Multitask Easy-First Dependency Parsing: Exploiting Complementarities of Different Dependency Representations

Yash Kankanampati, Joseph Le Roux,[†] Nadi Tomeh,[†] Dima Taji,[‡] Nizar Habash[‡]
Indian Institute of Technology - Dhanbad
Université Sorbonne Paris Nord,[†] New York University Abu Dhabi[‡]
yash.15je001733@am.iitism.ac.in
{leroux,tomeh}@lipn.fr, {dima.taji,nizar.habash}@nyu.edu

Abstract

We present a parsing model for projective dependency trees which takes advantage of the existence of complementary dependency annotations for a language. This is the case for Arabic with the availability of CATiB and UD treebanks. Our system performs syntactic parsing according to both annotation types jointly as a sequence of arc-creating operations following the Easy-First approach, and partially created trees for one annotation type are also available to the other as features for the score function. This method gives error reduction of 9.9% on CATiB and 6.1% on UD compared to a single-task baseline, and ablation tests show that the main contribution of this reduction is given by sharing tree representation between tasks, and not simply sharing BiLSTM layers as is usually performed in NLP multitask systems.

1 Introduction

Dependency parsing is the task of assigning a syntactic structure to a sentence by linking its words with binary asymmetrical typed relations. In addition to syntactic information, dependency representations encode some semantic aspects of the sentence which make them important to downstream applications including sentiment analysis (Tai et al., 2015) and information extraction (Miwa and Bansal, 2016).

In this paper we are interested in Arabic dependency parsing for which two formalisms have been developed (See §2). The first is the Columbia Arabic Treebank (CATiB) representation (Habash and Roth, 2009), which is inspired by Arabic traditional grammar and which focus on modeling syntactic and morpho-syntactic agreement and case assignment. The second is the Universal Dependency (UD) representation (Taji et al., 2017), which has relatively more focus on semantic/thematic relations, and which is coordinated in design with a number of other languages (Nivre et al., 2017). While previous work on Arabic dependency parsing (Marton et al., 2013; Taji et al., 2017) tackled these formalisms separately, we argue that they stand to benefit from multitask learning (MTL) (Caruana, 1993). MTL allows for more training data to be exploited while benefiting from the structural or statistical similarities between the tasks. We therefore propose to learn CATiB and UD dependency trees jointly on the same input sentences using parallel treebanks.

Deep neural networks are particularly suited for multitask scenarios via straightforward parameter and representation sharing. Some hidden layers can be shared across all tasks while output layers are kept separate. In fact, most deep learning architectures for language processing start with sequential encoding components such as BiLSTMs or Transformer layers which can be readily shared across multiple tasks. This approach is widely applicable even to tasks that use very different formalisms and do not have parallel annotations (Søgaard and Goldberg, 2016; Hashimoto et al., 2017). This type of sharing has also been shown to benefit (semantic and syntactic) dependency parsing, both transition-based (Stymne et al., 2018; Kurita and Søgaard, 2019) and graph-based (Sato et al., 2017; Lindemann et al., 2019). In addition to simple parameter sharing, joint inference across multiple tasks has been shown to be beneficial. Peng et al. (2017) perform decoding jointly across multiple semantic dependency formalisms with cross-task factors that score combinations of substructures from each (Peng et al., 2017). Joint inference however comes with increased computational cost. To address this issue, we introduce a multitask learning model for dependency parsing (§3.2). This model is based on greedy arc selection similar to the neural easy-first approach proposed in (Kiperwasser and Goldberg, 2016) (§3). We use tree-structured LSTMs to encode substructures (partial trees) in each formalism which are then concatenated across tasks and scored jointly. Hence, we model interactions between substructures across tasks while keeping computational complexity low thanks to the easy-first framework. Furthermore, this approach enables the sharing of various components between tasks, a richer sharing than the mere sequential encoder sharing found in most multitask systems (Kurita and Søgaard, 2019). Our multitask architecture outperforms the single-task parser on both formalisms (§4.3). The parser itself is open-source and can be found at `https://github.com/yash-reddy/MEF_parser`.

Beside efficiency, the tree-structured LSTM easy-first framework provides several advantages which makes it appealing in our settings. New arc selection decisions are conditioned on encoded representations of partially parsed structures in both formalisms with the latest information at each step. Since some word attachments are harder to find in one formalism than the other (longer range, ambiguous relations, etc.), we suppose that looking at the substructure involving such a word in one formalism may help make better decisions in the other. We do not need to postulate any priority between the tasks nor that all attachment decisions must be taken jointly which is computationally expensive, we leave the exact flow of information to be learned by the model. Additionally, (Kurita and Søgaard, 2019) showed that even when no easy-first strategy is hard-wired into their multitask semantic dependency parser, it gets nevertheless learned from the data in a reinforcement learning framework.

Some possible enhancements to our model are explored in §6.

Summary of contributions In this paper, (i) we propose a new multitask dependency parsing algorithm, based on easy-first hierarchical LSTMs, capable of decoding a sentence into multiple formalisms; (ii) we show that our joint system outperforms the single-task baseline on both CATiB and UD Arabic dependency treebanks; and (iii) demonstrate experimentally that linguistic information available in each formalism helps make better predictions for the other by showing that the parser learns to leverage information from each dimension to parse the other dimension better.

