses the model to parse unseen data automatically. Since developing such a resource is very time-consuming and tedious, one can take advantage of already extant resources by adapting them to a particular application. This reduces the amount of human effort required to develop a new language resource. In this paper, we introduce an algorithm to convert an HPSG-based treebank into its parallel dependency-based treebank. With this converter, we can automatically create a new language resource from an existing treebank developed based on a grammar formalism. Our proposed algorithm is able to create both projective and non-projective dependency trees.

**Keywords:** Treebank Conversion, the Persian Language, HPSG-based Treebank, Dependency-based Treebank

## 1. Introduction

Supervised statistical parsers require a set of annotated data, called a treebank, to learn the grammar of the target language and create a wide coverage grammar model to parse unseen data. Developing such a data source is a tedious and time-consuming task. The difficulty of developing this data source is intensified when it is developed from scratch with human intervention. It is possible to develop different treebanks for a language according to a grammar formalism, such as Phrase Structure Grammar (PSG) (Marcus et al., 1993), Head-driven Phrase Structure Grammar (HPSG) (Oepen et al., 2002), Lexical Functional Grammar (LFG) (Cahill et al., 2002), Combinatory Categorial Grammar (CCG) (Hockenamier and Steedman, 2007), Tree Adjoining Grammar (TAG) (Shen and Joshi, 2004), and Dependency Grammar (DepG) (Rambow et al., 2002) for English.

It is also possible to create a new language resource automatically from the existing treebank based on a grammar formalism, i.e. a new treebank can be created through the conversion of data developed based on one grammar formalism into another. This method of developing treebanks is less costly than developing one from scratch in terms of both time and expense.

In this paper, we propose an algorithm that converts an HPSG-based treebank into its parallel dependency-based treebank automatically. This algorithm is applied on the PerTreeBank, the HPSG-based treebank for Persian.

The structure of the paper is as follows: we describe the background of treebanking via conversion in Section 2. The properties of the PerTreeBank are described in Section 3. The algorithm for converting the existing treebank into its parallel dependency-based treebank is introduced in Section 4. The properties of the available dependency treebanks for Persian and their pros and cons are explained in Section 5. The tools and the experimental setup as well as the obtained results are described and discussed in Section 6. The paper is summarized in Section 7.

## 2. Treebanking via Conversion

The common properties among grammar formalisms make it possible for a monolingual treebank based on one grammar formalism to be converted into another formalism, for instance converting a treebank from Lexicalized TAG into HPSG for English (Tateisi et al., 1998; Yoshinaga and Miyao, 2002), PSG into HPSG for English (Miyao et al., 2005), PSG into HPSG for German (Cramer and Zhang, 2010), PSG into HPSG for Chinese (Yu et al., 2010), DepG into HPSG for Russian (Avgustinova and Zhang, 2010), HPSG into DepG for Bulgarian (Chanev et al., 2006), PSG into CCG for English (Hockenmaier, 2003; Hockenamier and Steedman, 2007), DepG into CCG for Italian (Bos et al., 2009) and Turkish (Çakici, 2005), PSG into TAG for English (Chen et al., 2006), PSG into LFG for English (Cahill et al., 2002; Burke, 2006) and Chinese (Guo et al., 2007) and Arabic (Tounsi et al., 2009) and French (Schluter, 2011), and PSG into DepG for French (Candito et al., 2010).

In this paper, we aim at creating a dependency-based treebank for Persian through an automatic conversion of an existing HPSG-based treebank.

## 3. The PerTreebank

The PerTreeBank is an HPSG-based treebank for Persian developed through a bootstrapping approach (Ghayoomi, 2012). The backbone of this treebank is

