A Cross-Task Flexible Transition Model for Arabic Tokenization, Affix Detection, Affix Labeling, POS Tagging, and Dependency Parsing

Stephen Tratz
Army Research Laboratory
Adelphi Laboratory Center
2800 Powder Mill Road
Adelphi, MD 20783
stephen.c.tratz.civ@mail.mil

Abstract

This paper describes cross-task flexible transition models (CTF-TMs) and demonstrates their effectiveness for Arabic natural language processing (NLP). NLP pipelines often suffer from error propagation, as errors committed in lower-level tasks cascade through the remainder of the processing pipeline. By allowing a flexible order of operations across and within multiple NLP tasks, a CTF-TM can mitigate both cross-task and within-task error propagation. Our Arabic CTF-TM models tokenization, affix detection, affix labeling, part-of-speech tagging, and dependency parsing, achieving state-of-the-art results. We present the details of our general framework, our Arabic CTF-TM, and the setup and results of our experiments.

1 Introduction

Natural Language Processing (NLP) systems often consist of a series of NLP components, each trained to perform a specific task such as parsing. These pipelines tend to suffer from error propagation—errors introduced by early components cascade through the remainder of the pipeline causing subsequent components to commit additional errors. Partial solutions from higher-level tasks (e.g., parsing) can aid in resolving the difficult decisions that must be made in solving lower-level tasks, as with part-of-speech tagging the classic "garden path" sentence example "The horse raced past the barn fell." To this end, this paper presents *cross-task flexible transition models* (CTF-TMs), which model multiple tasks and solve these tasks in a more flexible order than pipeline approaches. We implement and experiment with a CTF-TM for Arabic[1] language processing and report experimental results for it on Arabic tokenization (i.e., clitic separation), affix detection, affix labeling, part-of-speech tagging, and dependency parsing.

In addition to error propagation between modules within a parsing pipeline, errors may propagate within the parsing process itself due to the fixed order of operations of the parser. This is common for standard *transition-based* dependency parsing models (McDonald and Nivre, 2007), such as shift-reduce parsers, which incrementally construct a parse by processing the input in a fixed left-to-right or right-to-left fashion. However, using a transition model that allows a more flexible order of operations, such as Goldberg and Elhadad's (2010) parser, allows difficult decisions to be postponed until later, when more of the solution has been constructed. CTF-TMs extend this approach by modeling multiple tasks and providing this flexibility across tasks so that no one task needs to be complete before another can be partially solved.

As a morphologically rich language, Arabic requires a significant number of processing steps. Arabic uses a variety of affixes to inflect for case, gender, number (including dual), and mood, has clitics that attach to other words, permits both VSO and SVO constructions, and rarely includes short vowels in written form. The presence of clitics and the absence of written short vowels are particularly significant sources of ambiguity. As Tsarfaty (2006) argues for Modern Hebrew, a Semitic language that shares these characteristics, we contend that mor-

[1] This paper focuses on Modern Standard Arabic rather than any of the dialects.

Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically Rich Languages, pages 34–45,
Seattle, Washington, USA, 18 October 2013.

phological analysis and parsing should be done in a unified framework, such as a CTF-TM, rather than by separate components.

In this paper, we describe CTF-TMs, which can be used for a wide variety of NLP tasks, and present our Arabic CTF-TM for Arabic tokenization, affix detection, affix labeling, part-of-speech tagging, and dependency parsing as well as the results obtained in applying it to our dependency conversion of the Penn Arabic Treebank (ATB) (Maamouri et al., 2004; Maamouri and Bies, 2004). We find that our Arabic CTF-TM for tokenization, affix detection, affix labeling, POS tagging, and parsing achieves slightly better results than a similar CTF-TM that performs all the tasks except parsing. The CTF-TM that supports parsing appears to be more accurate at distinguishing between passive and active verbs as well as between nouns and adjectives—cases where the context is crucial for proper interpretation due to Arabic's ambiguities. Our system achieves tokenization accuracy similar to Kulick's (2011) state-of-the-art system for a standard split of the ATB part 3, and, in our experiments using ATB parts 1–3, our system achieves the highest labeled attachment, unlabeled attachment, and clitic separation figures (including pronomial clitics) for Arabic yet reported (although no other work can be compared directly).

2 Relevant Arabic Linguistics

Arabic has rich morphology, with a wide array of affixes and clitics and inflecting for case, number, gender, and, occasionally, mood. Coordinating conjunctions, pronouns, and most true prepositions, along with some other particles and the definite article, usually occur as clitics in Arabic. Thus, a space-delimited[2] sequence of Arabic characters may consist of multiple words, and identifying the boundaries between these must be done in order to produce syntactic parses. These boundaries can't be detected perfectly using simple deterministic rules. Significantly, short vowels, which are expressed using diacritics, are not typically written in Arabic, resulting in pervasive ambiguity. For example, active and passive forms of verbs vary only in their diacritics, and nouns and adjectives are both derived from Arabic roots using the same templates and, thus, look similar. A single Arabic token may permit a variety of different analyses, as the example in Table 1 illustrates.