2 Linguistic Background: CATiB vs. UD

The CATiB and the Arabic UD treebanks are currently the two largest Arabic dependency treebanks.[1] As both treebanks are in dependency representations, that leads to them sharing a number of similarities. However, there are a few differences between the two treebanks stemming from the granularity of their tag sets and their specific definitions of dependencies.

Granularity of tags One of the design features of CATiB is fast annotation (Habash and Roth, 2009), hence it has only six POS tags and eight dependency relations. On the other hand, UD aims to accommodate constructions in universal languages (Nivre et al., 2017), and therefore have finer-grained tagsets with 17 POS tags and 37 basic dependency relations. Naturally, a tag in CATiB, whether a POS tag or a dependency relation, may correspond to a number of tags in UD. Figure 1 illustrates that mapping through a number of examples. UD's noun (NOUN), adjective (ADJ), and number (NUM) tags correspond to CATiB's nominal (NOM) tag. Similarly, when it comes to modifiers in UD, if they are headed by a verb then they are oblique nominals (OBL), if they are headed by a noun then they can be nominal modifiers (NMOD), adjectival modifiers (AMOD), or numeric modifiers (NUMMOD), depending on the modifier's POS tag. However, in CATiB, they are all modifiers (MOD). On the other hand, the Idafa relation (IDF) in CATiB is also a nominal modifier in UD, but for this particular structure UD uses the possessive subtype (NMOD:POSS) to distinguish it from other nominal modifiers.

Definition of a dependency The philosophical difference between CATiB and UD is in how they define a dependency relation. CATiB focuses on modeling the assignment of case to make the tree structures

[1]For more information on Arabic natural language processing challenges, see Habash (2010).

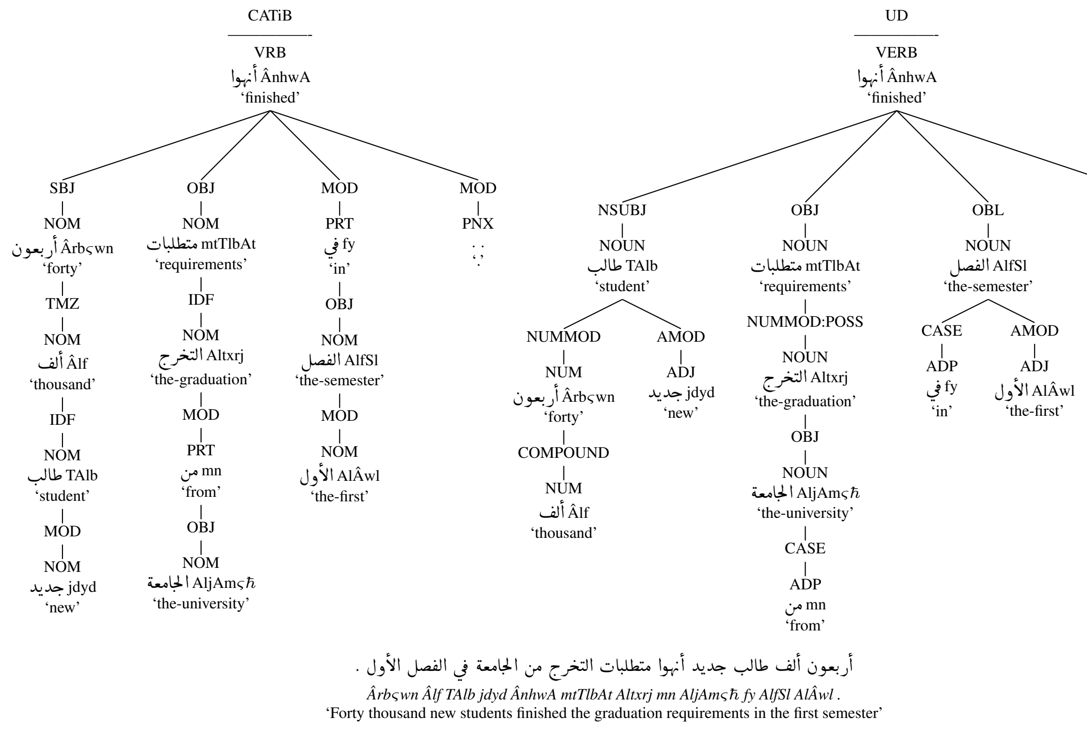

Figure 1: An example dependency tree in CATiB and UD representations

closer to traditional Arabic grammar analysis (Habash and Roth, 2009). As a result, function words tend to head their phrase structures. On the other hand, UD aims to minimize the differences between languages with different morphosyntactic structures, and therefore focus on the meaning (Nivre, 2016). This often makes content words the heads of phrase structures. We can see some of those differences in the examples in Figure 1, the most prominent of which may be prepositional phrase constructs. In CATiB, particles head these phrases since they modify the case assignment of the words that follow them. In contrast, particles in UD attach low under the content head of these phrases to keep the focus on the semantic meaning of the sentences.

Due to these differences and similarities in representation, we hypothesize that parsing the two treebank formalism along side each other will help improve both parsing outcomes.

3 Model

We briefly describe the single task Easy-First (EF) parsing algorithm Kiperwasser and Goldberg (2016) and its components, then we discuss the multitask extension (MEF) inspired by (Constant et al., 2016) adapted to the tree-structured LSTM in terms of the updated parsing algorithm and its components.