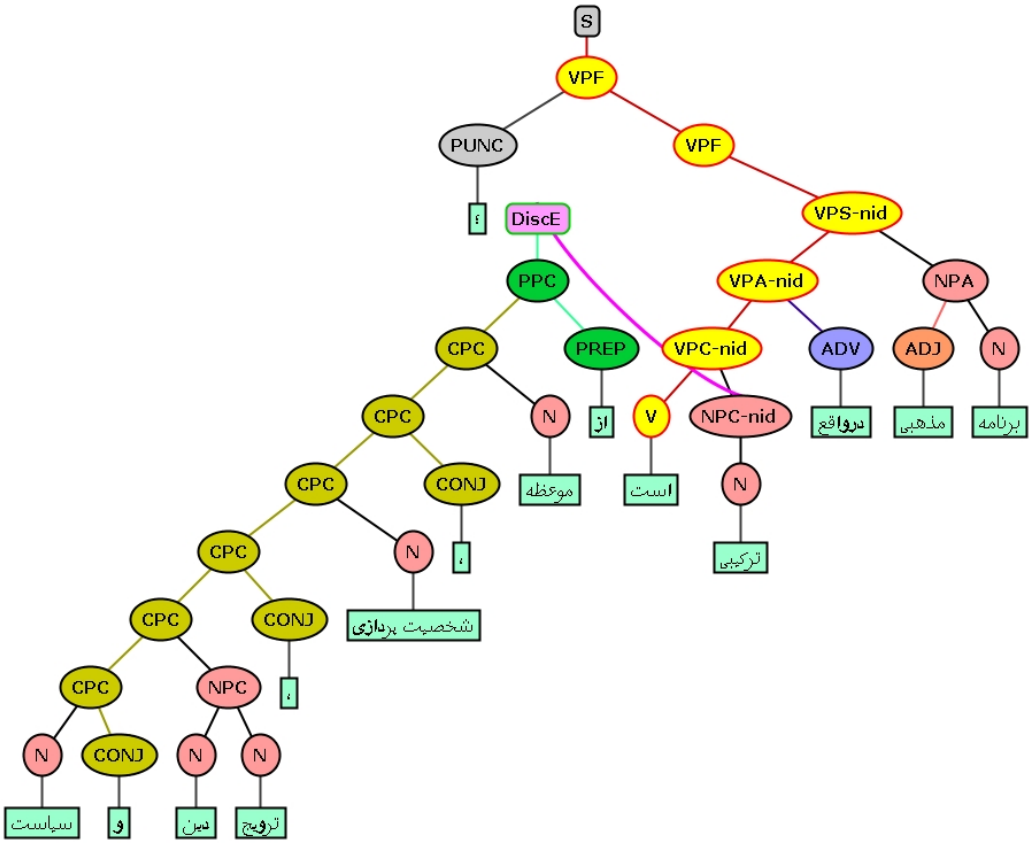


Figure 1: Tree analysis of Example 1 from the PerTreeBank

the HPSG formalism (Pollard and Sag, 1994). The constituent nodes of the phrase structure trees in this treebank contain four basic syntactic relations, which are the simulation of four basic schemas in HPSG to represent the syntactic relations in the constituents, such as the Head-Subject, Head-Complement, Head-Adjunct, and Head-Filler relations. Furthermore, this is a trace-based treebank, i.e. the canonical positions of the extraposed elements are determined in the trees with the empty node *nid* (non-immediate dominance) to represent traces. Later it is converted into a trace-less treebank such that the trace is removed and the functional label *-nid* is used for representing a slashed element in normal HPSG. Additionally, the provided trees are projective.

Figure 1 represents the tree analysis of Example 1 from the PerTreeBank.

(1) برنامه مذهبی درواقع ترکیبی است از موعظه، شخصیت‌پردازی، ترویج دین و سیاست .
barnāme-ye mazhabi darvāqe? program-EZ religious in_fact
tarkib-i ast az mo?eze combination-INDEF is of homily
šaxsiyatpardāzi tarvij-e din va characterization promotion-EZ religion and
siyāsat politics
'A religious program is a combination of homily, characterization, promotion of religion, and politics.'

As shown in the figure, the prepositional phrase which is a complement of the noun 'ترکیب' /tarkib/ 'combination' is extraposed to the post-verbal position. The Head-Filler relation is used to bind the slashed element.

## 4. Treebank Conversion Algorithm

To convert the PerTreeBank into its parallel dependency-based treebank, the structure of the treebank in the original format should be changed into the CoNLL shared task format. In this format, each lexical entry of the sentence is represented on one line and its relevant morpho-syntactic and semantic information is organized in separate columns (10 columns in CoNLL 2006 data format, and 14 columns in CoNLL 2009 data format). Algorithm 1 is employed to convert the PerTreeBank into its parallel dependency-based treebank, called the Dependency PerTreeBank (DPTB).