والى	wAlY	'ruler'
وإلى+ي	w+AlY+y	'and to me'
وأل+ي	w+<ly	'and I follow'
وآل+ي	w+|l+y	'and my clan'
وآلي	w+|ly	'and automatic'

Table 1: Possible interpretations for the text *wAlY* (Habash and Rambow, 2005).

3 CTF-TM Framework

Error propagation is not simply a problem that occurs between components in a pipeline but one that often occurs within a single component's processing. Since transition systems can use the partially built solution for feature generation, incorrect actions taken early on result not only in an invalid final solution, but the invalid partial solution may dissuade the system from making correct decisions with respect to other parts of the solution. If a transition system can postpone decisions it is not confident of until later, the partial solution created by performing other actions may provide more or better information that enables the system to properly resolve more difficult decisions. This "easy-first" strategy is adopted by Goldberg and Elhadad's (2010) parsing system, which starts with an ordered list of unattached words and, in each iteration, creates a new arc between any of the adjacent pairs of words in the list and removes the daughter node (word) from the list.

This strategy is much more *flexible* than shift-reduce style parsing because the system has more options available to it at any one step for building up the solution. However, simply having flexibility within a single component does not reduce error propagation to or from other components in a pipeline and, to mitigate the potential for this, one may use a *cross-task flexible transition model* (CTF-TM) that does not have to wait for lower level tasks to be 100% complete before starting work on higher level tasks.

[2]Technically, space-and-punctuation-deliminated.

McDonald and Nivre (2007) define a *transition system* as follows:

1. a set C of *parse configurations*, each of which defines a (partially built) dependency graph G
2. a set T of transitions, each a function $t : C \rightarrow C$
3. for every sentence $x = w_0, w_1, ..., w_n$
   (a) a unique *initial* configuration c_x
   (b) a set C_x of *terminal* configurations

These systems start at the initial configuration and use a scoring function $s : C \times T \rightarrow \mathbb{R}$ to repeatedly select and follow the locally optimal transition, stopping when a terminal configuration is reached.

We make a few changes to McDonald and Nivre's transition system definition in order to explain our framework. First, to support modeling of multiple tasks, instead of referring to parse configurations, we simply use the term *configuration*, defining it to represent a partially built *solution* rather than a dependency graph. Second, we specify that there exists a routine for enumerating a set of *anchors* for any given configuration. Anchors are an organizational concept for dealing with arbitrary data structures; each anchor acts as a hook into some portion of the configuration that may be changed. Finally, there exist routines for enumerating legal *actions* that can be performed in relation to any anchor and, for training, a routine for verifying that performing a given action will lead to a configuration consistent with the final solution. The performance of an action constitutes a transition between configurations.[3] It is quite straightfoward to adapt Goldberg and Elhadad's (2010) parsing approach to any configuration that is indexable by anchors, and in so doing we are able to create cross-task flexible transition models.

[3]For example, in a fixed order, one-word-at-a-time POS tagging system, there would be only one anchor—the word currently being labeled—but, for a one-at-a-time POS tagger capable of tagging words in any order, the anchor set would contain the entire list of still-unlabeled words. The POS labeling actions for the anchors in each of these cases constitute transitions to new configurations.

4 Our Arabic CTF-TM

4.1 Tasks

Our Arabic CTF-TM system performs the following tasks: split a series of space-delimited Arabic tokens into words (tokenization), identify the bounds of affixes within the words (affix detection), label the affixes (affix labeling), label the words with their parts of speech (POS tagging), and construct a labeled dependency tree (dependency parsing). Tokenization, part-of-speech tagging, and dependency parsing are frequent topics in NLP literature. Affix identification and labeling are parts of morphological analysis that are sometimes completely ignored or are performed using an external morphological analyzer. Identifying affixes and labeling them can help the overall system contend with lexical sparsity issues as well as utilize the information encoded by the affixes (e.g., person).

4.2 Anchors and Actions

The configurations that the system deals with have anchors of two types, token anchors and affix anchors. The initial configuration consists of an ordered list of neighboring token anchors (neighborhood), each of which corresponds to one of the original space-delimited tokens. As processing continues, new token anchors may be created by splitting off clitics, new affix anchors may be created by identifying substrings of tokens as affixes, and token anchors will be removed from the ordered list to become daughter nodes of their neighbors, attached via labeled dependency arcs. The complete list of actions that can be performed on the anchors, which, as described earlier, constitute the transitions between configurations, are as follows:

Tokenization

1. Separate a proclitic of length *l* from a token anchor, creating a new token anchor for the clitic and reducing the width of the original token
2. Separate an enclitic of length *l* from a token anchor, creating a new token anchor for the clitic and reducing the width of the original token

Affix Detection

3. Create an affix (prefix) anchor from the first *l* characters of a token anchor that are not labeled as part of an affix (If the affix is the definite determiner *Al*, which we treat as an affix for consistency with the ATB's tokenization scheme, it is automatically labeled as DET and removed from further processing for the sake of efficiency.)