3.1 Single-task Model

The EF parsing model builds dependency trees bottom-up. Intuitively, it can be seen as a greedy version of the Eisner algorithm (Eisner, 1996) where each span contains at most one subtree and where each subtree, once created, is fixed and must be a part of the final structure. The algorithm maintains a sequence of pending subtrees. Two adjacent subtrees in the sequence may combine, by adding an arc from the root of one subtree to the other, and be replaced by a new (bigger) subtree. Since there are few subtrees to consider — at most $n + 1$ pending subtrees for a sentence of n words — the context of each decision, *ie* each arc creation between subtrees, can depend on the whole subtrees involved provided they can be encoded with a fixed representation. This was first noted in (Kiperwasser and Goldberg, 2016) where a subtree root was represented via the latent states of two LSTM recurrent networks, each encoding the sequence of modifiers in one direction (left or right) going outwards. Since modifiers are themselves encoded this way, this recursion effectively represents the whole subtree. EF relies on several

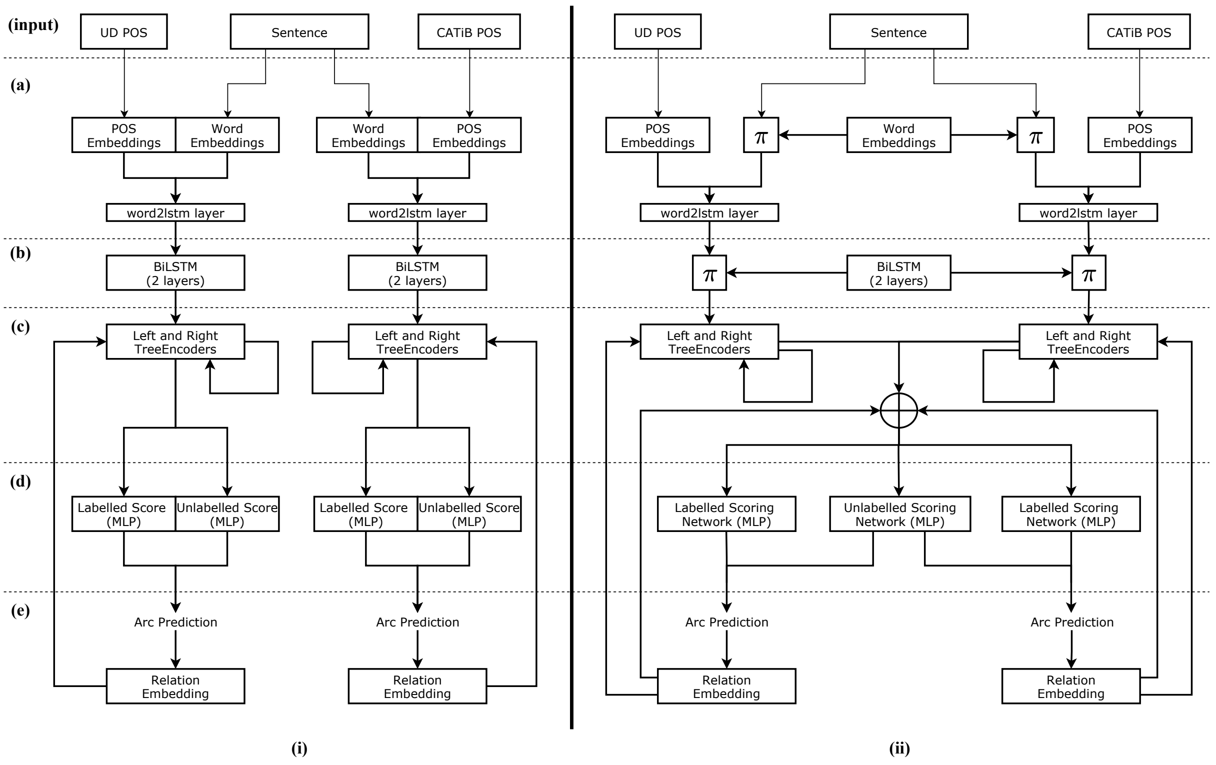

Figure 2: A component-wise comparison of the disjoint Easy-First Parsers (i) and our maximally connected system (ii). We use the component-based representations that showcase the interactions between various components to illustrate the differences in connectivity. The rectangle with π represents a proxy for a shared component and $\oplus$ represents concatenation.

components which compute representation at different stages of the calculation. These components can be seen in Figure 2 where they drawn as rectangles in areas (a) to (e). We review them before presenting the algorithm.

Word representation (a) and context (b) For each position i in the sentence $t = t_0 \dots t_n$, its word embedding and its POS tag embedding are looked up and concatenated, and then passed through an affine transformation. The sequence of such vectors is fed to a 2-layer bidirectional LSTM (Graves et al., 2005) in order to obtain a sequence of contextual token representations $v = v_0 \dots v_n$.

Tree representation (c) and (e) The representation of a pending tree root h_i is the concatenation of the representations of its two sequences of modifiers (one for each side) l_i and r_i: $h_i = [l_i; r_i]$. The sequence of left modifier trees $l_i = v_i, m_1, \dots, m_{k_l}$ ordered from the head outward is represented by the latent state of an LSTM network where each left modifier m_k is first transformed before it is read by the LSTM as $m'_k = \tanh(W[m_k; x_k] + b)$ where x_k is the embedding of the label of the arc from w_i to w_k. A similar encoding, depending on a second LSTM, gives r_i. At the initialization step of EF, pending single-node subtrees are encoded with this method, where the sequence of modifiers at each position i is restricted to v_i on both sides.