In the conversion process, we employ a depth-first searching approach to identify the constituents. In this process, each tree is traversed to find sibling leaves. Each pair of the sibling leaves comprises a dependency relation in which the head is determined using a head table. The head table is provided by extracting the grammar rules from the original treebank. A modified version of the Stanford Typed Dependencies (de Marneffe and Manning, 2008) is used for determining the types of the dependency relations. Appendix 1 represents the hierarchy of the dependency relations. After recognizing each pair of the sibling leaves as a dependency relation, the leaves are removed and the

**Algorithm 1** Converting a constituency treebank into a dependency treebank

**Input:** A tree of a sentence from the HPSG-based treebank for Persian,
The Head Table (HT),
The modified version of Dependency Relations (DR) based on the Stanford Typed Dependencies (de Marneffe and Manning, 2008)

**if** Projective option selected **then**
  **repeat**
    Step1: Traverse the tree to find a pair of sibling leaves
    Step2: Select the head node of the sibling leaves based on HT
    Step3: Detect the DR between the sibling leaves based on the head-daughter dependency relations
    Step4: Write the corresponding DR according to the CoNLL format
    Step5: Remove both of the leaves and transfer the head information to the parent node
  **until** The tree root node is met
**else if** Non-projective option selected **then**
  **repeat**
    Do Step1
    **if** The parent node has a slashed element **then**
      Transfer the information of the leaf along with the slashed element to the parent node as 'np-head'
    **else if** The parent node has the Head-Filler relation **then**
      Use the 'np-head' of the leaf with a slashed element instead of the head
    **end if**
    Do Steps 2 to 5
  **until** The tree root node is met
**end if**
Write the ROOT relation for the head of the tree
Sort the CoNLL format according to the sequence of the words in the input sentence

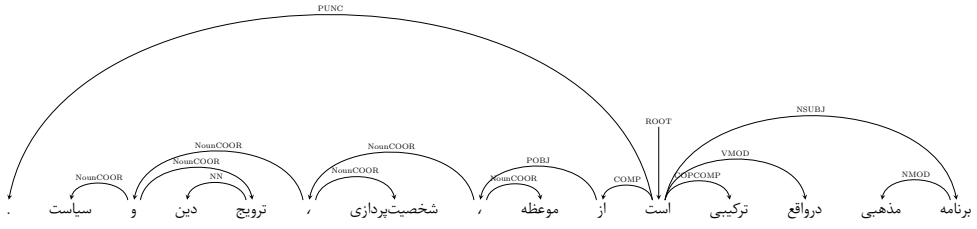


Figure 2: Projective dependency relation of Example 1

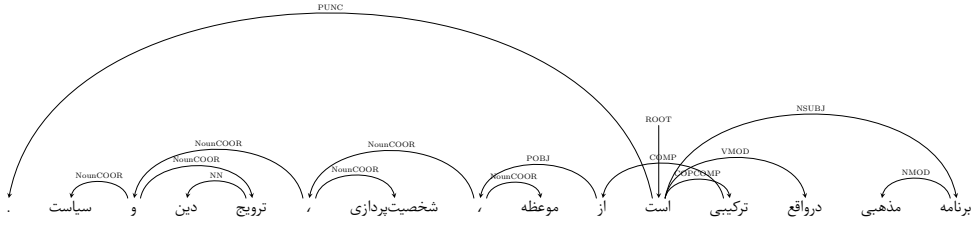


Figure 3: Non-projective dependency relations of Example 1

information of the head node is transferred to the parent node. The process of removing the leaves in each step results in new leaves which are in fact the old parents, and they need to be processed in the next steps. The conversion step continues in a loop to process all nodes of the tree, and it halts when the root of the tree is reached.

Since the positions of the slashed elements are determined in the trees (displayed by the '$-nid$' functional label on trees as represented in Figure 1), it is possible to convert them into non-projective trees as well. In this conversion, after traversing a tree to find a pair of sibling leaves, if a parent node contains a slashed element, the information of the slashed element in addition to the head node is transferred to the parent node. To draw a non-projective tree, it is important to keep the information of the leaf that has a slashed element. This information, which is related to the head of a non-projective tree, is temporally kept in a variable, which we called it 'np-head', and the information in the variable is transferred to upper nodes until the Head-Filler relation is met. When the slashed element is bound, the 'np-head' information is retrieved, and it is used for defining the dependency relation.

Algorithm 1 makes it possible to automatically create a dependency-based treebank from an existing HPSG-

based treebank without any human intervention. The size of the converted treebank depends on the size of the original data set. The converted data set can be used for training statistical dependency parsers.
Figures 2 and 3 represent the projective and non-projective dependency trees of the constituency tree in Figure 1 created through the automatic conversion.