4. Create an affix (suffix) anchor from the last l characters of a token anchor that are not labeled as part of an affix

POS and affix labeling

5. Assign a label l to the anchor (Affixes are automatically removed from further processing after labeling)

Dependency parsing

6. Create a dependency arc with label d between a token anchor and the preceding unattached neighbor token anchor and remove the attached anchor from the neighborhood
7. Create a dependency arc with label d between a token anchor and the following unattached neighbor token anchor with label l and remove the attached anchor from the neighborhood
8. Swap the position of two neighboring token anchors (this adds Nivre-style (2009) non-projectivity support as described by Tratz and Hovy (2011))

General

9. Mark an anchor as fully processed and remove it from further processing

The dependency labels, POS labels, clitic lengths, and affix lengths used to define the actions are all collected automatically from the training data. [4]

The actions are subject to the following constraints/preconditions:

1. Labeling is only valid if the anchor has not been labeled
2. Tokens may only be labeled with token labels, prefixes with prefix labels, and suffixes with suffix labels (as determined by the training data)
3. Affix strings observed in the training data may not be labeled with any label not used with them in the training data
4. Token anchors may not be assigned labels that do not co-occur with the labels of any already-labeled affixes and vice versa
5. A prefix creation action may only be applied to a token anchor that doesn't yet have a prefix
6. Proclitics may not be created and detached if the token already has a prefix, and enclitics are similarly restricted by the presence of a suffix
7. Clitics may not be detached from a token that has already been attached to another token via a dependency arc
8. A dependency arc with label x may not be created between token anchors T_1 and T_2 if 1) one or both are labeled and 2) no arc between similarly POS tagged anchors exists in the training data
9. Swap actions may not undo previous swaps
10. Marking a token anchor as fully processed may only occur if it has already been labeled, and it must either be 1) the last unattached token or 2) already attached

[4] Training examples with clitics that are invalid (i.e., too long) are discarded at the beginning of training.

4.3 Scoring Function

For our scoring function, like Goldberg and Elhadad, we use the structured perceptron algorithm (Collins, 2002) with parameter averaging. This has previously been shown to produce strong results when modeling multiple NLP tasks (Zhang and Clark, 2008).

4.4 Features

For a given anchor[5], the system extracts features from the partially built solution (e.g., the text, affixes, POS tags, and syntactic dependencies of the anchor and nearby anchors). The same feature templates are used for all action types except the affix labeling actions—affix labeling is applied to affix anchors instead of word-level anchors, and, since all templates are defined relative to an anchor, the templates must be different. The system uses no external resources (e.g., lexicons, morphological analyzers). We leave out a more exhaustive listing and description of the features due to space limitations[6], the fact that the focus of this paper is not on the value of any particular feature template but rather on our overall approach and experimental results, and because we plan to release our code, which will be more helpful for reproducibility efforts.

4.5 Data Preparation

For our experiments, we use the original written form of the data from the latest versions of the first three parts of the Penn Arabic Treebank (ATB) (Maamouri et al., 2004; Maamouri and Bies, 2004) as well as the new broadcast news collection (Maamouri et al., 2012).[7] We convert the constituent trees into dependency trees and adjust the part-of-speech tags.

[5] 'A given action' may be more correct technically, but our implementation is set up to share the same set of string-based features across all actions associated with a given anchor.

[6] Simply listing the feature templates in a normal font size with minimal (insufficient) explanation would require well over a page. The set of feature templates is based upon the templates used by Tratz and Hovy's (2011) English parser, which are given in Tratz's (2011) thesis.

[7] We use version 4.1 of ATB part 1, 3.1 of part 2, 3.2 of part 3, and 1.0 of the broadcast news transcriptions.

4.5.1 Dependency Conversion

The two main Modern Standard Arabic dependency treebanks currently available are the Columbia Arabic Treebank (CATiB) (Habash and Roth, 2009) and the Prague Arabic Dependency Treebank (PADT) (Hajič et al., 2004). CATiB has over 228,000 manually annotated words as well as an automatic ATB conversion. It uses only 8 dependency relations (subject, object, predicate, topic, idafa, tamyiz, modifier, and flat) and 6 part-of-speech tags, and it has not yet been publicly released by the LDC. The PADT, which was used in the CoNLL 2006 and 2007 shared tasks (Buchholz and Marsi, 2006; Nivre et al., 2007), is much smaller, with only about 148,000 annotated tokens. Since we want a large annotated corpus with fine-grained labels, we create our own ATB conversion.

4.5.2 Transformations

In addition to converting the ATB's constituent parses to dependency trees, we make a handful of other changes. Following Green and Manning (2010) and others, sentences headed by *X* nodes are deleted because the treebank annotators considered them unbracketable or somehow erroneous. Following Rambow et al. (2005), Treebank sentences headed by *TOP* elements containing multiple *S* daughters are split into separate sentences.[8] Additionally, if the dependency converter concludes that an *S* node without treebank functional tags is dependent upon another *S* node and is separated from it via sentence-final punctuation (e.g., an exclamation point), these *S* nodes are separated into distinct sentences as well. For the broadcast news data, we remove all subtrees headed by *EDITED* tags to make it more closely resemble newswire text.[9]

Since we adhere to the tokenization scheme used by the ATB, and we do not split off the determiner *Al* as its own tree token. Instead, we treat it as a prefix.

The words referred to as *inna and her sisters* are annotated using two different part-of-speech categories and syntactic structures in the ATB. In our conversion, both ATB structures are converted to the same dependency structure headed by the INNA word, similar to CATiB (Habash and Roth, 2009).