Arc scoring (d) Since it would be intractable to take the complete sequence h into account when considering the weights of possible labeled arcs between adjacent subtrees h_i and h_j, the context of decision is restricted to a window of k previous and k following trees, with k small (set to 2 in our experiments, after having experimented with $k = 1$ and $k = 3$.). . Left (resp. right) context lc (resp. rc) is simply represented as the concatenation of the k trees before h_i (resp. after h_j) in the sequence, padded if necessary. Moreover, the scoring function is decomposed as the sum of the unlabeled score and the labeled score. For instance, a right arc going from w_i to w_j labeled with l, the scoring function is defined as: $s(h_i, h_j, \rightarrow, l, lc, rc) = s_U(h_i, h_j, \rightarrow, lc, rc) + s_L(h_i, h_j, \rightarrow, l, lc, rc)$. These two functions can be compactly implemented by feed-forward multi-layer perceptrons which given trees and contexts return a vector of scores for all combinations of labels and directions.

Algorithm More formally, a sentence is represented as a sequence of tokens $t = t_0 \dots t_n$ where t_i such that $i \geq 1$ is the i^{th} token of the sentence and t_0 is a dummy root symbol. EF starts by converting each token to a pending tree consisting of a single node t_i and computes its fixed-size vector representation h_i. This gives a sequence of trees[2] $h = h_0 \dots h_n$.

Then, for n steps, EF goes through all pairs of adjacent pending trees (h_i, h_j) in the sequence h and predicts the maximum scoring labeled arc. Scores are interpreted as quantifying confidence or *easiness* of the decision. If the triple (h_i, h_j, l) gives the highest scoring arc and this arc is right-oriented (resp. left-oriented) then the arc $w_i \rightarrow^l w_j$ is created (resp. $w_j \rightarrow^l w_j$), h_j (resp. h_i) is removed from the sequence, and finally h_i (resp. h_j) is updated to reflect the addition of a rightmost (resp. leftmost) modifier. This process stops after n steps, when only one tree remains, and the set of created arcs is returned. Parsing amounts to a sequence of arc-creating actions, depending on the scoring function and its parameters. These can be learned from examples via gradient descent in a supervised setting using teacher forcing and max-margin objective. We refer interested readers to (Kiperwasser and Goldberg, 2016) for more details.

3.2 Multitask Model

Since we are interested to see how the flow of syntactic information can be shared between CATiB and UD, we adapt the neural EF (Kiperwasser and Goldberg, 2016) to *multidimensional* EF (MEF), following the nomenclature of Constant et al. (2016), with two tasks, one for each syntactic representation. We now review the changes from EF to MEF, looking at the algorithm and at the components.

Algorithm MEF is similar to EF but operates on two sequences of trees $h^{(c)}, h^{(u)}$ (one for CATiB, one for UD) which we assume to be built from the same sequence of tokens $t = t_0 \dots t_n$. MEF iterates for $2n$ arc-creating steps until only one tree remains in each sequence. Each step finds the maximum scoring labeled arc between pairs of adjacent trees as before, but it now visits the two sequences. Thus the change in the algorithm is minimal, but it should be noted that the construction of dependency trees can be interleaved. Of course, in order to take advantage of this added complexity, a task should be able to *peek* at the other to get further information. To this end we need to keep track of how a specific token and its partial tree representation is encoded in both tasks. If token t_i is represented by a subtree h_i in one task, we note $\bar{h_i}$ its counterpart, its representation for the other task. EF components may or may not make use of this *extra-dimensional* representation. This gives a variety of models with gradually increased sharing of representations, ranging from the model depicted in Figure 2 *(i)* where there is no sharing at all, to model *(ii)* with sharing at every stage of the architecture. We describe how this sharing is implemented.

Word representation and context The simplest form of sharing is at the token level. If shared, there is only one look-up table to store embeddings for both tasks, otherwise each task has its own table. The contextual token representation sharing has three degrees: we can share both biLSTM layers, the lower layer only, or none at all.

Arc scoring If sharing this component, we add counterparts as additional parameters. Taking the arc scoring example from the previous section $s(h_i, h_j, \rightarrow, l, lc, rc)$ becomes $s(h_i, h_j, \rightarrow, l, lc, rc, \bar{h_i}, \bar{h_j}, \bar{lc}, \bar{rc}, \bar{l})$ where $\bar{lc}$ and $\bar{rc}$ are the concatenations of the counterparts trees in left and right context. $\bar{l}$ is the embedding of the dependency label of the arc to w_j in the other task, or special vector indicating that its head selection has not be performed yet. This gives the system a way to learn to wait for the other task to take a decision.

Arc Selection We see that the multitask system has a choice of parsing dimensions to make. While the base parsing algorithm itself does not impose any constraint on which dimension should be preferred,

[2]In the following, we will abuse terminology and use the same notation to identify trees and their vectorized representations.