## 5. Dependency Treebanks for Persian

In addition to our dependency-based treebank, there are two other Persian dependency treebanks, namely the Uppsala Persian Dependency Treebank (UPDT) (Seraji et al., 2012) and the Persian Dependency Treebank (PerDT) (Rasooli et al., 2013). Both of these dependency treebanks are developed via bootstrapping approaches. In this section, we briefly explain the general properties of each treebank. Next, we focus on four phenomena, namely clitics, multiple words, coordination phrases, and determiner phrases along with their corresponding annotation schemes.

**Treebank Quantity and Dependency Relations**: Currently DPTB contains 1,028 sentences (27,026 word tokens), and 49 labels are used for defining the dependency relations. UPDT contains 6,000 sentences (151,671 word tokens), and the dependency relations are defined by 102 labels. The dependency relations in these two dependency treebanks are modified versions of the dependency relations in de Marneffe and Manning (2008). PerDT contains 29,982 sentences (498,081 word tokens), and 46 labels are used for defining the dependency relations.

**Part-of-Speech Tags and Lemmatization**: In the three treebanks, morpho-syntactic information about the words is available in their Part-of-Speech (POS) tags, but there are differences between the treebanks regarding the granularity degree of this information. DPTB contains more fine-grained POS tags from the Bijankhan Corpus[1] (Bijankhan, 2004) than the other two treebanks. In terms of the availability of lemma information, DPTB and PerDT contain lemmas, but UPDT is not lemmatized.

**Clitics**: Clitics have two functions: morphological, and syntactic. Ezāfe[2] is an example of a morphological clitic. Ezāfe is available in the POS tags of DPTB, but this information is overlooked in UPDT and PerDT.
Possessive or object pronouns as well as copulative verbs are examples of syntactic clitics. In DPTB, these kinds of clitics are split from their hosts to represent an accurate syntactic analysis, whereas in UPDT and PerDT, these sorts of clitics remain unanalyzed.

**Projectivity vs Non-projectivity**: DPTB can have both projective and non-projective trees for discontinuous constructions, whereas in UPDT and PerDT only non-projective trees exist.

[1] http://ece.ut.ac.ir/dbrg/bijankhan/

[2] Ezāfe is an enclitic which is pronounced but mostly not written.

**Multiple Words**: DPTB and UPDT provide an internal syntactic analysis for light verb constructions, while light verb constructions are mostly considered as one token in PerDT except the cases where there is a long distance between the pre-verbal element and the light verb. In these cases, the pre-verbal element dominates the intervening elements as its argument. For instance, 'تبدیل کردن' /tabdil kardan/ 'to change' and 'دچار کردن' /dočār kardan/ 'to afflict' are two compound verbs that consist of a pre-verbal noun, which might take intervening elements as its argument, and a light verb. The auxiliary verb 'خواستن' /xāstan/ 'will, would' might appear between the two elements as well. This auxiliary verb is underlined in Examples 2 to 4 from DPTB, UPDT, and PerDT, respectively. In DPTB and UPDT, the auxiliary is recognized as one token and the dependency relation 'AUX' is defined for it, whereas in PerDT the auxiliary is attached to the light verb and it is remained unanalyzed.

(2) در این کارخانه روزانه ۴ تن زباله جامد به کود طبیعی خوشبو تبدیل خواهد شد .

dar in kārxāne ruzāne 4 ton zobāle-ye jāmed
in this plant daily 4 ton garbage-EZ dry
be kud-e tabiʔi-ye xošbu
to fertilizer-EZ natural-EZ sweet-smelling
tabdil xāhad šod
change will.3SG become.3SG

'In this plant, 4 tons of dry garbage will be turned into natural, sweet-smelling fertilizer daily.'

(3) مطالب نادرست را افراد مغرض و سخن‌چین و جاسوس به تو رسانده‌اند که می‌خواهند جامعه را دچار تفرقه کنند .

matāleb-e nādorost rā afrād-e
items-EZ incorrect DOM individuals-EZ
moqrez va soxančin va jāsus be
malevolent and rumor-mongering and prying to
to resānde-and ke mi-xāh-and
you transmitted-3PL that IMPF-will-3PL
ǰāmeʔe rā dočār-e tafraqe kon-and
society DOM cause-EZ discord do-3PL

'Malevolent, rumor-mongering, prying individuals have transmitted incorrect items to you, desiring to afflict society with discord.