We treat the focus particle *AmmA* like a preposition in our dependency structure, following CATiB.

4.5.3 Dependency Label Scheme

Our dependency scheme consists of a total of 35 labels. Many of these are similar to those of Stanford's basic dependency scheme for English (de Marneffe and Manning, 2008), although they are somewhat closer to a similar scheme used by (Tratz and Hovy, 2011). The list of relations is presented in Table 2.

Most of the relations are self-explanatory or correspond to similar labels in either Tratz and Hovy's (2011) scheme for English or CATiB's (Habash and Roth, 2009) scheme for Arabic. A few are new or significantly different from their similarly named counterparts in other schemes and are described in greater detail below.

- *adjnom* — connects the head of an NP to that of a sister NP (occurs with apposition and preposition-like nouns)
- *advcl* — connects verbal nouns to their syntactic governor in what resemble English's adverbial participle clauses
- *advnp* — connects NPs with treebank adverbial function tags (e.g., -LOC, -TMP, -DIR), which are often headed by preposition-like nouns, to what they modify
- *fidafa* — for false idafa (idafa-like structures that are headed by adjectives instead of nouns)
- *kccmp* — connects a clausal complement that is part of a past progressive or habitual construction to the head verb *kana*
- *lakinna* — similar to *cc* but used with the sister of inna *lakinna* instead of coordinating conjunctions
- *part* — particle modifier; connects particles (other than FOCUS_PART) to their governors
- *rcmod* — connects a bare relative clause to its head
- *reladv* — connects an adverbial WH- clause to its governor
- *relmod* — connects the head of a WH- node to the relativized word
- *ripcmp* — connects a clause to the relative or interrogative pronoun that heads it

4.5.4 Part-of-Speech Tag Scheme

The Penn Arabic Treebank uses complex part of speech tags for the entire tree token such as DET+NOUN+NSUFF_FEM_SG+CASE_DEF_GEN. Across the treebank data used in our experiments, there are a total of 579 such tags, which are composed of 179 different parts separated by plus signs. Each part corresponds to a substring of the

[8] The ATB often has multiple sentences, or even entire paragraphs, annotated under a single *TOP* element.

[9] The *EDITED* tag "is used to show the repetition and restarting of constituents that are repaired by subsequent speech" (Maamouri et al., 2012).

adjnom	adjunct nominal	intj	interjection	prep	preposition modifier
advcl	adverbial clause	iobj	indirect object	punct	punctuation modifier
advmod	adverbial modifier	idafa	idafa	*rcmod*	(bare) relative clause modifier
advnp	adverbial noun phrase	**fidafa**	false idafa	**reladv**	relative pronoun adverbial
cc	coordinating conjunction	flat	flat structure	**relmod**	relative pronoun modifier
ccinit	initial coordinating conjunction	**kccmp**	kana clausal complement	**ripcmp**	relative/interrogative pronoun complement
ccomp	clausal complement	**lakinna**	*see text*	sc	subordinating conjunction modifier
combo	combination term	neg	negation	subj	subject
conj	conjunction	obj	object	tmz	tamyiz
cop	copula complement	objcomp	object complement	tpc	topicalized element
dep	unspecified dependency	**part**	particle modifier	voc	vocative
det	determiner	pcomp	preposition complement		

Table 2: Syntactic dependency scheme used in this work. Labels that aren't self-explanatory or similar to the labels used by Tratz and Hovy (2011) for English or CATiB for Arabic (Habash and Roth, 2009) are in bold (for completely new relations) or italics (for similarly named but semantically different relations)

vowelized version of the word.[10] Due at least in part to the enormity of this label set, simpler schemes are often preferred, such as the "Bies" labels (Bikel, 2004; Kulick et al., 2006), Diab's (2007) labels, Kulick's (2011) labels, and CATiB's labels (Habash and Roth, 2009). Marton et al. (2010) find that using simpler schemes allow them to get better parsing results when using predicted POS tags due to the relatively poor performance of taggers trained using the full ATB scheme.

The part-of-speech tag scheme we use is quite similar to that of the original ATB but has several simplifications. These changes are listed below.

1. Possessive and direct object pronoun clitics are all given the same label (PRON_OPP) (50 fewer tags; mapping back to the originals is trival in almost all cases)
2. .VN forms of NOUN and ADJ are merged with their respective more generic categories
3. Interrogative and relative adverbial and pronoun labels are merged together into RI_ADV and RI_PRON
4. Noun suffix labels (e.g., NSUFF_MASC_PL_GEN, NSUFF_MASC_PL_ACC) with genitive or accusative case distinctions are merged because there is no distinction in unvowelized form
5. Labels for dual masculine noun suffixes are merged with their plural counterparts (no distinction in the unvowelized forms)
6. Demonstrative pronoun labels are collapsed to DEM_PRON (person and number information is easily recovered)
7. The words called *inna and her sisters* are labeled INNA instead of PSEUDO_VERB or SUB_CONJ

[10]Since we use the original written form of the data and the internal segmentation of the words are only provided for the vowelized versions, we project the segmentation into the original written forms, discarding any parts that weren't actually written (e.g., case labels associated with unwritten diacritics).

Since our system splits off clitics and identifies the affixes, the tagging is performed at the individual morpheme level instead of producing a single all-encompassing tag for the entire token.