System Configurations				**CATiB DEV**		**UD DEV**	
Embeddings	**Context**	**Node Encodings**	**Scoring network**	**LAS**	**UAS**	**LAS**	**UAS**
Easy-First baseline (Nothing is shared) (Kiperwasser and Goldberg, 2016)				85.16	87.49	84.24	87.22
NotShared	NotShared	NotShared	Shared	85.26	87.29	83.99	87.02
NotShared	NotShared	Rep	NotShared	85.84	87.71	84.61	87.39
NotShared	NotShared	Rep	Shared	85.23	87.26	84.40	87.20
NotShared	OneOne	Rep+DepRel	NotShared	85.91	87.56	84.49	87.42
NotShared	OneOne	Rep+DepRel	Shared	85.64	87.60	84.49	86.95
NotShared	OneOne	NotShared	NotShared	85.11	87.52	84.31	87.25
NotShared	OneOne	NotShared	Shared	85.12	87.41	83.99	87.10
NotShared	Two	Rep	NotShared	85.99	87.84	84.59	87.25
NotShared	Two	Rep	Shared	85.85	87.67	84.31	87.19
NotShared	Two	Rep+DepRel	NotShared	85.99	87.74	84.49	87.32
NotShared	Two	Rep+DepRel	Shared	85.78	87.72	84.57	87.16
NotShared	NotShared	NotShared	NotShared	85.02	87.28	84.24	87.31
NotShared	NotShared	NotShared	Shared	85.02	87.39	84.13	87.23
NotShared	NotShared	Rep	NotShared	85.78	87.76	84.62	87.13
NotShared	NotShared	Rep	Shared	85.58	87.44	84.55	87.11
NotShared	OneOne	Rep+DepRel	NotShared	85.97	87.79	84.38	87.22
NotShared	OneOne	Rep+DepRel	Shared	85.75	87.36	84.30	87.07
Shared	OneOne	NotShared	NotShared	85.47	87.27	84.67	87.42
Shared	OneOne	NotShared	Shared	85.43	87.55	84.56	87.34
Shared	Two	Rep	NotShared	85.80	87.86	85.03	87.66
Shared	Two	Rep	Shared	85.60	87.57	84.67	87.53
Shared	Two	Rep+DepRel	NotShared	85.95	88.06	84.93	87.36
Shared	Two	Rep+DepRel	Shared	85.81	87.98	84.74	87.32
Shared	NotShared	NotShared	NotShared	85.16	87.95	84.71	87.33
Shared	NotShared	NotShared	Shared	84.96	87.92	84.58	87.21
Shared	NotShared	Rep	NotShared	86.44	88.39	**85.17**	**87.66**
Shared	NotShared	Rep	Shared	86.19	88.11	84.86	87.70
Shared	OneOne	Rep+DepRel	NotShared	**86.66**	**88.51**	85.16	87.72
Shared	OneOne	Rep+DepRel	Shared	86.23	88.26	84.90	87.38
Shared	OneOne	NotShared	NotShared	85.64	87.96	84.59	87.47
Shared	OneOne	NotShared	Shared	85.54	87.97	84.58	87.40
Shared	Two	Rep	NotShared	85.92	87.81	84.82	87.47
Shared	Two	Rep	Shared	85.93	87.97	84.68	87.44
Shared	Two	Rep+DepRel	NotShared	86.36	88.17	84.84	87.50
Shared	Two	Rep+DepRel	Shared	86.16	87.99	84.87	87.61

Table 1: Performance of systems with varying degrees of sharing across each component (See §4.2).

we explore strategies where one dimension is completely parsed before the other or the parser alternates between dimensions for the selection of the arc to be added to the corresponding partial parse forest (absolute parity). We experimented with absolute parity and freely learnt selection strategies, but found no significant differences between these strategies.

4 Experiments

4.1 Data

Both CATiB and UD are automatically-converted treebanks. CATiB is created by converting parts 1, 2, and 3 of the Penn Arabic Treebank (PATB) (Maamouri et al., 2004) constituency treebank to the CATiB dependency representation. The UD treebank is an automatic conversion from the CATiB treebank (Taji

Model	CATiB TEST		UD TEST	
	LAS	UAS	LAS	UAS
Easy-First baseline (Kiperwasser and Goldberg, 2016)	84.63	86.93	83.71	86.65
Joint System (REP)	85.96	87.93	**84.76**	87.17
Joint System (REP+DEPREL)	**86.15**	**88.08**	84.71	**87.19**

Table 2: LAS and UAS results for the best performing systems against baseline on test sets.

et al., 2017). Both treebanks follow the PATB tokenization, with Alif wasla normalization. Additionally, all numeral tokens were replaced by a unique NUM token.

Treebank Statistics In our experiments, we split the treebanks into TRAIN, DEV, and TEST following the guidelines detailed by Diab et al. (2013). Our treebanks have 1,986 trees in DEV (73,945 tokens), 1,963 trees in TEST (74,125 tokens), and after excluding the non-projective trees, 15,603 trees in TRAIN (577,564 tokens).

4.2 Experimental Configuration

Experiments were carried out with systems built using the exhaustive combination of the proposed component configuration. We use the same hyper-parameters for each component and training setup as Kiperwasser and Goldberg (2016) to keep the systems comparable, with the parameters initialized using a Xavier initialization. The training was carried out for 50 epochs using the Adam optimizer (Kingma and Ba (2014)) with learning rate, moving average for mean and the moving average for variance set to 0.001, 0.9 and 0.999 respectively. Following is the specification of the combinations of system design hyper-parameters we explored: (i) WORD EMBEDDING: Word embeddings are NOTSHARED (baseline) or SHARED; (ii) CONTEXT: BiLSTMs for contextual representations are NOTSHARED (baseline), ONEONE (first layer shared) , or TWO (both layers shared); (iii) ENCODINGS: Encoded representations are NOTSHARED (baseline), REP (only node representations shared), or REP+DEPREL (node and dependency relation representations shared); (iv) SCORING FUNCTION: Unlabelled Scoring Network is either NOTSHARED (baseline) or SHARED.