(4) تعریف‌های غلوآمیز و نقد غیرکارشناسی شما را دچار جهل مرکب خواهدکرد .

tʔrif-hā-ye qolovāmiz va naqd-e
praise-PL-EZ exaggerating and criticism-EZ
qeyrekāršenāsi šomā rā dočār-e
unprofessional you DOM cause-EZ
jahl-e morakkab xāhad_kard
bewilderment-EZ complex will.3SG_did.3SG

'Exaggerated praise and unprofessional criticism will afflict you with deep bewilderment.'

Figures 4 to 6 represent the dependency trees of Examples 2 to 4 based on the annotation schemes and dependency labels of DPTB, UPDT, and PerDT.

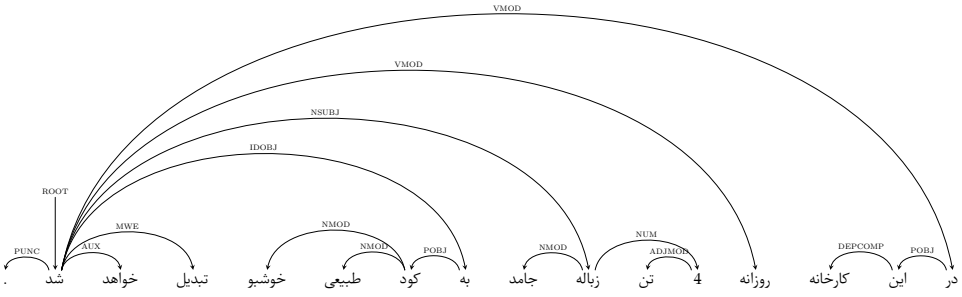


Figure 4: Dependency tree of Example 2 from DPTB

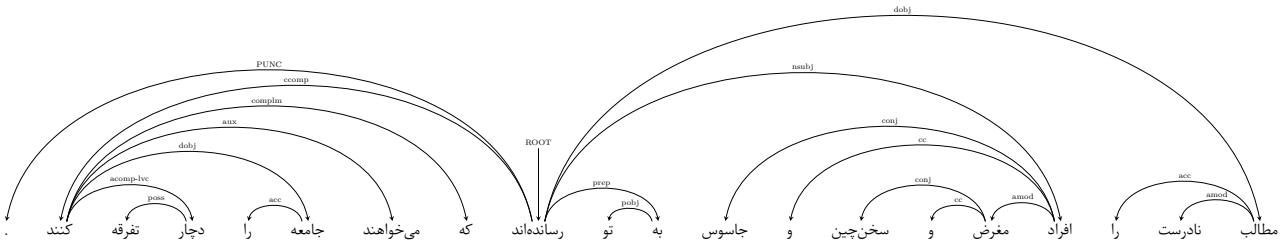


Figure 5: Dependency tree of Example 3 from UPDT

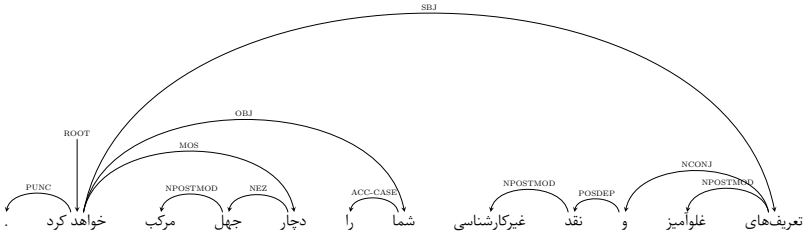


Figure 6: Dependency tree of Example 4 from PerDT

**Coordination Phrases**: In DPTB, a coordination is the head of a coordination phrase as represented in Figure 2, whereas in UPDT a coordination is not a head at all, and in PerDT a coordination is the head of the second coordinated element. Figures 5 and 6 represent the annotation scheme of a coordination phrase in UPDT and PerDT, receptively.

**Determiner Phrases**: In DPTB, a determiner, such as 'همه' /hame/ 'all, every', is the head of a determiner phrase, but this is not the case in UPDT and PerDT where a noun is the head. The annotation scheme of Examples 5 to 7, which contain the determiner 'همه' /hame/ 'all, every', is represented in Figures 7 to 9, respectively.

(5) در دنیای بورن همه چیز می‌تواند اتفاق بیفتد :

dar donyā-ye born hame čiz mi-tavān-ad
in world-EZ Born every thing IMPF-can-3SG
ettefāq bey-oft-ad
happen SUBJ-fall-3SG

'In Born's world everything can happen :'

(6) در غرب همه چیز حول و حوش مادیات دور می‌زند .

dar qarb hame čiz hol va hoš-e
in west every thing about and periphery-EZ
māddi-yāt dor mi-zan-ad
material-PL around IMPF-hit-3SG

'In the west, everything revolves around material things.'