Some of the part-of-speech tags (mostly instances of DIALECT, TYPO, TRANSERR, and NOT_IN_LEXICON tags) are automatically corrected/improved during the dependency conversion based upon the original constituent parse.

4.6 Filtering

Sentences containing invalid clitics are not used in training both because they are erroneous and because including them would require allowing the system to perform actions that should not occur (i.e., splitting off a clitic of length 8); similarly, training examples with more than 20% of their tokens tagged as DIALECT, TRANSERR, LATIN, PARTIAL, GRAMMAR_PROBLEM, and/or TYPO are ignored on the assumption that including them would harm the model. This filtering process is not applied in testing.

4.7 Data Split

We train and test models using three different splits of the data. The first split is based upon the split used by Zitouni et al. (2006) in their diacritization work and is the same as that used by Marton et al. (2013) in their parsing work and by Kulick (2011) in his tokenization and part-of-speech tagging work, in order to facilitate better comparison. However, Marton et al. use the CATiB conversion of a slightly earlier version of the data (3.1, not 3.2), and, thus, the results are not directly comparable. This split places

Part	Use	Files	Sent	Toks	Tree Toks	Affixes
1	train	514	4090	101629	116892	49057
	dev	110	909	22932	26261	11074
	test	110	823	20825	24127	10032
2	train	351	3011	102795	120605	56273
	dev	75	559	20869	24619	11245
	test	75	630	20518	24078	11078
3	train	509	11350	287945	341033	145621
	dev	45	1029	26347	31200	13828
	test	45	992	25299	29938	12220
BN	train	68	5504	82388	98040	48190
	dev	26	1801	29873	35676	17890
	test	26	2082	34361	41192	20366

Table 3: Counts of the number of files, sentences (Sent), original space-delimited tokens (Tok), ATB tree tokens (Tree Toks), and affixes in the experimental data.

the first (in name and chronological order) 85% of the documents in ATB part 3 in training, the next 7.5% in development, and the final 7.5% in test.

In the second split, we use data from the first three parts of the ATB, each of which consists of documents coming from a different newswire source. Parts 1 and 2 are split 70%/15%/15% training/dev/test, and we reuse the split of part 3 just mentioned. Under this setup, we train two different CTF-TMs, one that performs all of the tasks and one that performs all of the tasks except parsing. This enables us to test whether modeling parsing task improves performance on the lower level tasks.

In the final split, we use the splits for parts 1–3 plus the data in LDC's annotated broadcast news transcripts (Maamouri et al., 2012). Unlike parts 1–3, the broadcast news data are drawn from a variety of sources. Files from sources with three or more files are split across training, development, and test, with the latest documents being placed in test. [11] This experiment illustrates how the system performs when additional, out-of-domain data are included.

Statistics for the data are given in Table 3.

4.8 Evaluation Measures

Dependency parsing quality is measured in terms of labeled and unlabeled attachment scores (LAS and UAS), which indicate the percentage of words attached to their correct parent and, in the case of LAS, whose attachment is labeled with the correct dependency. Since a given space-delimited token may not be tokenized into words correctly, the dependency arcs are only counted as correct if they occur between the correct words (spans of character indices). We measure part-of-speech tagging in terms of F-score (F1) and require that the tree token have the correct bounds (was tokenized correctly) and have the correct label.

Normally, we would choose LAS on the development set as the measure for determining the version of the model to keep for testing because it measures performance on the highest-level task (labeled dependency parsing). However, since one of the CTF-TMs does not perform parsing, we instead use POS tagging F1. In general, we observe that the scores are highly correlated, making the point moot. For the ATB part 3 experiment, POS tagging F1 peaks on iteration 437.[12] For the second experiment, POS tagging F1 peaks at iteration 301 for the CTF-TM with parsing and iteration 278 for the one without. For the third experiment, the highest score occurs on iteration 431.

4.9 Results and Discussion

The results for the various experimental setups are presented in Table 4.

ATB 3 Experiment When using the same split of ATB part 3 as Kulick (2011) and Marton et al. (2013), the system correctly tokenizes 99.3% of the space-delimited tokens, similar to Kulick's (2011) accuracy (99.3%) and slightly higher than the 99.0% figure Kulick calculates for MADA. Though these results are obtained using our dependency conversion of the ATB rather than the original, we use the same tokenization scheme. The POS labeling F1 score of 95.8 can't be compared well with any other work due to differences in tag schemes, which vary greatly, as well as use of gold tokenization and other differences. Our system obtains 84.9 UAS and 82.0 LAS, which are higher than Marton et al.'s best results of 84.0 UAS and 81.0 LAS, but they were using a different conversion (CATiB) of a different version of the data (3.1, not 3.2) as well as gold tokenization, so the results are not directly comparable.

Framework Internal Experiment The CTF-TM

[11] We will make the exact list of files used in the training, development, and test sets available.

[12] We run 500 iterations for each experiment, which can take as long as a week using a quad-core machine. However, little improvement is seen after the first 100 iterations.