4.3 Results

We present the results of all models on the DEV set in Table 1, and the results of the best performing systems on the TEST set in Table 2. The Labelled and Unlabelled Attachment Scores (LAS and UAS) are computed with punctuation included. It can be seen that our proposed configurations show considerable improvements over single-task baseline models. The best systems are the ones in which the word embeddings are shared, the contextual representations are generated using one common and one task-specific BiLSTM layer, and the representations are shared with or without the corresponding dependency relation of the node. We denote these systems as Joint System (REP) and Joint System (REP + DEPREL) in subsequent analysis. We see that we improve upon both the CATiB and UD scores by considerable margins of 1.52 points and 1.05 points on LAS respectively.

4.4 Analysis of Model Architecture Settings

Since we explored the full space of combinations of a number of model architecture settings, we proceed to study the contributions of each of theses settings. In Table 3, we present the average LAS of all the systems that share a particular model setting, e.g., out of the 36 systems, 18 include the setting WORD EMBEDDING:NOTSHARED, and 18 WORD EMBEDDING:SHARED.

We see clearly from these settings that the best combinations for CATiB are WORD EMBEDDING:SHARED, CONTEXT:ONEONE, ENCODINGS:REP+DEPREL, and SCORING FUNCTION:NOTSHARED. UD best parameters are the same except for ENCODINGS:REP. In general sharing components is beneficial except for sharing the scoring function, which is reasonable. Since the values are small, we were interested in studying their degree of overlap with each other, i.e., how linear is their contributions. We consider the baseline features (as in the Kiperwasser and Goldberg (2016)'s parser)

		Average LAS		**Δ Baseline**	
		CATiB	UD	CATiB	UD
WORD EMBEDDING	NOTSHARED	85.57	84.67	0.00	0.00
	SHARED	85.85	85.08	0.28	0.41
CONTEXT	NOTSHARED	85.61	84.83	0.00	0.00
	ONEONE	85.79	84.95	0.18	0.12
	TWO	85.72	84.85	0.12	0.02
ENCODINGS	NOTSHARED	85.26	84.66	0.00	0.00
	REP	85.85	85.00	0.59	0.34
	REP+DEPREL	86.02	84.97	0.76	0.31
SCORING FUNCTION	NOTSHARED	85.80	84.95	0.00	0.00
	SHARED	85.62	84.80	-0.18	-0.14

Table 3: Analysis of the contribution of various model architecture settings.

to have a 0.00 value; while the other settings are relative increases or decreases from it. See columns Δ **Baseline** in Table 3. We then computed an estimated prediction of the LAS score by adding the Δ values to the Kiperwasser and Goldberg (2016) baseline system. Surprisingly, the resulting scores computed by linear sum of the Δs has a 0.89 correlation with the actual scores in CATiB, and 0.96 correlation with the actual scores in UD. This suggests that the contributions of different settings are largely independent.

4.5 Qualitative Analysis

We studied the improvements in prediction in our best systems compared to the baseline system for both CATiB and UD. We did a careful analysis of the trees, and in terms of the POS tag categories we find that nominals, particles, and punctuation are the biggest beneficiaries in CATiB, whereas the nominals, as in nouns, adjectives, and adverbs, are the biggest winners in UD. This makes sense since the particles in UD are functional case markers that are restricted in their usage. Therefore, we do not see as much of an improvement since they are already almost perfect. It is also worth noting that particles and punctuation are known to be particularly hard cases connected to semantic ambiguity, making their improvement consistent with our expectations. In terms of relations, the biggest improvements we saw were in terms of modifiers. In CATiB, the biggest improvements are in the assignments of MOD, PRD, and TMZ. The constructs of PRD and TMZ in particular are cases that are modeled differently in UD. We can see that illustrated in Figure 1 in the number construct, where it attaches lower in UD rather than higher as it does in CATiB. In UD, the biggest contributors are NMOD, OBJ and IOBJ, but there is generally an improvement across the board in all relations. Finally, when we examined the different frames, we find on average an 11% reduction in the errors, in both CATiB and UD, compared to the baseline in terms of complete incorrect frames. The most changes in positive terms involve modifiers and Idafa (IDF in CATiB, and NMOD:POSS in UD) constructions, where we improve in identifying the full correct frame. However, it is worth noting that the worst cases involve similar constructs, where both systems seem to be too eager to assign modifiers creating much more complex frames than needed.

4.6 Analysis of Switching Frequency

In order to better understand the interplay between transition-based parsing and the inter-dependence of CATiB and UD, we tracked the changes in context when making parsing decisions. We measured the percentage of consecutive decisions that lead to arcs in different dimensions, which we refer to as the *switching frequency*. The Joint Systems (REP) and (REP+DEPREL) attained a switching frequency of 62% and 68% respectively. This indicates that the models are sensitive to inter-dimensional contexts: arcs added in one dimension are helpful to parse the other. We tested this hypothesis further by forcing our system to parse one dimension completely before switching to the other (during training and prediction). This system can be thought of as a pipeline model which has a switching frequency of 0% while still sharing components. We found that for CATiB, the LAS score of the best system decreased on the

DEV set from 86.66 to 86.41 in the pipeline setup CATiB $\rightarrow$ UD, and to 86.37 in UD $\rightarrow$ CATiB. For UD, the score decreased from 85.17 to 85.06 in pipeline CATiB $\rightarrow$ UD and to 85.00 in UD $\rightarrow$ CATiB.