(7) همه چیز را دوباره بشور !

hame čiz rā dobāre be-šur
every thing DOM again IMP-wash

'Wash everything again!'

## 6. Evaluation

In this section, we report the performance of the Malt (Nivre et al., 2006) and Mate (Bohnet, 2009) statistical dependency parsers trained with the whole set of data for the three Persian dependency treebanks. We use the 10-fold cross-validation for the evaluation. Table 1 represents the performance of the parsers for

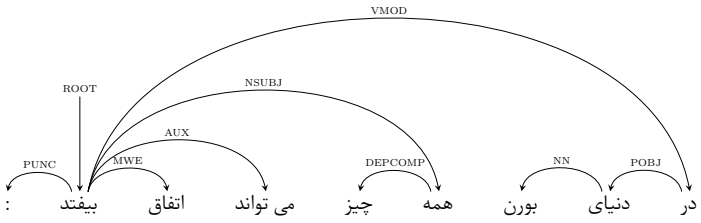


Figure 7: Dependency tree of Example 5 from DPTB

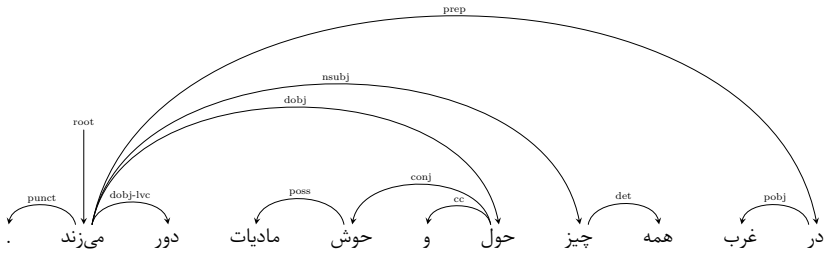


Figure 8: Dependency tree of Example 6 from UPDT

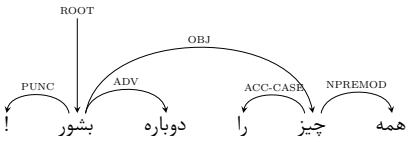


Figure 9: Dependency tree of Example 7 from PerDT

DPTB, UPDT, and PerDT using gold POS tags in the evaluations to reduce the impact of tagging on parsing. As can be observed from the results, for all of the treebanks the Mate parser outperforms the Malt parser. Moreover, the parsers trained with UPDT and PerDT perform better than the parsers train with DPTB. This is not a surprising result, because both UPDT and PerDT contain more data to build more accurate grammar models. PerDT performs the best due to having more annotated data and more coarse-grained dependency labels which result in a higher accuracy rate in predicting the dependency labels. However, the advantage of DPTB, compared with the other two dependency treebanks, is that this language resource is created without any human effort, whereas UPDT and PerDT are developed with human intervention. Additionally, increasing the amount of data in the PerTreeBank will automatically increase the size of DPTB.

Table 1: Dependency parsing results

| Tool | Treebank | Labeled Attachment | Unlabeled Attachment | Label Accuracy |
|---|---|---|---|---|
| Malt | DPTB | 76.63 | 82.04 | 81.10 |
| | UPDT | 79.13 | 83.71 | 86.98 |
| | PerDT | 83.84 | 87.86 | 86.19 |
| Mate | DPTB | 80.17 | 84.38 | 85.64 |
| | UPDT | 81.36 | 85.21 | 90.03 |
| | PerDT | 90.29 | 92.69 | 92.53 |

## 7. Summary

In this paper, we introduced an algorithm that converts an HPSG-based treebank into its parallel dependency-based treebank. This algorithm was applied on the PerTreeBank, an HPSG-based treebank for Persian. The main advantage of the converted dependency-based treebank over the other existing dependency treebanks for Persian was that it was created relatively effortlessly and without human intervention. The converted dependency-based treebank could be used for training statistical dependency parsers.

## 8. Ackowledgement

In this research, Masood Ghayoomi is funded by the German Research Foundation (DFG) via the SFB 732 “Incremental Specification in Context”.

## References

Avgustinova, Tania and Zhang, Yi. (2010). Conversion of a Russian dependency treebank into HPSG derivations. In *The 9th International Workshop on Treebanks and Linguistic Theories*, pages 7–18.