Train	Eval Data	Tok Acc	POS F1	Affix Bounds F1	Affix Label F1	UAS	LAS
3	3 Dev	99.5	96.6	98.7	98.4	86.3	83.8
3	3 Test	99.3	95.8	98.4	97.9	84.9	82.0
1,2,3	1,2,3 Dev	99.6	97.1	99.1	98.9	88.3	86.0
1,2,3	1,2,3 Test	99.6	96.8	99.0	98.7	87.4	84.8
1,2,3,BN	1,2,3 Dev	99.6	97.1	99.1	98.9	88.5	86.2
1,2,3,BN	1,2,3 Test	99.6	96.8	99.0	98.8	87.5	85.0
1,2,3,BN	1,2,3,BN Dev	99.5	96.0	98.8	98.5	87.4	84.6
1,2,3,BN	1,2,3,BN Test	99.3	95.7	98.7	98.4	86.6	83.8
Without Parsing							
1,2,3	1,2,3 Dev	99.6	96.9	99.1	98.9	NA	NA
1,2,3	1,2,3 Test	99.5	96.5	98.9	98.6	NA	NA

Table 4: Results for the various experiments (Exp) for both the development and test portions of the data, including per-token clitic separation (tokenization) accuracy, part-of-speech tagging F1, affix boundary detection F1, affix labeling F1, and both unlabeled and labeled attachment scores.

that does parsing and the CTF-TM that doesn't achieve similar overall results for the different tasks (other than parsing, of course). However, when looking deeper at the individual POS tagging mistakes that one system made more often by one system than the other, (see Tables 5 and 6), we observe that the parsing CTF-TM does a better job with labeling some parts-of-speech. For instance, the non-parsing system mismarks passive verbs as active more than 29% more often than the other. In Arabic, passive and active forms of verbs are only distinguished by their short vowels, which are typically unwritten, and, thus, the context is of particular importance in distinguishing between the two. The non-parsing system also has more trouble with the distinction between nouns and adjectives, which is likely because adjectives are derived using the same templatic structures as nouns (Attia et al., 2010) and, thus, context is, once again, of great importance.

Broadcast News Experiment The scores obtained in the experiment with the broadcast news data are slightly lower than in the second experiment. However, this appears to be because the broadcast news portions of the development and test sections are more difficult to parse than the remainder. If we apply the model to the development and test sections of parts 1, 2, and 3, we observe that the results, which are given in Table 4, are higher than those of the model trained without the broadcast news data.

Gold	Prediction	Errors		Diff
		-parse	+parse	
NOUN	ADJ	297	238	-59
ADJ	NOUN	328	298	-30
VB_IV_PASS	VB_IV	109	80	-29
VB_PV_PASS	VB_PV	86	68	-18
VB_PV	NOUN	104	88	-16
VB_IV	VB_PV	12	22	+10
INNA	SUB_CONJ	9	2	-7
VB_PV	VB_IV	19	13	-6
NOUN	NOUN_PROP	140	134	-6
ADJ	NOUN_PROP	32	27	-5

Table 5: Top 10 POS mistakes made more often by either the CTF-TM with parsing or the CTF-TM without on the ATB part 1, 2, and 3 development set.

Tag	#Gold	Tag	#Gold
NOUN	26195	INNA	1456
ADJ	7491	SUB_CONJ	641
NOUN_PROP	5913	VB_PV_PASS	231
VB_PV	3478	VB_IV_PASS	207
VB_IV	2682		

Table 6: Counts for the POS tags mentioned in Table 5.

5 Related Work

5.1 Semitic Language Parsing

Much of the Arabic parsing research to date uses the pipeline approach, either running a tokenizer prior to parsing or simply assuming the existence of gold tokenization (Bikel, 2004; Buchholz and Marsi, 2006; Kulick et al., 2006; Nivre et al., 2007; Marton et al., 2010; Marton et al., 2011; Marton et al., 2013). Of course, using gold tokenization results in optimistic

evaluation figures.[13]

Other methods exist however. For example, to parse Modern Hebrew, Cohen and Smith (2007) combine a morphological model with a syntactic model using a product of experts. Another alternative is lattice parsing, which can be used to jointly model both tokenization and parsing (Chappelier et al., 1999). Curiously, while researchers of Modern Hebrew parsing find lattice parsers outperforming their pipeline systems (Goldberg and Tsarfaty, 2008; Goldberg and Elhadad, 2011; Goldberg and Elhadad, 2013), Green and Manning (2010) obtain the opposite result in their Arabic parsing experiments, with the lattice parser underperforming the pipeline system by over 3 points (76.01 F1 vs 79.17 F1). Why lattice parsing may help in some cases but not others is not clear.

Some Arabic parsing work focuses on the usefulness of various features and part-of-speech tagsets. Marton et al. (2013) examine various morphological features and part-of-speech tagsets, employing MADA (Habash and Rambow, 2005; Habash et al., 2009) to predict form-based morphological features and an in-house system (Alkuhlani and Habash, 2012) to predict functional morphological features. Dehdari et al. (2011) investigate the best set of features for Arabic constituent parsing and try several approaches for selecting an optimal feature set, finding that the best-first with backtracking algorithm is the most effective in their experiments.