5 Related Work

Arabic Syntactic Dependency Parsing Earlier work on syntactic dependency parsing for Arabic had focused mainly on CATiB representation. Marton et al. (2013) explored the use several morpho-syntactic features in the easy-first framework, while Shahrour et al. (2015; Shahrour et al. (2016) used MaltParser (Nivre et al., 2006). Taji et al. (2017) presented the UD treebank more recently and conducted experiments on CATiB and UD separately in a single-task settings. Multitask systems that have been developed for Arabic were part of efforts to build one multilingual system for all UD dependencies. We present the first effort on multitask joint parsing for multiple Arabic formalisms.

Multitask Dependency Parsing Research towards using multitask deep learning settings to resolve NLP tasks has been an active ongoing subject since the early work of Collobert and Weston (2008) where a single model is trained to perform multiple tasks. Success of such methods is largely due to the effectiveness in deep learning of parameters sharing across multiple models, and learning of joint representations of structures across multiple tasks.

Most similar to our setup is the 2015 SemEval shared task on semantic dependency parsing (Oepen et al., 2015) where three distinct, parallel semantic annotations over the same common texts are available. In this context, several multitask parsers have been proposed. Peng et al. (2017) presented a multitask model with BiLSTMs parameters sharing and low-rank tensor scoring that evaluates the joint fitness of trees across multiple tasks. Their joint inference procedure, however, involves third-order arc interactions which makes it computationally expensive. On the same task, Kurita and Søgaard (2019) describe a model that shares a representation of the complete partial parse forest of one dimension while taking decisions on the other. This is similar in spirit to our sharing of parameters and representations. Furthermore, they show that their transition-based parser effectively learns easy-first strategies with policy gradient based reinforcement learning. This motivates further our choice of parsing framework. In fact, several other multitask systems for semantic dependency parsing have been proposed (Hershcovich et al., 2018; Stanovsky and Dagan, 2018; Peng et al., 2018; Lindemann et al., 2019; Prange et al., 2019) but none of them build on the easy-first framework nor target the Arabic language. Along the lines of multitask easy-first parsers, Constant et al. (2016) introduced a joint model for learning from multiple treebanks simultaneously. They show that syntactic dependency representations and tree-based representations of multiword expressions can help each other. However, they do not use a neural architecture and perform experiments only on English and French.

When no parallel annotations on the same text are available, it has been shown that a single model can be trained to perform multiple tasks as well. Ammar et al. (2016) uses a single model for multilingual parsing trained from multilingual treebanks. Similar multitask models have been developed for cross-domain dependency parsing trained with heterogeneous treebanks (Stymne et al., 2018; Sato et al., 2017). Unlike our approach, decoding for each task is performed independently.

6 Conclusion and Future Work

We presented a dependency parsing model based on a multidimensional neural Easy-First model (Kiperwasser and Goldberg, 2016; Constant et al., 2016). This architecture enables sharing representation at various levels of abstraction, and at different time steps of the parsing process, which makes it possible to communicate information and to learn when sharing information is important across dimensions. We tested this model on two syntactic dependency formalisms for Arabic that vary on the morpho-syntactic (CATiB) to semantic (UD) spectrum. Our experiments showed that this architecture gives a 9.9% error reduction on CATiB, and 6.1% error reduction on UD. Further analysis of this reduction shows that its main contributor is not the sharing of lexical information, as is commonly done in multitask systems, but the sharing of partial dependency trees given as input for arc weight prediction.

Future work will explore further sharing between parsers. In particular, we expect tree encoders could benefit from the additional information (although redundancy with tree sharing in the score function could hurt the system). We also plan to work on model improvements to address the limitation arising from added dimensions. For instance, with the addition of the second dimension, the notion of oracle (Goldberg and Nivre, 2013) used for training becomes more fragile, since the number of correct actions grows, and the order in which to perform them is unknown and can have some long term consequences on the action in the other dimension. It would be interesting to explore how reinforcement learning, where the notion of planning ahead is crucial, could help.

Acknowledgements

This work is partially supported by a public grant overseen by the French National Research Agency (ANR) as part of the program Investissements d'Avenir (reference: ANR-10-LABX-0083). It contributes to the IdEx Université de Paris - ANR-18-IDEX-0001. This work is also partially supported by the ANR as part of the ParSiTi project (reference: ANR-16-CE33-0021).

References

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. One parser, many languages. *CoRR*, abs/1602.01595.

Rich Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on International Conference on Machine Learning*, ICML'93, page 41–48, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 160–167, New York, NY, USA. Association for Computing Machinery.

Matthieu Constant, Joseph Le Roux, and Nadi Tomeh. 2016. Deep lexical segmentation and syntactic parsing in the easy-first dependency framework. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1095–1101, San Diego, California, June. Association for Computational Linguistics.

Mona Diab, Nizar Habash, Owen Rambow, and Ryan Roth. 2013. LDC Arabic treebanks and associated corpora: Data divisions manual. *arXiv preprint arXiv:1309.5652*.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

Yoav Goldberg and Joakim Nivre. 2013. Training deterministic parsers with non-deterministic oracles. *Transactions of the Association for Computational Linguistics*, 1:403–414.