Bijankhan, Mahmood. (2004). naqše peykarehāye zabāni dar neveštane dasture zabān: mo'arrefiye yek narmafzāre rāyāneyi ["The role of corpora in writing a grammar: Introducing a software"]. *Journal of Linguistics*, 19(2):48–67.

Bohnet, Bernd. (2009). Efficient parsing of syntactic and semantic dependency structures. In *Proceedings of the 13th Conference on Computational Natural Language Learning: Shared Task*, pages 67–72, Stroudsburg, PA, USA. Association for Computational Linguistics.

Bos, Johan, Bosco, Cristina, and Mazzei, Alessandro. (2009). Converting a dependency-based treebank to a categorial grammar treebank for Italian. In Passarotti, M., Przepiórkowski, Adam, Raynaud, S., and van Eynde, Frank, editors, *Proceedings of the 8th Workshop on Treebanks and Linguistic Theories*, pages 27–38.

Burke, Michael. (2006). *Automatic Treebank Annotation for the Acquisition of LFG Resources*. Ph.D. thesis, Dublin City University.

Cahill, Aoife, Mccarthy, Mairead, Genabith, Josef Van, and Way, Andy. (2002). Automatic annotation of the Penn treebank with LFG f-structure information. In *LREC Workshop on Linguistic Knowledge Acquisition and Representation: Bootstrapping Annotated Language Data*, pages 8–15.

Candito, Marie, Crabbé, Benoît, and Denis, Pascal. (2010). Statistical French dependency parsing: Treebank conversion and first results. In *Proceedings of the 7th International Conference on Language Resources and Evaluation*, pages 1840–1847, La Valletta, Malta. European Language Resources Association.

Çakici, Ruken. (2005). Automatic induction of a CCG grammar for Turkish. In *Proceedings of the ACL Student Research Workshop*, pages 73–78.

Chanev, Atanas, Simov, Kiril, Osenova, Petya, and Marinov, Svetoslav. (2006). Dependency conversion and parsing of the BulTreeBank. In *Proceedings of the LREC workshop Merging and Layering Linguistic Information*, pages 16–23.

Chen, John, Bangalore, Srinivas, and Shanker, Vijay K. (2006). Automated extraction of tree-adjoining grammars from treebanks. *Natural Language Engineering*, 12(3):251–299.

Cramer, Bart and Zhang, Yi. (2010). Constraining robust constructions for broad-coverage parsing with precision grammars. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 223–231, Stroudsburg, PA, USA. Association for Computational Linguistics.

de Marneffe, Marie-Catherine and Manning, Christopher D. (2008). The Stanford typed dependencies representation. In *Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ghayoomi, Masood. (2012). Bootstrapping the development of an HPSG-based treebank for Persian. *Linguistic Issues in Language Technology*, 7(1).

Guo, Yuqing, van Genabith, Josef, and Wang, Haifeng. (2007). Treebank-based acquisition of LFG resources for Chinese. In *Proceedings of the 12th International Lexical Functional Grammar Conference*, Stanford, California. CSLI Publications.

Hockenamier, Julia and Steedman, Mark. (2007). CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn treebank. *Computational Linguistics*.

Hockenmaier, Julia. (2003). *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, School of Informatics, University of Edinburgh, Edinburgh, Scotland, UK.

Marcus, Mitchell P., Santorini, Beatrice, and Marcinkiewicz, Mary Ann. (1993). Building a large annotated corpus of English: the Penn treebank. *Computational Linguistics*, 19(2).

Miyao, Yusuke, Ninomiya, Takashi, and Tsujii, Jun'ichi. (2005). Corpus-oriented grammar development for acquiring a head-driven phrase structure grammar from the Penn Treebank. In *Proceedings of the 1st International Joint Conference on Natural Language Processing*, pages 684–693, Berlin, Heidelberg. Springer-Verlag.

Nivre, Joakim, Hall, Johan, and Nilsson, Jens. (2006). MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, pages 2216–2219.

Oepen, Stephan, Flickinger, Dan, Toutanova, Kristina, and Manning, Christoper D. (2002). LinGO Redwoods. A rich and dynamic treebank for HPSG. In *Proceedings of the 1st Workshop on Treebanks and Linguistic Theories*, pages 139–149.

Pollard, Carl J. and Sag, Ivan A. (1994). *Head-driven Phrase Structure Grammar*. University of Chicago Press.