5.2 Other Languages

There has been a flurry of recent research involving the joint modeling of dependency parsing and lower-level tasks[14] for a variety of languages, with most of the attention focused on Chinese. While lacking Arabic's morphological richness, Chinese has its own challenges, such as word segmentation and part-of-speech ambiguities, which have led researchers to develop new unified approaches for processing it. Qian and Liu (2012) train independent models for word segmentation, POS tagging, and parsing but then incorporate them together during decoding. Li et al. (2011), Li and Zhou (2012), Hatori et al. (2011), and Ma et al. (2012) present systems that jointly model Chinese POS tagging and dependency parsing. Li et al. (2011) use a dynamic programming approach similar to Koo and Collins (2010), Li and Zhou (2012) present a shift-reduce style system that uses structured perceptron and beam search, Hatori et al. (2011) implement a shift-reduce style algorithm that utilizes dynamic programming and beam search in the manner of Huang and Sagae (2010), and Ma et al. (2012) extend Goldberg and Elhadad's (2010) easy-first approach to support both dependency parsing and POS tagging and is thus similar to our work. Hatori et al. (2012) extend their previous system to tackle word segmentation, and Ma et al. (2013) build upon earlier work by implementing beam search to get better results. Li and Zhou (2012) side step some of the issues of Chinese word segmentation by parsing structures of words, phrases, and sentences in a unified framework using a structured perceptron and beam search.

Some researchers focus their work on other languages. Lee et al. (2011) present a graphical model for morphological disambiguation and dependency parsing that they apply to Latin, Ancient Greek, Hungarian, and Czech. Bohnet and Nivre (2012) present a shift-reduce style system similar to Li and Zhou's (2012) system that jointly models POS tagging and labeled dependency parsing, achieving state-of-the-art accuracy on Czech, German, Chinese, and English.

6 Conclusion

In this paper, we described cross-task flexible transition models (CTF-TMs) and demonstrated their viability for Arabic tokenization, affix detection, affix labeling, part-of-speech labeling, and dependency parsing, obtaining very strong results in each tasks. We plan to release our software in the near future, including the software for converting the ATB to dependency parses, and would like to release our dependency conversion of the Penn Arabic Treebank via the LDC.

[13] Green and Manning (2010) find that using automatic tokenization provided by MADA (Habash et al., 2009) instead of gold tokenization results in a 1.92% F score drop in their constituent parsing work.

[14] Systems that jointly model POS tagging and *constituent* parsing have existed for some time.

7 Future Work

In the future, we plan to integrate beam search into the training and decoding. We want to add support for the recovery of diacritics, roots, and derivation templates, and we would like to apply modified versions of our system to other languages.

Our choice of anchors, operations, and constraints represent one possible design for an Arabic CTF-TM. Other options, such as creating unlabeled dependencies and adding labels in subsequent operations, restricting clitic separation to a hand-crafted list of clitics, utilizing information from a dictionary or morphological analyzer, or following some sort of coarse-to-fine labeling scheme, are also possible, and we hope to investigate more of these options.

References

Sarah Alkuhlani and Nizar Habash. 2012. Identifying broken plurals, irregular gender, and rationality in arabic text. In *Proceedings of EACL 2012*, pages 675–685.

Mohammed Attia, Jennifer Foster, Deirdre Hogan, Joseph Le Roux, Lamia Tounsi, and Josef Van Genabith. 2010. Handling Unknown Words in Statistical Latent-Variable Parsing Models for Arabic, English and French. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 67–75.

Dan M Bikel. 2004. *On the parameter space of generative lexicalized statistical parsing models*. Ph.D. thesis, University of Pennsylvania.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1455–1465.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on Multilingual Dependency Parsing. In *Proceedings of the 10th Conference on Computational Natural Language Learning*, pages 149–164.

Jean-Cédric Chappelier, Martin Rajman, Ramón Aragüés, and Antoine Rozenknop. 1999. Lattice Parsing for Speech Recognition. In *Proc. of 6ème conférence sur le Traitement Automatique du Langage Naturel (TALN 99)*, pages 95–104.

Shay B Cohen and Noah A Smith. 2007. Joint Morphological and Syntactic Disambiguation. In *Proceedings of the EMNLP-CoNLL 2007*.

Michael J. Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and experiments with Perceptron Algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The Stanford typed dependencies representation. In *COLING 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*.

Jon Dehdari, Lamia Tounsi, and Josef van Genabith. 2011. Morphological Features for Parsing Morphologically-Rich Languages: A Case of Arabic. In *Proceedings of the Second Workshop on Statistical Parsing of Morphologically Rich Languages*.

Mona Diab. 2007. Toward an Optimal POS Tag Set for Modern Standard Arabic Processing. In *Proceedings of Recent Advances in Natural Language Processing*.

Yoav Goldberg and Michael Elhadad. 2010. An Efficient Algorithm for Easy-First Non-Directional Dependency Parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 742–750.

Yoav Goldberg and Michael Elhadad. 2011. Joint Hebrew Segmentation and Parsing Using a PCFG-LA Lattice Parser. In *Proceedings of ACL 2011*.

Yoav Goldberg and Michael Elhadad. 2013. Word Segmentation, Unknown-word Resolution, and Morphological Agreement in a Hebrew Parsing System. *Computational Linguistics*, 39(1):121–160.

Yoav Goldberg and Reut Tsarfaty. 2008. A Single Generative Model for Joint Morphological Segmentation and Syntactic Parsing. *Proceedings of ACL-08: HLT*.