A. Graves, S. Fernández, and J. Schmidhuber. 2005. Bidirectional lstm networks for improved phoneme classification and recognition. In W. Duch, J. Kacprzyk, E. Oja, and S. Zadrozny, editors, *Artificial Neural Networks: Biological Inspirations - ICANN 2005, LNCS 3697*, pages 799–804. Springer-Verlag Berlin Heidelberg.

Nizar Habash and Ryan Roth. 2009. CATiB: The Columbia Arabic Treebank. In *Proceedings of the Joint Conference of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 221–224, Suntec, Singapore.

Nizar Habash. 2010. *Introduction to Arabic natural language processing*, volume 3. Morgan & Claypool Publishers.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1923–1933, Copenhagen, Denmark, September. Association for Computational Linguistics.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2018. Multitask parsing across semantic representations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 373–385, Melbourne, Australia, July. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Easy-first dependency parsing with hierarchical tree lstms. *Transactions of the Association for Computational Linguistics*, 4:445–461.

Shuhei Kurita and Anders Søgaard. 2019. Multi-task semantic dependency parsing with policy gradient for learning easy-first strategies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2420–2430, Florence, Italy, July. Association for Computational Linguistics.

Matthias Lindemann, Jonas Groschwitz, and Alexander Koller. 2019. Compositional semantic parsing across graphbanks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4576–4585, Florence, Italy, July. Association for Computational Linguistics.

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus. In *Proceedings of the International Conference on Arabic Language Resources and Tools*, pages 102–109, Cairo, Egypt.

Yuval Marton, Nizar Habash, and Owen Rambow. 2013. Dependency parsing of modern standard Arabic with lexical and inflectional features. *Computational Linguistics*, 39(1):161–194.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116, Berlin, Germany, August. Association for Computational Linguistics.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy, May. European Language Resources Association (ELRA).

Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Eckhard Bick, Cristina Bosco, Gosse Bouma, Sam Bowman, Marie Candito, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Fabricio Chalub, Jinho Choi, Çağrı Çöltekin, Miriam Connor, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Nizar Habash, Jan Hajič, Linh Hà Mỹ, Dag Haug, Barbora Hladká, Petter Hohle, Radu Ion, Elena Irimia, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Hiroshi Kanayama, Jenna Kanerva, Natalia Kotsyba, Simon Krek, Veronika Laippala, Phuong Lê Hồng, Alessandro Lenci, Nikola Ljubešić, Olga Lyashevskaya, Teresa Lynn, Aibek Makazhanov, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Shunsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Nina Mustafina, Kaili Müürisep, Luong Nguyễn Thị, Huyền Nguyễn Thị Minh, Vitaly Nikolaev, Hanna Nurmi, Stina Ojala, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Lauma Pretkalniņa, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Loganathan Ramasamy, Livy Real, Laura Rituma, Rudolf Rosa, Shadi Saleh, Manuela Sanguinetti, Baiba Saulīte, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Lena Shakurova, Mo Shen, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Takaaki Tanaka, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Larraitz Uria, Gertjan van Noord, Viktor Varga, Veronika Vincze, Jonathan North Washington, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, and Hanzhi Zhu. 2017. Universal dependencies 2.0.

Joakim Nivre. 2016. Universal dependencies: Dubious linguistics and crappy parsing? COLING.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, and Zdeňka Urešová. 2015. SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 915–926, Denver, Colorado, June. Association for Computational Linguistics.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep multitask learning for semantic dependency parsing. *CoRR*, abs/1704.06855.

Hao Peng, Sam Thomson, Swabha Swayamdipta, and Noah A. Smith. 2018. Learning joint semantic parsers from disjoint data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1492–1502, New Orleans, Louisiana, June. Association for Computational Linguistics.

Jakob Prange, Nathan Schneider, and Omri Abend. 2019. Made for each other: Broad-coverage semantic structures meet preposition supersenses. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 174–185, Hong Kong, China, November. Association for Computational Linguistics.

Motoki Sato, Hitoshi Manabe, Hiroshi Noji, and Yuji Matsumoto. 2017. Adversarial training for cross-domain universal dependency parsing. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 71–79, Vancouver, Canada, August. Association for Computational Linguistics.

Anas Shahrour, Salam Khalifa, and Nizar Habash. 2015. Improving Arabic diacritization through syntactic analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1309–1315, Lisbon, Portugal.

Anas Shahrour, Salam Khalifa, Dima Taji, and Nizar Habash. 2016. CamelParser: A system for Arabic syntactic analysis and morphological disambiguation. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 228–232.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 231–235, Berlin, Germany, August. Association for Computational Linguistics.

Gabriel Stanovsky and Ido Dagan. 2018. Semantics as a foreign language. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2412–2421, Brussels, Belgium, October-November. Association for Computational Linguistics.

Sara Stymne, Miryam de Lhoneux, Aaron Smith, and Joakim Nivre. 2018. Parser training with heterogeneous treebanks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 619–625, Melbourne, Australia, July. Association for Computational Linguistics.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China, July. Association for Computational Linguistics.

Dima Taji, Nizar Habash, and Daniel Zeman. 2017. Universal dependencies for Arabic. In *Proceedings of the Workshop for Arabic Natural Language Processing (WANLP)*, Valencia, Spain.