Rambow, Owen, Creswell, Cassandre, Szekely, Rachel, Taber, Harriet, and Walker, Marilyn. (2002). A dependency treebank for English. In *3rd International Conference on Language Resources and Evaluation*, pages 857–863.

Rasooli, MohammadSadegh, Kouhestani, Manouchehr, and Moloodi, Amirsaeid. (2013). Development of a Persian syntactic dependency treebank. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 306–314, Atlanta, Georgia.

Schluter, Natalie. (2011). *Treebank-based Deep Grammar Acquisition for French Probabilistic Parsing Resources*. Ph.D. thesis, Dublin City University.

Seraji, Mojgan, Megyesi, Beata, and Nivre, Joakim. (2012). Dependency parsers for Persian. In *Proceedings of the 10th Workshop on Asian Language Resources*, pages 35–43.

Shen, Libin and Joshi, Aravind K. (2004). Extracting deeper information from richer resource: EM models for LTAG treebank induction. In *International Joint Conference on Natural Language Processing*.

Tateisi, Yuka, Torisawa, Kentaro, Miyao, Yusuke, and Tsujii, Jun'ichi. (1998). Translating the XTAG English grammar to HPSG. In *The 4th International Workshop on Tree Adjoining Grammars and Related Formalisms*, pages 172–175.

Tounsi, Lamia, Attia, Mohammed, and van Genabith, Josef. (2009). Automatic treebank-based acquisition of Arabic LFG dependency structures. In *Proceedings of the European Chapter of the Association for Computational Linguistics 2009, Workshop on Computational Approaches to Semitic Languages*, pages 45–52, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yoshinaga, Naoki and Miyao, Yusuke. (2002). Grammar conversion from ltag to hpsg. In *Proceedings of the Sixth ESSLLI Student Session*, pages 309–324.

Yu, Kun, Yusuke, Miyao, Wang, Xiangli, Matsuzaki, Takuya, and Tsujii, Junichi. (2010). Semi-automatically developing Chinese HPSG grammar from the Penn Chinese treebank for deep parsing. In *Proceedings of the International Conference on Computational Linguistics*, pages 1417–1425, Beijing, China.

**Appendix 1**

- *BOT* - bottom
  - *ROOT* - root
  - *DEP* - dependent
    - *PUNC* - punctuation
    - *ARG* - argument
      - *COMP* - complement
        - *ADJCOMP* - adjective complement
        - *ADVCOMP* - adverb complement
        - *AUX* - auxiliary
        - *COPCOMP* - copula complement
        - *COORCOMP* - coordination complement
        - *CCOMP* - clausal complement
        - *DETCOMP* - determiner complement
        - *INTCOMP* - interjection complement
        - *NN* - nominal complement
          - *POSS* - possession complement
          - *POSSESSIVE* - possessive complement
          - *REFLCOMP* - reflexive complement
        - *OBJ* - object
          - *DOBJ* - direct object
          - *IDOBJ* - indirect object
          - *POBJ* - preposition object
          - *XOBJ* - dropped object
        - *SUBJ* - subject
          - *NSUBJ* - nominal subject
          - *RELSUBJ* - relativizer subject
          - *XSUBJ* - dropped subject
      - *MOD* - modifier
        - *ADJMOD* - adjective modifier
        - *CLMOD* - clausal modifier
        - *DETMOD* - determiner modifier
        - *MWE* - multi-word expression modifier
        - *INTMOD* - interjection modifier
        - *NMOD* - noun modifier
          - *NUM* - numeric modifier
          - *APPOS* - appositional modifier
          - *REFLMOD* - reflexive modifier
        - *NUMBER* - element of compound number numeric
        - *PREPMOD* - preposition modifier
        - *PURPCL* - purpose clause modifier
        - *RCMOD* - relative clause modifier
        - *RES* - residual modifier
        - *VMOD* - verb modifier
    - *CC* - coordination
      - *ADJCOOR* - adjective coordination
      - *ADVCOOR* - adverb coordination
      - *CLCOOR* - clause coordination
      - *DETCOOR* - determiner coordination
      - *NOUNCOOR* - noun coordination
      - *NUMCOOR* - number coordination
      - *PREPCOOR* - preposition coordination
      - *RESCOOR* - residual coordination
      - *VERVCOOR* - verb coordination
    - *MARKER* - marker
      - *ACC* - accusative marker
      - *COMPM* - complementizer marker
      - *MARK* - clausal marker

.