Spence Green and Christopher Manning. 2010. Better Arabic Parsing: Baselines, Evaluations, and Analysis. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 394–402.

Nizar Habash and Owen Rambow. 2005. Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop. In *Proceedings of the 43rd Annual Meeting of Association for Computational Linguistics*, pages 573–580.

Nizar Habash and Ryan Roth. 2009. CATiB: The Columbia Arabic Treebank. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*.

Nizar Habash, Owen Rambow, and Ryan Roth. 2009. MADA+TOKAN: A Toolkit for Arabic Tokenization, Diacritization, Morphological Disambiguation, POS Tagging, Stemming and Lemmatization. In *Proceedings of the 2nd International Conference on Arabic Language Resources and Tools (MEDAR)*.

Jan Hajič, Otakar Smrz, Petr Zemánek, Jan Šnaidauf, and Emanuel Beška. 2004. Prague Arabic Dependency

Treebank: Development in Data and Tools. In *Proceedings of the NEMLAR International Conference on Arabic Language Resources and Tools*.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2011. Incremental joint pos tagging and dependency parsing in chinese. In *IJCNLP*, pages 1216–1224.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2012. Incremental joint approach to word segmentation, pos tagging, and dependency parsing in chinese. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 1045–1053.

Liang Huang and Kenji Sagae. 2010. Dynamic Programming for Linear-Time Shift-Reduce Parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1077–1086.

Terry Koo and Michael Collins. 2010. Efficient Third-order Dependency Parsers. In *Proceedings of ACL 2010*, pages 1–11.

Seth Kulick, Ryan Gabbard, and Mitchell Marcus. 2006. Parsing the Arabic Treebank: Analysis and Improvements. In *Proceedings of the Treebanks and Linguistic Theories Conference*, pages 31–42.

Seth Kulick. 2011. Exploiting Separation of Closed-Class Categories for Arabic Tokenization and Part-of-Speech Tagging. *ACM Transactions on Asian Language Information Processing (TALIP)*, 10(1):4.

John Lee, Jason Naradowsky, and David A Smith. 2011. A discriminative model for joint morphological disambiguation and dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 885–894.

Zhongguo Li and Guodong Zhou. 2012. Unified dependency parsing of chinese morphological and syntactic structures. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1445–1454.

Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, Wenliang Chen, and Haizhou Li. 2011. Joint models for chinese pos tagging and dependency parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1180–1191.

Ji Ma, Tong Xiao, Jingbo Zhu, and Feiliang Ren. 2012. Easy-First Chinese POS Tagging and Dependency Parsing. In *Proceedings of COLING 2012*, pages 1731–1746, Mumbai, India.

Ji Ma, Jingbo Zhu, Tong Xiao, and Nan Yang. 2013. Easy-first pos tagging and dependency parsing with beam search. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 110–114, Sofia, Bulgaria. Association for Computational Linguistics.

Mohamed Maamouri and Ann Bies. 2004. Developing an Arabic Treebank: Methods, Guidelines, Procedures, and Tools. In *Proceedings of the Workshop on Computational Approaches to Arabic Script-based languages*, pages 2–9.

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

Mohamed Maamouri, Ann Bies, and Seth Kulick. 2012. Expanding Arabic Treebank to Speech: Results from Broadcast News. In *Proceedings of LREC 2012*.

Yuval Marton, Nizar Habash, and Owen Rambow. 2010. Improving Arabic Dependency Parsing with Lexical and Inflectional Morphological Features. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*.

Yuval Marton, Nizar Habash, and Owen Rambow. 2011. Improving Arabic Dependency Parsing with Form-based and Functional Morphological Features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1586–1596.

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.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the Errors of Data-Driven Dependency Parsing Models. In *Proceedings of the EMNLP-CoNLL 2007*, pages 122–131.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 Shared Task on Dependency Parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*.

Joakim Nivre. 2009. Non-Projective Dependency Parsing in Expected Linear Time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*.

Xian Qian and Yang Liu. 2012. Joint Chinese Word Segmentation, POS tagging and Parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 501–511.

Owen Rambow, David Chiang, Mona Diab, Nizar Habash, Rebecca Hwa, Khalil Simaan, Vincent Lacey, Roger Levy, Carol Nichols, and Safiullah Shareef. 2005. Parsing arabic dialects. In *Final Report, JHU Summer Workshop*.

Stephen Tratz and Eduard Hovy. 2011. A Fast, Accurate, Non-Projective, Semantically-Enriched Parser. In *Proceedings of EMNLP 2011*.

Stephen Tratz. 2011. *Semantically-Enriched Parsing for Natural Language Understanding*. Ph.D. thesis, University of Southern California.

Reut Tsarfaty. 2006. Integrated Morphological and Syntactic Disambiguation for Modern Hebrew. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 49–54.

Yue Zhang and Stephen Clark. 2008. Joint Word Segmentation and POS Tagging Using a Single Perceptron. In *Proceedings of ACL 2008*, pages 888–896.

Imed Zitouni, Jeffrey S Sorensen, and Ruhi Sarikaya. 2006. Maximum entropy based restoration of Arabic diacritics. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 577–584.